CERNET — A High-Speed Packet-Switching Network

Edited by J. M. Gerard
Propriété littéraire et scientifique réservée pour tous les pays du monde. Ce document ne peut être reproduit ou traduit en tout ou en partie sans l’autorisation écrite du Directeur général du CERN, titulaire du droit d’auteur. Dans les cas appropriés, et s’il s’agit d’utiliser le document à des fins non commerciales, cette autorisation sera volontiers accordée.

Le CERN ne revendique pas la propriété des inventions brevetables et dessins ou modèles susceptibles de dépôt qui pourraient être décrits dans le présent document; ceux-ci peuvent être librement utilisés par les instituts de recherche, les industriels et autres intéressés. Cependant, le CERN se réserve le droit de s’opposer à toute revendication qu’un usager pourrait faire de la propriété scientifique ou industrielle de toute invention et tout dessin ou modèle décrits dans le présent document.

© Copyright CERN, Genève, 1981

Literary and scientific copyrights reserved in all countries of the world. This report, or any part of it, may not be reprinted or translated without written permission of the copyright holder, the Director-General of CERN. However, permission will be freely granted for appropriate non-commercial use.

If any patentable invention or registrable design is described in the report, CERN makes no claim to property rights in it but offers it for the free use of research institutions, manufacturers and others. CERN, however, may oppose any attempt by a user to claim any proprietary or patent rights in such inventions or designs as may be described in the present document.
ABSTRACT

A general mesh-structured high-speed computer network has been designed and built. This network provides communication between any pair of connected user computers over distances of up to 6km and at line speeds of 1 to 5 Mbit/second. The network is composed of a communication subnet providing a datagram service, complemented by tasks in the connected machines to implement an end-to-end logical link protocol.

Details are given of the overall structure as well as the specific modules of which the system is composed.
# CONTENTS

<table>
<thead>
<tr>
<th>PREFACE AND ACKNOWLEDGEMENTS</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDITORIAL PREFACE</td>
<td>x</td>
</tr>
<tr>
<td>I.  INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>I.1 Background</td>
<td></td>
</tr>
<tr>
<td>II. AN OUTLINE OF CERNET</td>
<td>3</td>
</tr>
<tr>
<td>II.1 The Computers</td>
<td>3</td>
</tr>
<tr>
<td>II.2 The Data-links</td>
<td>6</td>
</tr>
<tr>
<td>II.3 The Protocols and Software</td>
<td>6</td>
</tr>
<tr>
<td>III. PROTOCOLS</td>
<td>9</td>
</tr>
<tr>
<td>III.1 Line protocol</td>
<td>9</td>
</tr>
<tr>
<td>III.2 End-to-End protocol</td>
<td>11</td>
</tr>
<tr>
<td>III.3 File Access protocol</td>
<td>12</td>
</tr>
<tr>
<td>III.4 Virtual Terminal protocol</td>
<td>13</td>
</tr>
<tr>
<td>IV. COMMUNICATION SUBNETWORK HARDWARE</td>
<td>15</td>
</tr>
<tr>
<td>IV.1 Type of subnetwork</td>
<td>15</td>
</tr>
<tr>
<td>IV.2 Node computers and peripherals</td>
<td>16</td>
</tr>
<tr>
<td>IV.3 CERN data-link hardware</td>
<td>16</td>
</tr>
<tr>
<td>IV.4 User hardware connection</td>
<td>17</td>
</tr>
<tr>
<td>IV.5 Reliability and diagnostic aids</td>
<td>18</td>
</tr>
<tr>
<td>V. SOFTWARE EVOLUTION AND SUPPORT AIDS</td>
<td>20</td>
</tr>
<tr>
<td>V.1 Program preparation and loading</td>
<td>20</td>
</tr>
<tr>
<td>V.2 BCPL usage</td>
<td>23</td>
</tr>
<tr>
<td>VI. COMMUNICATION SUBNETWORK SOFTWARE</td>
<td>26</td>
</tr>
<tr>
<td>VI.1 Basic system</td>
<td>26</td>
</tr>
<tr>
<td>VI.2 Software overview</td>
<td>28</td>
</tr>
<tr>
<td>VI.3 Principal tasks</td>
<td>29</td>
</tr>
<tr>
<td>VII. HOST CONNECTIONS</td>
<td></td>
</tr>
<tr>
<td>VII.1 IBM</td>
<td></td>
</tr>
<tr>
<td>VII.1.1 Basic aims of the CERNET connection</td>
<td>31</td>
</tr>
<tr>
<td>VII.1.2 Factors that determined design and implementation</td>
<td>31</td>
</tr>
<tr>
<td>VII.1.3 An overall view of the Transport Manager</td>
<td>32</td>
</tr>
<tr>
<td>VII.1.4 Some details of the Transport Manager implementation</td>
<td>34</td>
</tr>
<tr>
<td>VII.1.5 The Transport Manager user interface package</td>
<td>35</td>
</tr>
<tr>
<td>VII.1.6 An overall view of the File Manager</td>
<td>35</td>
</tr>
<tr>
<td>VII.1.7 The file access user interface package</td>
<td>36</td>
</tr>
<tr>
<td>VII.1.8 Miscellaneous CERNET jobs running on IBM machines</td>
<td>36</td>
</tr>
</tbody>
</table>
VII.2 CDC
   VII.2.1 Hardware connection
   VII.2.2 Software components
   VII.2.3 The network Interface program
   VII.2.4 Satellite coupler testing
   VII.2.5 Further information

VII.3 File Manager operations

VII.4 Real-time programs (IBM)

VIII. USER CONNECTIONS

VIII.1 The portable Transport Manager
   VIII.1.1 The portable TM kernel
   VIII.1.2 System-dependent routines

VIII.2 PDP-11 connections

VIII.3 VAX-11 connections

VIII.4 Nord-10 connections
   VIII.4.1 The portable Transport Manager (TM)
   VIII.4.2 The Transport Manager User Interface Package
   VIII.4.3 The File Access User Interface Package
   VIII.4.4 File Access and Job Submission

VIII.5 HP21mx connections
   VIII.5.1 Applications
   VIII.5.2 The Transport Manager and User Interface Package
   VIII.5.3 The File Access User Interface Package
   VIII.5.4 General background program HPMG
   VIII.5.5 The interactive program UMG

VIII.6 Connection of OMNET

IX. NETWORK SERVICES AND FACILITIES

IX.1 The probe packet

IX.2 Services provided to user computers
   IX.2.1 Name service
   IX.2.2 Simple File Transfer

X. OPERATIONAL AIDS

X.1 The Control Centre
   X.1.1 Display facilities
   X.1.2 Monitoring facilities
   X.1.3 Control facilities
   X.1.4 Monitoring methods

X.2 Message logging and analysis

X.3 Remote console

XI. OPERATIONS

XI.1 Coordination

XI.2 Performance statistics

XI.3 Operational weaknesses
XI. NETWORK USAGE
   XI.1 On-line analysis 60
   XI.2 Data sampling 60
   XI.3 Individual file transfers 61
   XI.4 Disk backup 62
   XI.5 Inter-mainframe traffic 62
   XI.6 Miscellaneous 63

XIII. RELIABILITY, PERFORMANCE AND THROUGHPUT 65
   XIII.1 Reliability 65
   XIII.2 Performance 66
   XIII.3 Throughput 67

XIV. FUTURE DEVELOPMENTS 68
   XIV.1 Software 68
   XIV.2 Data-Link Interfaces 68
   XIV.3 Connection to other networks 69

REFERENCES 70

APPENDICES

A: LIST OF NETWORK PROJECT NOTES (NPNS) 73

B: PRINCIPAL MODCOMP TASKS 78
PREFACE AND ACKNOWLEDGEMENTS

This report gives a technical description of CERNET, the CERN data communications network, and fulfills an undertaking given to the CERN Finance Committee at the time of contract adjudication for the project to make the designs generally available. It includes sections describing the hardware, the software and the utility programs of the Network. The authors also give some indications as to how the software was developed, why certain design decisions were taken and how they turned out in practice, as well as summarising the uses made of, and the performance obtained from, CERNET as at the end of 1980.

The text refers readers who want more detailed information to the Network Project Notes (NPN). These Notes are the technical documentation written during the project and they are referenced throughout the text by their NPN number. A complete list of the NPN titles is given in Appendix A. They are available on request, as long as stocks last, from the Data Handling Division Secretariat at CERN.

A development of this kind is of necessity a large collaborative effort of many people. Over the five years of the project some sixty people have contributed directly to some technical aspect or other of the realisation of CERNET. On average, the team has been made up of about seven full-time CERN staff, together with some part-time staff, and a number of visitors (Fellows, Scientific Associates and Technical Students) who joined the team for various periods of a few weeks to two years. A complete list of the people who have participated in the CERNET development is included, and it is a pleasure to thank them all for the contributions they have made to the work of bringing CERNET to its present stage of full and reliable operation.

A number of people have carried special responsibility for different aspects of the development and their technical and management skills have been vital in bringing the exercise to a successful conclusion. They are:

- J.D.Blake (CDC software)
- H.Davies (project leader to end 1976)
- J.M.Gerard (node software, network services)
- J.Joosten (data-links)
- M.G.N.Hine (initial feasibility study)
- I.McLaren (user support)
- C.J.Piney (node software)
- D.M.Sendall (user computer software)
- M.J.Sheehan (operations)
- D.Wiegandt (IBM software)

I would also like to record my thanks to J.M.Gerard, who took over at short notice for an extended period in 1977/78 when, due to illness, I had an enforced absence from CERN. He
successfully led the project through the critical part of its first phase, namely the introduction of the first user services of CERNET.

Hardware construction has been very effectively handled by the Data Handling Division Electronics Workshop. Our documentation has been well looked after by Mmes Martine and Monnet and the Divisional Secretariat, whilst this report has been produced with the help of the Document Reproduction Services.

Finally, I would like to acknowledge the help and continued support which the CERNET project has had from the Data Handling Division Leader, P.Zanella, and the Group Leaders T.Bloch, J.Ferguson, C.Jones, D.M.Sendall, C.Verkerk and D.O.Williams, from whose groups the people who worked on the network were drawn.

G.R.Macleod
CERNET Project Leader
EDITORIAL PREFACE

A number of persons have combined to write this report. Rather than trying to give exact details of authorship I prefer to simply give an alphabetically ordered list, together with an expression of thanks for their efforts:

J.D.Blake
J.M.Gerard
G.R.Macleod
J.Ogilvie
C.J.Piney
D.Samyn
M.Sheehan
A.Vascotto
D.Wiegandt

The report is the first of its kind to be produced using the Photon typesetting service of CERN. It is a pleasure to acknowledge the help of Ms M-T.Monnet in editing the source document, the kind assistance of A.Berglund, who provided the software necessary to generate computer input to the Photon, and D.Ball, D.Stungo, Ms K.Wakley and the documentation service for all the work leading to the production of this first report.

J.M.Gerard
CERNET Project Participants

Over the period 1975-1980 the following people have at some time contributed to the study, design and/or implementation of CERNET.

C.Adams  J.Anthonioz-Blanc  D.Bain
D.Ball  D.Barton  J.D.Blake
Ms C.Bloch  R.Bock  D.Botterill
S.Brodecker  H.Brodmann  D.Brown
T.Bruins  H.Davies  R.Dobinson
Ms M.K.Downie  B.W.Evershed  F.Fabiani
R.A.Fjellheim  F.Fluckiger  D.J.Garnett
J.M.Gerard  C.Harman  M.G.N.Hine
G.R.Hoffmann  G.Hunter  N.J.D.Jacobs
J.Joosten  Y.Karita  P.van de Kerk
F.Königshofer  J.G.Lee  G.R.Macleod
E.McIntosh  I.McLaren  M.Metcalf
J.Montuelle  J.Nerdel  J.Ogilvie
C.F.Parkman  R.Pieters  C.J.Piney
R.Platteaux  M.Richards  J.Richter
Ms E.M.Rimmer  J.Rochez  D.Samyn
M.Scire  B.Segal  D.M.Sendall
M.J.Sheehan  P.Sparmann  M.Standley
T.Streater  S.Steppel  A.Vascotto
R.P.Watts  S.Westin  Ms V.White
D.Wiegandt  I.Willers  D.Wiskott
CHAPTER 1

INTRODUCTION

I.1 Background

Practically ever since CERN began acquiring large mainframe computers it has been attempting to make their computing power available to the minicomputers in experimental areas. Early attempts to link directly on-line to the computer centre were made difficult by the inability of the mainframe operating systems to cater for the mixture of on-line real-time data analysis with off-line batch environment work and, later, terminal access. For this reason, the idea of on-line data acquisition and analysis in the computer centre was temporarily abandoned in favour of two alternative approaches, which were pursued in parallel.

One approach involved using the central computers only in batch mode, using a dedicated front-end to support data-links and file handling. This was implemented in a system called FOCUS [1] which was in operation from 1968 until the end of 1978. Here a CDC 3000 lower series computer was equipped with data-links to various experimental set-ups. By means of terminals connected to FOCUS the physicists could send data sample files to the CDC 3000 file base, manipulate source files, initiate transfer of jobs (including the data sample files) to the central CDC computers and retrieve output for inspection or printing. At its peak (1970-1975) FOCUS was handling about 20 simultaneous terminal users and about 10 data-links, plus three Remote Job Entry stations. However, its services were tending to become overstretched and it could not easily be extended to include the central IBM computers, installed in 1976.

In an alternative approach, for the very large OMEGA experimental facility, a medium size CII 10070 computer was purchased specifically to provide real-time data analysis and associated support facilities including terminals. The data communications function was performed by a network of PDP-11 computers called OMNET [2]. The CII 10070 was logically in the centre of this network, with the terminals being connected to the various PDP-11s. This system also lasted until the end of 1978, at which time the CII 10070 was discarded as outdated, expensive to maintain and not powerful enough. However, the OMNET PDP-11 network was retained and has been connected to CERNET.

During the mid-1970s it became clear that both FOCUS and the OMEGA data handling system would need to be replaced fairly soon. With the sophistication of modern mainframe computers and operating systems and the proposed acquisition of large IBM mainframes and mass storage facilities to complement the existing CDC mainframes, it was also felt that the various experimental facilities could benefit by being integrated into the main computer centre. In addition, one had to take into account the growth, both inside and outside CERN, of other computer networks constructed for particular purposes.

The decision was thus made in 1975 to construct a general purpose data communications network (now called CERNET) inside CERN, to be used for computer-computer
communications. The performance should be such as to allow data transfer at speeds comparable to that of writing data onto magnetic tape.

One important general criterion laid down from the beginning was that centralised recording of raw data was not an objective of CERNET. In other words, it was to remain standard policy that the recording, on magnetic tape, of raw data generated by physics experiments should be done on minicomputers local to the experiment. However, the possibility to send samples of the raw data to the central computers for analysis on a much shorter timescale than that obtainable by physical transfer of a magnetic tape was considered extremely important.

It was also clear that the lifetime of CERNET was likely to be at least 10 years. For such a timescale it is very difficult, if not impossible, to forecast the exact requirements of all the likely subscribers of CERNET. Therefore the definition of CERNET had to be widened to include general purpose facilities over and beyond those specific to the collection and analysis of event samples.

After completion of a preliminary feasibility study, the first phase of the CERNET project was authorised in December 1975, with the design aim to provide

i) Data-link connections from the SPS North Area to the computer centre for experiment data sample calculations.

ii) Switchable and extendable data-link features for medium speed traffic between computers on-site.

This was completed by the end of 1978, by which time the basic packet switching network was in regular service, with six switching node computers and twenty user computers communicating with the CDC and IBM central computers. The first user was the European Muon Collaboration experiment NA2, which used CERNET regularly from March 1978 to process data on the IBM 370/168, whilst the first user of the network services to the CDC 6400/7600 was experiment NA4 from September 1978.

The second phase of the development was to extend the CERNET service to the West experimental area and other parts of the site, and to augment the user services and operational facilities provided through the network. These included more extensive file access and transfer facilities, remote job entry from user computers and output retrieval, resource-sharing between the IBM and CDC systems and, finally, the control centre software for operation of the network. By the end of 1980, this second phase has been completed. The traffic handled by the network is about 2.25 giga-bytes per week and the trend shows a continuing increase. There are now fourteen nodes and about forty user computers, as well as twenty OMNET computers, communicating via CERNET, mainly with the central CDC and IBM systems, from all regions of the CERN site. The project has been completed within the initial authorised estimates of 3 MFr$\$ materials costs and sixty man-years of effort over the period 1975-1980.
CHAPTER II

AN OUTLINE OF CERNET

This chapter describes CERNET in outline and and introduces the various elements of the network (and terms used) for which detailed descriptions are given in subsequent chapters.

II.1 The Computers

At the end of 1980 CERNET consists of over 50 computers interconnected by high-speed data-links. It provides program-to-program communication between these computers, using a packet-switching technique for the transmission of messages between them.

The computers making up CERNET are either subscriber or node computers.

The subscribers are those connected to CERNET primarily in order to communicate one with another. Most subscribers are simply user computers, i.e. computers in physics or engineering applications which use services provided by, or through, CERNET. A few subscribers, however, have a special status as host computers in that they provide some kind of computing service to other subscribers.

The nodes, together with the data-links, form the communications sub-network. This provides the basic functions of accepting packets of information (in a standard format and up to 2046 bytes long) from subscribers, transmitting them via an appropriate route through the sub-network and delivering them to the subscriber to which they were addressed. Figure 1 shows a schematic view of the CERNET configuration as at end-1980 showing the main connections overprinted on a map of the CERN site. Each small dot represents a user mini-computer, usually in an experimental hall or development laboratory, connected by a CAMAC data link (thin lines) to a CERNET node computer. Each small square represents a pair of node computers, which are Modcomp Classics in the computer centre and Modcomp II/45's elsewhere. The inter-node connections are represented by the thicker lines, which represent two or more data-links. In the enlarged inset of the the computer centre the connections to the main hosts on CERNET, the laboratory's large CDC and IBM systems, are shown, together with node-pair connections and links to user computer either in the computer centre itself or in adjacent buildings.

Figure 2 shows schematically the details of CERNET connections in the computer centre, showing the mesh structure of the network which offers redundancy to allow for node or link failures. The circles represent the node computers (all Modcomp Classics) with their two character hexadecimal identifiers. The nodes 11 and 12 are for user computer connections, 17 and 18 for the main CDC and IBM connections, whilst 13 and 14 provide back-up for these latter. (The pairs 11-12, 14-17, 18-13 correspond to the node pairs shown as small squares in figure 1). Apart from the connections to the CDC and IBM systems which are made by channel couplers, all other connections are CERNET data links. The dotted lines represent a group of user links which can be switched from another node in the case of
node-failure. Thus, for instance, the six user links to node 14 can, if needed, be switched to node 13. The interconnection between nodes is such that any user link is always backed-up by
an alternative route to the CDC or IBM system in the event of a node or an inter-node link failure.

![Diagram showing network connections](image)

*Figure 2: Details of CERNET connections at the computer centre*

Each subscriber is connected to a node, whilst each node has at least two, and often three or more, connections to other nodes. Nodes have been installed in pairs and hardware link switches have been built to connect groups of links for subscribers to either node of a pair. In this way, alternative routing and some hardware redundancy has been built into CERNET in order to ensure a very high level of availability of the communications sub-network. This is necessary as the CERN accelerators and the central computer service run 168 hours per week for several weeks at a stretch during experiment periods, whereas the CERNET team is not staffed to provide more than a single shift, 5 days per week maintenance service. Thus in most cases of data-link or node failure a user will continue to obtain a connection through CERNET to a host by alternative routing, allowing time for the maintenance services to repair the fault.

There are fourteen nodes at present in service, eight of which are Modcomp Classic 7860 computers [3] and six are Modcomp II/45 computers [4]. The Classics are installed in the region of highest network traffic, which is the CERN Computer Centre, whilst the II/45's are installed in the North and West Experimental areas and the Experimental Physics Division
Laboratory area. In addition two II/45's are still reserved for hardware and system development. The furthest nodes are some 5 kilometres from the Computer Centre. Two more nodes (Classics) are being installed in the new Underground Experimental Areas for operation early in 1981. All of the nodes use a version of Modular Computer System's MAXCOM operating system [5] which has been extensively modified at CERN to meet CERNET specific requirements.

The principal hosts connected to CERNET are the large CDC and IBM systems in the Laboratory's computer centre. These are the CDC 6500 and Cyber 170/720 systems (which are the front-ends to a CDC 7600, itself not directly attached to CERNET) and the IBM 370/168 and 3032 systems. Practically the whole range of services provided by the Computer Centre is therefore available to the CERNET user computers. In addition to its role as a host, the IBM system may also be a user computer, as programs running in the IBM may take an active role in communicating with programs running in other subscribers on the network. The software to allow a similar mode of operation has not yet been made available on the CDC system.

The fifty user computers connected to CERNET are mainly mini-computers dedicated to some specific application. This may be a physics experiment, where the mini carries out the function of data acquisition or equipment control, or it may be in physics support, where the mini is used to carry out software development or equipment testing. The user computer may be any of the “preferred types” of mini-computer which are at present supported in CERN. These are Norsk-Data ND-10 or ND-100 systems with SINTRAN III, Digital Equipment PDP-11 systems with RSX-11M or VAX-11 systems with VMS, or Hewlett Packard HP21mx systems with RTEIVB

II.2 The Data-links

The data-links of the network have been designed and built at CERN. They are full-duplex links capable of transmission speeds of several Megabits per second over several kilometres, and use asynchronous serial transmission over twisted pairs in standard 6-POD TV cables. All of the data-links, whether inter-node or node-subscriber, are logically identical. However, the data-link interfaces for the subscribers are built in the form of CAMAC modules [6], whereas those for the Modcomp node computers are made in a form suitable for direct connection to the Modcomp Input/Output units. The use of CAMAC allows the same data-link interfaces to be used for all types of subscriber computers and is convenient in a laboratory where all the computers used in experiments have CAMAC installed. The CAMAC CERNET data-link modules are now commercially available. A description of the communications sub-network hardware is given in Chapter IV. Special arrangements for connection to CERNET were made only for the large CDC and IBM systems, for which channel-to-channel couplers supplied by Modcomp were used, and for the connection of the OMNET PDP-11 computer network. This has been connected through a pair of PDP-11's which act as a gateway between the two networks, and for which a PDP Unibus version of the CERNET data-link interface was built.

II.3 The Protocols and Software

Communications between programs in different CERNET subscriber computers is made by the exchange of messages between them. In principle this is quite simple, requiring three main steps. Firstly the two partners must agree to communicate and a transmission path, a logical link, be opened between them. Secondly the messages must be transmitted correctly or any error condition reported. Thirdly the logical link must be closed down at the end of the communication. In practice even the simplest communication will involve two subscriber computers and at least one node and two data-links. The opportunities for confusion and error
are legion, so to ensure an orderly communication requires the specification of sets of rules, or Protocols, which govern different aspects of each communication. These are defined in a hierarchy corresponding to different levels of complexity, any protocol only using the procedures defined in the levels below.

In CERNET three levels of protocol are apparent to the subscribers. At the lowest level the line protocol defines transmission of packets between adjacent computers. At the intermediate level the End-to-End protocol defines transmission of messages between two subscriber computers. At the highest level are the File Access protocol, defining file transfers and file operation between a subscriber and a host, and the Virtual Terminal protocol, defining communication between a terminal on a subscriber and a host. A description of the protocols is given in Chapter III.

The line protocol defines the sequence of control words and data which must be exchanged between two computers to transmit correctly a single packet of up to 2046 bytes of information, or to initiate a recovery procedure if an error is detected. It is, of course, based upon a lower level definition contained in the design of the physical hardware of the data-link, whose electrical and timing patterns in fact compose the signals making up the data and control bits of the packet as it is transmitted over the link. Thus in each computer in CERNET a set of routines implements the line protocol rules in order to send and receive packets over its links. The way in which this is done depends upon the type of computer and its I/O arrangements.

Software on the node computers enables them to send or receive packets over its data links, to send them on an appropriate route towards their destination, to keep and update status information on links and routes and to communicate with other nodes. In particular, information is exchanged in the form of packets between nodes in order to allow control and monitoring of communications subnetwork facilities. In this way, information on the availability of subscribers, routing of packets, loading of nodes, updating of network status, and aids to computer centre operations are provided. This general node software is described in Chapter VI.

The control centre is another important piece of node software. As its name suggests its main purpose is to provide the computer centre operators with an effective and convenient way of controlling and monitoring CERNET operations. The control centre, which is described in Chapter X, collects and updates information on network status, provides graphical (colour) display facilities for this information, and enables the operators to send commands to any of the nodes for control purposes.

Some node software, particularly that associated with the control centre or with network services (see Chapter IX), is only available on some nodes, and may be on different nodes at different times. In order to simplify network use and operations, such software, which may need to be addressed at any time by any computer in the network, is accessible in a virtual machine (node or host) with a permanent name. The routing software then sees that messages destined for these virtual machines is routed to the correct physical node computer.

The End-to-End protocol governs the dialogue for opening and closing logical links, for breaking down long messages into packets and re-assembling them after transmission, and for detection of, and recovery from, errors. It is implemented on each subscriber in a software package called the Transport Manager (TM), with which is associated a Transport Manager User Interface Package (TM UIP). By making function calls to this latter, an applications program on a user computer can gain access to CERNET. Commands available in this way are: Connect with the local Transport Manager; Open or close a logical link; Send or receive a message on a given link; Wait for completion of a TM UIP function.

A machine-independent Portable Transport Manager, written in the BCPL language, is available, and versions of the TM and TM UIP derived from this are available for all of the
"preferred types" of mini-computers mentioned earlier. Chapter VIII describes the software for the connection of user computers to CERNET.

The File Access protocol has been implemented on the CDC and IBM hosts, with restricted versions available for PDP-11, VAX-11 and DEC-10 computers. It is implemented on each host as a CERNET File Manager (FM) program. Associated with this is File Manager User Interface Package (FM UIP) implemented on each user computer. Again, by function calls to this FM UIP, a user program can issue commands to obtain the required file operation on the file base of the host computer. Commands available are: Connect or initialise; Specify host and open or close a logical link; Open, close, save or purge a single file; Submit a file as a job to a host; Get or put records in a sequential file; Retrieve error information. A detailed description of the CDC and IBM host connections to CERNET is given in Chapter VII.

Finally, the Virtual Terminal protocol has been implemented for the PDP-11 and Nord-10 subscribers, which allows terminals of these mini-computers to be used as terminals to the WYLBUR terminal system which runs on the IBM computers. The minicomputers obey a CERN-defined Virtual Terminal definition and drive the real terminal. They communicate through the CERNET Transport Manager with "bridge" software in the IBM system. The protocol is described in section III.4.
CHAPTER III

PROTOCOLS

The protocols defined for CERNET are:

i) The Line protocol
ii) The End-to-End protocol
iii) The File Access protocol
iv) The Virtual Terminal protocol

In order to prevent excessive data storage in the communication subnetwork, a consistent scheme has been adopted in the design of the protocols: data should never be sent before it is requested. This is known as a receive-master scheme, and simplifies decisions as to who should set timeouts, and who should take the initiative in the case of errors. In a strict receive-master scheme, the slave — i.e. the sender — can at most request guidance from the master — i.e. the receiver — in case of problems. The CERNET protocols follow this procedure.

III.1 Line protocol

The line protocol is designed to ensure the safe delivery of packets from one node to its neighbour. This implies that correct delivery is only reported if the packet arrives error-free. The line protocol provides flow control, error checking and recovery by the exchange of control-words and data.

A description of the operation of this protocol is given below, and full details of bit-settings, etc. are available in NPN 13.

The transfer of data is controlled by the exchange of control-words, and, since the protocol is full duplex, there must be some way of identifying whether a control-word refers to the sender or receiver process. This discrimination is done by the setting of one bit in the control-word. In the description that follows, the correct setting of this bit will be taken for granted.

Before data can be sent, its length (in 16-bit words) must be given to the sender and receiver hardware. This allows the interfaces to calculate the required Cyclic Redundancy Checksum, so that the sender can send it and the receiver check it after the last data word. This wordcount is carried in the 10 least significant bits of a control-word which also carries a bit specifying that this word holds a wordcount.

The flow control and error recovery are controlled by sending control-words containing specific bits, which can be combined to specify multiple actions. These control-words may also carry bits to indicate the occurrence of errors in the previous operation. The control functions are as follows:
Control-words sent by the Receiver (Commands)

ACK : Acknowledge. The transfer is complete. If no error bits are present, the transfer was successful. The presence of error bits means that even after the maximum number of retries, no transfer was possible. In either case, the sender can release the buffer containing the output data.

RWC : Ready for wordcount. The transmitter should respond by sending either the wordcount of the next transfer or an acknowledgement of receipt of this control-word if no data is available for output. Response to this control-word is checked by a timeout started when the RWC is sent.

RDA : Ready for data. The transmitter should send the data. The arrival of the data is checked by a timeout after receipt of a valid wordcount.

ERR : Error indication. The accompanying bits, which describe the error, should never travel separately. They are set to indicate the status generated by execution of the previous request, if any error occurred.

Control-words sent by the Transmitter (Responses)

WCT : Word-count. The length of the next output buffer, as described earlier.

ARWC : Acknowledge ready for wordcount. The response to RWC if no data is available. Once data becomes available, the actual wordcount can then be sent — as can the data if RDA was also received.

Two simple examples of the working of this protocol are given below:

Ex. I: Simple transfer:

```
Receiver
Input request initiated.

Transmitter

RWC-----------------------------

ARWC-----------------------------

Output request initiated.

WCT-----------------------------

RDA-----------------------------

DATA-------------------------------------------------

Input request terminated.

ACK-----------------------------

Output request terminated.
```
Fig. 2: Exchange with several simultaneous commands and one lost data transfer.

Receiver

Input request initiated.

Output request initiated.

Transmitter

RWC+RDA

WCT

DATA

Timeout expires.

RWC+RDA+ERR(timeout)

WCT

DATA

Input request terminated

ACK

Output request terminated.

As can be seen from the preceding example, the error recovery is simple and robust. These were the main criteria for adopting this scheme rather than any other that might be faster but less reliable. In practice this choice has been justified by the extremely low error rate in the system.

III.2 End-to-End protocol

Once the procedure for transmitting packets reliably between neighbours has been established, a means of communicating between subscribers, possibly across several nodes, must be designed. The software implementing this procedure is called the Transport Manager.

In a similar way to that in which the line protocol controls the communication by means of control-words, the end-to-end protocol relies on the exchange of special control packets. Control information and data can both travel in the same packet.

The design of a Transport Manager requires the definition of an addressing scheme, to identify both the relevant Transport Manager and the user of this Transport Manager. In addition, the rules for setting up and closing a communication must be defined, as well as those for data transfer and error checking and recovery.

A Transport Manager address is a 16-bit quantity, with the most significant byte representing the destination machine, and the least significant byte used to address the relevant Transport Manager in that machine. In practice, the lower byte is usually zero. The upper byte is further structured into two 4-bit values: region and machine. This scheme simplifies the naming of the various subscriber machines.

The communication subnetwork examines only the most significant byte for the routing.

Processes addressable via the Transport Manager are identified by a name of up to 8 characters.

The source and destination processes are identified on establishment of the communication: a 'logical link' is set up between the two entities whose names are carried in a 'request link' packet. This packet also carries the local 'logical link number' by which the communication will be known by the source Transport Manager. Once the communication is accepted, an 'accept link packet' is returned. This packet also indicates the number by which the accepting Transport Manager knows the link. These logical link numbers are included in all subsequent packets exchanged between the Transport Managers over this logical link.

The end-to-end protocol follows the receive-master principle by the exchange of 'message
windows' (not to be confused with the IBM WINDOWS software package mentioned later). The receiver indicates to the sender the range of message numbers that it will accept. The sender must acknowledge this window information, and can send, in correct sequence, the messages whose numbers have been authorised in this way. The receiving Transport Manager checks the sequencing of messages as they arrive.

A message may be longer than a packet, in which case the sending Transport Manager will fragment it. The receiving Transport Manager will check the fragment sequencing and reassemble the complete message before passing it on to the relevant process.

The successful receipt of a message is indicated by the transmission of a new 'window' whose lower bound no longer authorises the sending of the message just received.

Timeouts are set to protect the transmission of read requests — i.e. an increase in the window upper bound — and message transmission. If a read request is not acknowledged, it is repeated. If a transmitted message is not acknowledged, a control packet is sent to the receiver to ask the status of the last transmission, i.e. whether it was the data or the acknowledgement that was lost. Duplicated packets are ignored. Any other error in the packet sequencing entails the initiation of a retransmission request, and the faulty message is sent again. In this way, the receiver controls the entire dialogue. In case of an excessive number of retries, the communication is terminated.

A communication is terminated by either party sending a close link request to the other. This request should also be acknowledged and is checked by timeout.


III.3 File Access protocol

In any computing environment it is normal that data be grouped together in the form of files. There is a tremendous variety of formats for such files, resulting from such factors as the type of information in the file, the storage medium used and the access method used to make the data available to the programmer. In addition, there are the complications due to the use of computers from different manufacturers.

The basic idea of use of CERNET is to allow information exchange between computers of different types. In particular, it is required that some computers, called hosts, offer a service by which remote subscribers can access the host's file base in a consistent manner. This is the service offered by the File Manager.

The File Manager must be able to deal with several simultaneous requests, and distinguish the particular requirements of each connected user program. This can only be done by including an extra protocol over and above the standard end-to-end protocol.

What is provided in CERNET (in its widest sense) is a file access protocol, allowing any user program to connect to any host, identify itself and the file which it is interested in and then perform various operations on that file. The operations include read, write, open, close and so on. The design of the file access protocol was done at a very early stage of CERNET. This is logical when one realises that it is the service of interest to almost every user. It has since been revised in various small ways, but the original concepts and ideas remain valid.

The file access protocol is defined in NPN 4, which also contains a good explanation of the design aims and limitations. In general that NPN provides the best overview of the protocol. However, there are some general remarks worth making here.

One main objective is to define a very limited number of data types. The obvious example is that of a character file in whatever internal code is chosen for any particular host. This considerably simplifies life for the user, who does not really want to be over-concerned with the details on the chosen host.
A second objective is to define a very limited number of device types. This essentially says what sort of file is involved, irrespective of the data type. Of course the data type must in practice sometimes be compatible with the device type if the latter is for instance a line-printer. Again the number of device types is very limited to keep life simple.

Finally the file can be opened in several possible modes, according to what the user intends to do with the file (read/write etc.).

The protocol as originally defined was essentially a handshake type, in which each individual transfer had to be explicitly requested by the user. When it became clear that data transfer rates would sometimes be a critical factor it was realised that the handshake could be removed from the file access protocol level by restricting the use of a single logical link to one file at a time. In other words the File Manager may deal with input/output for an opened file by means of the ‘window’ information carried at the level of the end-to-end protocol. This has the disadvantage that the File Manager cannot always send an error message in case of problems, but the advantage of increased speed. In fact for binary-type files, in which the unit of transfer is a complete record, the actual data transfer phase uses only the end-to-end protocol, thus allowing the sophisticated user to do double-buffering of data and to get a pipeline effect for data transfer if so required.

In practice the file access protocol has been almost exclusively used to write programs for the transfer of complete files. Its use has shown that a high data rate can be achieved and that a limitation on the number of types of file which can be handled is perfectly acceptable to the user community. In case of some requirement for new types of file it is relatively easy to augment the number of types.

III.4 Virtual Terminal protocol

As the number of minicomputer hosts connected to CERNET increased, it became apparent that a requirement existed to communicate, via CERNET, between terminals attached to these minicomputers and the terminal systems running on the large central hosts. This was particularly the case with PDP-11 minicomputers wanting to use Wybur on the IBM. As a result, the elements of a Virtual Terminal Protocol were defined so that this communication could proceed in an orderly manner. The detailed definition is in NPN 78.

The Virtual Terminal Protocol (VTP) is the set of rules for the process-to-process communication which drives a Virtual Terminal, where one process is the host computer terminal system and the other process is a program in the user computer which is driving the user’s real terminal. Thus the VTP definition is dependent on the definition of the Virtual Terminal and vice-versa.

The Virtual Terminal (VT) is an imaginary device which provides a standard representation of a canonical terminal by means of the VTP. The VT should not be concerned with the highly device-dependent characteristics of real terminals. e.g. timing considerations, specific ways to erase the screen, etc. Instead, it is the task of the remote program driving the real terminal to map the VT onto the real terminal based on local knowledge of its characteristics. The VT is thus the ideal terminal in an imperfect world.

The following points were considered to be relevant in the design of the VTP/VT:

i) The VTP must be simple, keeping the remote program which handles the real terminal easy to write and as small as possible.

ii) Efficiency of CERNET link usage is not important due to the low data rates.

iii) The VT is an ASCII device.

iv) The VTP is symmetric. There is an agreed list of Item Types which can be exchanged between the two communicating processes and both processes must be able to accept any Item on the agreed list.

Data is exchanged in the VTP by Messages using the Transport Manager user interface to
CERNET. Each message is composed of one or more Items. Each Item is made up of a one-byte item-code field followed by a one-byte length field followed by a variable-length parameter field. The length indicates the number of bytes in the parameter. This structure permits the program driving the real terminal to process messages in a data driven way, without the need to inspect each character of a text string. In addition, item codes can be processed using a simple jump table.

Two Communication Modes are defined. The Alternate Mode is basically that only one partner is allowed to transmit text-type Items at any given time. The alternation is controlled by sending <turn> and <plse> Items. The Free Running Mode allows either party to send text-type Items as and when they wish subject to flow control rules. Two Items, <hold> and <cont>, are defined for handling flow control. At present all implementations use the alternate mode.

The scope of the VT can be very large due to the widely differing features provided by the competing manufacturers of real terminals. Further to this, the imagination for future devices is almost unlimited. The VTP, as defined, is open-ended in the sense that it is very easy to add new Items for new features. However, this fact coupled with the knowledge that many real terminals would only possess a subset of the facilities of an all embracing VT led to the definition of the Verification process. This process is a simplified negotiation process in which both partners agree to a list of Item types which they are prepared to exchange.
CHAPTER IV

COMMUNICATION SUBNETWORK HARDWARE

The complete CERNET network as described in this report includes a full range of services available to the subscribers. However, any such network invariably requires at the heart of it a mechanism to provide communication facilities for these subscribers. Such a mechanism is often called the communication subnetwork. This chapter gives an explanation of the various entities which form the CERNET communications subnetwork.

IV.1 Type of subnetwork

There are many possible options when choosing to construct a communication subnetwork. For instance there is a choice between circuit-switched networks and packet-switched networks. The latter may be virtual call or datagram types. The topology may be ring-shaped, star-shaped or mesh. A discussion of all these options is beyond the scope of this report, which is restricted to the choice made, and its reasons, for CERNET.

The general principle adopted was to base the network upon a set of interconnected minicomputers known as nodes. High speed communication links should exist between these nodes and also from the nodes to the various other computers, both large and small, to be connected to CERNET.

The type of method was chosen as a packet-switching one, rather than circuit-switching, because it was felt that packet-switching was both general purpose and very flexible. It was also hoped that packet-switching could be made to work sufficiently fast for the proposed types of application by a suitable choice of hardware and protocols: this hope has, so far, been realised. The general topology of the network was foreseen as mesh-type, with particular data-links being included according to either traffic requirements or safety back-up needs.

The particular type of packet-switching network chosen was one in which the subnetwork is willing to accept blocks of data (of a limited size) i.e. "packets", from one subscriber and will try to deliver them to a destination subscriber. The head of each block of data is assumed to contain in a fixed place the identification of the destination subscriber. Each such packet is treated as a single entity, without the subnetwork knowing to what extent it forms part of a wider communication between the subscribers. This means that the subnetwork is essentially a datagram network.

The overall view of the communication subnetwork is thus a mesh network of packet-switching node computers. Subscriber computers may be connected to one or more nodes. Each node is capable of accepting suitably addressed packets and forwarding them via the most appropriate route towards their eventual destination.

An important feature particular to CERNET is that the route between any two computers connected onto CERNET is fixed as long as no external event makes a change necessary. This is made possible by the fact that CERN is a private site on which transmission cables can
be laid wherever the data traffic warrants them. The consequence, which simplifies the usage, is that delivery of data packets between two subscribers is normally made in the order in which the data packets enter the subnetwork.

IV.2 Node computers and peripherals

The first essential was to choose a computer for CERNET nodes. The requirements for the nodes, which are basically for data input/output but not much central processor power, suggested that a minicomputer should be adequate. However, a particularly important requirement was for high performance channel connections to the IBM and CDC mainframe computers of the CERN computer centre.

The choice of computer for CERNET was, at the time, a choice between various minicomputers which could provide a variety of performance and hardware devices but in general no software specifically designed for high-speed packet-switching. After the usual type of evaluation, benchmarks, cost comparison, etc., it was decided to do the implementation on Modular Computer Services (Modcomp) Model II series computers [4]. Since 1979 Modcomp Classic 7860 computers [3] have also been introduced as network nodes.

One particular advantage of the Modcomp computers was the availability of channel connectors to both CDC 6000 series computers and IBM 370 series computers. Another positive point was a 4-port memory controller allowing simultaneous accesses from the central processor, CDC/IBM couplers and data-link controller. The data-link controller was an already available Modcomp special product meant to be used with non-Modcomp equipment.

The normal use of minicomputers requires somewhere a support computer capable of storing and editing source files, assembling individual program modules, linking individual modules into a complete core load image and assisting in the loading of that image into the minicomputer. Such a support computer is frequently a minicomputer of the same type but equipped with a large range of peripherals (rotating mass storage, terminals etc.). However, for CERNET, this function of support computer has been provided by existing mainframes. Thus the general configuration of the Modcomps has been kept simple in all cases.

The general configuration for all of the Modcomps consists of the minimum equipment necessary for a stand-alone system. Individual Modcomps vary somewhat in the amount of memory. Four Modcomps each have a CDC coupler and an IBM coupler. This is intended to permit two possible connections to each of the four mainframe computers (two CDC and two IBM). All Modcomps have a 30 character/second hard copy console for operator input/output, plus a floppy disk unit for local loading of programs and a simple 9600 baud serial interface for utility connections such as two Tektronix 4027 colour displays. The interfacing of inter-computer serial data-links to Modcomps is via a data-link controller (a readily available special product for Modcomp II's) or an input/output processor (an IOP, standard for the newer Modcomp Classic range). The actual links are discussed in more detail in the next section.

IV.3 CERN data-link hardware

As a result of CERN's previous work on high speed data communications [9], there was already a high level of technical expertise on the construction and use of high speed links. Also, there was already a large number of standard high-quality twisted pair cables over much of the site. Thus it was decided that, regardless of computer or type of network, the actual data-links would be designed at CERN.

The basic specification for the data-links was for a full-duplex connection working at a basic clock-rate of 2.5 megabits/second, with the possibility of a later increase to 5 megabits/second over short distances. The technology used is asynchronous serial transmission based on units of one 16-bit word framed by one start bit and one stop bit. In
addition the goal of efficient full duplex working requires the interleaving of control-words and data words, which is achieved by adding on an extra control/data bit. Finally there is a parity bit per word, which means that each 16-bit word is accompanied by four extra bits, thus bringing the effective line speed to 2 million data bits per second.

The distances over which the data-links must work varies from a few metres to a few kilometres. Whenever the total distance between two linked computers exceeds 30 metres, each must also have a data set to boost incoming balanced signals for long distance transfer. For distances above 1.5 kilometres, it is necessary to insert repeaters at intervals, in order to reshape and relock the bit stream.

For any single full-duplex data-link, therefore, each of the two Modcomps at the ends of the link will have a receiver module using two wires, a sender module using another two wires and usually a data set. As will be explained later, a connection from a Modcomp to a minicomputer uses exactly the same data-link interface in the Modcomp and a compatible CERN-built interface in the minicomputer.

The actual design of data-link interface modules can in theory be independent of the line protocol to be employed for data transmission. However, the modern concept is to treat the two in an integrated manner such that some parts of the line protocol software can be migrated into the hardware. This latter approach was already being adopted in the design of the data-link interfaces. Thus, for instance, the hardware will intercept some control-words of a particular format in order to set its own internal registers. In addition it will calculate and check the value of a Cyclic Redundancy Checksum (CRC) for each complete block of data transferred.

There is a general concept in the whole of the network that the computer destined to receive data is the one which controls initiation of data transfer. This is known as the receive-master system. A corollary of this is that, when data transfer does begin, the receiver must be willing to accept the specified data as fast as it arrives. At the level of data-link interfaces, this is done by having a First-in First-out (FIFO) buffer for each receiver module. This buffer is large enough to take the largest single block of data allowed to be sent (1023 16-bit words) at the full line-speed regardless of whether the CPU of the Modcomp is busy or idle. DMA transfer into main memory can then be initiated by the CPU at a suitable time, so that the FIFO buffer can be regarded as external memory of the Modcomp.

A further feature of every data-link is a bootstrap program in Read-only Memory (ROM). Execution of this program can be initiated by the arrival of a specific control-word over any data-link, and the program will then start a bootstrap sequence which leads to the complete Modcomp core-load over the specified link. This is called "Remote Fill", and the object is to allow any Modcomp to force loading of its neighbouring node if it considers this to be necessary.

More detailed information on the hardware of the Modcomp links is given in NPNs 25 and 37, whilst the line protocol is outlined in Chapter III on protocols and described in detail in NPN 13.

IV.4 User hardware connection

Viewed from the CERNET nodes, the hardware connecting most user computers is no different from that linking two CERNET nodes together. In other words it is the high-speed asynchronous serial full-duplex interface described earlier.

In practice, the interface to most user computers is via a CAMAC [6] interface. For this, CERN has developed a pair of CAMAC modules (a sender and a receiver) which fit into a normal CAMAC system crate. In addition, it is necessary to have a data set between the CAMAC modules and the 4-wire cable leading to the nearest CERNET node, since this cable is invariably longer than 30 meters.
The details of the CERNET CAMAC link are given in NPN 76.

The great advantage of such a CAMAC link is that all minicomputers used in experimental data acquisition and treatment tend to have a fairly standard CAMAC setup, as well as utility routines to enable programs to use CAMAC equipment. In fact there are a standard set of ESONE FORTRAN calls [10], to enable CAMAC functions to be performed, available on most minicomputers at CERN.

For the particular case of the connection to the OMNET network of PDP-11 computers (see Chapter VIII), the two PDP-11/10 computers which act as gateways do not have CAMAC. It was therefore convenient to construct a CERNET-compatible interface which connects directly onto the Unibus. Details of this interface are given in NPN 34.

IV.5 Reliability and diagnostic aids

To achieve a high availability of the communications subsystem, special attention has been given to redundancy and maintainability.

![Diagram of dual Classic node](image)

Figure 3: Topology of dual Classic node

All important nodes are duplicated and supplementary hardware has been designed to make use of this duplication without manual intervention. The 'busswitch' for instance, allows switching of a block of seven links that connect normally to an Input-Output Processor (IOP) of a CLASSIC computer from one IOP to another. There are four IOP's on a CLASSIC, of which three are available for data-links. The topology of a dual node, each with a busswitch, is given in figure 3: On each machine, the first block of seven links is directly connected to IOP2. The second block is connected via the busswitch to IOP3 to which the free IOP4 of the other node is connected as well. If one node breaks down, the second block of links can be switched to the free IOP on the other node. Only the first block of seven links will then be unavailable, but these are generally less important (another path is possible or the connection has a low priority). The switching is done under program control but can also be done manually by flipping a switch.

Another important feature is, that every link can be tested by an engineer without
stopping the activity of the node in which it resides. A link is first isolated from the system by means of an operator command, then the engineer loads a hardware test task into the system and activates the task for the link to be tested. The task runs at the lowest priority and takes all the spare time of the system to produce error statistics. To test a complete connection between two nodes, the engineer loads in each node a test task communicating with the other. If a link is shown to be faulty, it can be replaced without stopping the node.

The long line transmission equipment (data set) has also some features to enhance the system:

![Diagram of data sets](image)

*Figure 4: User with two entries*

i) A particularly important user can be given two entries in the communications subsystem. Instead of giving him two complete links, which is inconvenient and expensive, the data set at his double node can switch the link from the user computer between two different links residing in one or two nodes (see figure 4). This caters for the breakdown of a link at a node or the node itself if not equipped with a busswitch (the Modcomp II does not allow for a busswitch due to its tight timing).

ii) A data set is also able to detect if a user computer or a node is powered down or physically disconnected. This information is propagated to the other end and made available as a status bit to the software. If any repeaters are included in this path, they relay this information as well.

iii) Each data set has a switch by which the serial data can be looped back to its originator. This feature allows for testing a part of a complete link between two computers and simplifies fault finding.

iv) A data set module can be replaced without powering down the crate and without affecting the operation of the other data sets in the same crate.
CHAPTER V

SOFTWARE EVOLUTION AND SUPPORT AIDS

Once a decision has been made concerning the hardware configuration of the node computers, the whole question of software languages, systems, support etc. must be answered. Clearly, one attempts to use as much as possible existing software and support services, whether provided by the manufacturer or the local computer centre. However, the overall objectives and design goals may make it necessary to develop extra software or services, or to modify existing but inadequate software.

In the case of CERNET, it was implicit in the choice of hardware configuration that software support would be obtained from a separate computer system. The main advantage of this approach is the simplicity of the Modcomp configuration and the avoidance of a separate Modcomp hardware/software configuration for program preparation. However, it does require the provision of software on the supporting system, such as cross-assemblers, linking loaders, down-line loading programs etc. This is a field in which CERN already has much experience and existing software on which to build.

One important decision, taken very early on, was to avoid assembly language programming wherever possible. It was therefore necessary to choose a high-level programming language which would nonetheless allow the programmer to access the Modcomp memory directly when necessary. The BCPL language [11] was chosen for this purpose, and extended use has been made of it, not only for Modcomp software but also for system preparation software on the support system. In addition, the major module of software to allow the interaction between processes running in different CERNET-connected computers is written in BCPL. All of this is possible because of the high degree of machine-independence of BCPL and its availability on the complete range of computers at CERN.

V.1 Program preparation and loading

The original choice of a support computer system for the Modcomps was between the CDC 6000 series computers and the CII 10070 computer used to support the OMNET network of PDP-11 computers. The latter was chosen because its support role for the PDP-11 computers meant that there was already some software for cross-linking assembled program modules and down-line loading.

The facilities already provided by the CII 10070 complex included a file base for source and object module files, a terminal access system including good file editing, job submission to, and output retrieval from, the CII 10070. There were also cross-assemblers, a link editor with library search facilities, and a relatively simple object module format for both absolute and relocatable programs. All of this software was for the purpose of supporting the various PDP-11 computers of OMNET, and allowing down-line loading of absolute or relocatable modules. Much of it could be used, in original or modified form, for the support of
Modcomps, with the reasons for modifications almost always due to the byte-addressing, byte-swapping feature of PDP-11s.

The facilities which had to be written essentially from scratch were a cross-assembler for Modcomp assembly language programs and a cross-compiler for Modcomp BCPL programs. The speed with which these facilities were provided was due to the technical expertise of the people involved and the fact that both could be written in BCPL.

The cross-assembler was the first necessity. Its original implementation was fairly simple, but later additions of conditional assembly and macros made it powerful and suitable as a model for subsequent cross-assemblers for a variety of mini and microcomputers.

The first use of the cross-assembler was to assemble stand-alone programs and produce an object module file which could simply be punched out on paper tape for loading into a Modcomp via the paper tape reader included on each of the first four Modcomps bought by CERN. Initially, the paper tape had to include a standard Modcomp bootstrap loader on the front, although later this bootstrap loader was taken from floppy disk.

The first, very important, stand-alone program was an extension of the standard Modcomp utility program. This is a program normally held on the first track(s) of a floppy disk, and which, in the original form provided by Modcomp, includes utility routines for reading/ writing/copying floppy disks. The extension facilities included a package consisting of four routines (paper tape loader, down-line loader, core to disk dump and core to console dump) which were self relocating to the high end of Modcomp memory so that they could be retained there even with the standard MAXCOM system running. There were also facilities to do a down-line dump from floppy disk to the CII 10070, as well as allowing the Modcomp console to act as a terminal console to the CII 10070.

In the choice of peripherals for the Modcomps, it was decided that each Modcomp should have its own particular connection to the supporting CII 10070, so as to allow for down-line loading/dumping independently of the other Modcomps. The alternative of having only a subset of Modcomps connected, which would have been the case had the CDC 6000s been used for support, would have been difficult to use without a form of inter-Modcomp connections. Of course, once a working version of the network software exists, the communication subnetwork itself provides such connections.

The physical connection of each Modcomp was a simple serial interface link, operating at 9600 baud, connected to a PDP-11 which in turn could access the CII 10070. The PDP-11 essentially saw the Modcomp as a half-duplex character input/output terminal. Down-line loading or dumping only required a rudimentary dialogue between the Modcomp and the PDP-11, for which some PDP-11 software had to be written. The PDP-11 had to be told of the name of the file in the CII 10070, after which the data transfer could be carried out on a simple block by block basis.

The format of the object module files to be down-line loaded from the CII 10070 to the Modcomp was essentially that already existing for PDP-11s, i.e. a sequence of blocks each containing load address, byte count, data and checksum. The addition of relocation information blocks allowed for down-line loading of overlays at a later stage, with relocation being done in the PDP-11 after some additions to the initial Modcomp to PDP-11 dialogue concerning module length and required base address.

The preparation of files for down-line loading to a Modcomp thus involved the use of the cross-assembler (later also the cross-compiler for BCPL) to produce relocatable or absolute object modules in a CII 10070 object module format suitable for subsequent use of the standard CII linking loader. At this stage, the Modcomp was thought of as a 32-bit computer, although of course only the least significant 16 bits of each word were actually relevant. When the complete relocatable or absolute program module had been created, a separate program converted it to the final, much simpler, blocked format object module suitable for down-line
need know nothing. In practice, for reasons of efficiency, the addresses of a small number of
utility subroutines are held in fixed core locations, and the code generator may use indirect
jumps to these when necessary. These utility subroutines are concerned with procedure
entry/exit and variable length shifting.

The next immediate requirement is for a run-time library. Much of such a library is fairly
standard, such as the interpretation of the format string of a WRITEF call. However, the
problem ultimately becomes that of interfacing a number of low-level routines to MAXCOM.
Two such low-level routines are clearly the character input/output routines RDCH and
WRCH. The interfacing problem is solved by having a run time library routine, of necessity
assembly code, which simply calls on the MAXCOM Request Executive (REX) service. By
this means, not only does the run-time library have access to the particular REX calls for
input/output, but also any user task can have access to any REX service.

The size of the run-time library and the large number of tasks, each to be written as a
self-contained BCPL program, made it clear that tasks could not each have their own personal
copy of the run-time library. Thus the run-time library is completely re-entrant, using the stack
of the calling task as work-space where necessary. In particular some of its routines (including
REX) need to create and execute code, and this also is done in the stack.

The run time library also reflects the requirements of the majority of the network tasks, as
does the use of the global vector in the 0-100 range. As an example, when a task is activated
in MAXCOM (see section VI.1) it is normally because there is a queue vehicle on its work
queue. For BCPL tasks, the address of this queue vehicle is put into a particular global
(called VEHICLE) before the BCPL code if the task is executed. Of course, the programmer
must know what action to take according to the contents of the vehicle.

For a BCPL task to be integrated into a MAXCOM system, the compiled code of the
task must obey certain rules. For instance the start, end and default transfer addresses must
be defined as external symbols in order to be correctly entered into various relevant tables, and
the task must start by calling an initialisation routine to set up the globals, stack etc. Rather
than adapting the code generator to do this, the system-independent code produced by the
code generator is sandwiched between two assembly code modules specifically created for the
particular task. These modules are normally short and simple, but can when necessary include
procedures available to the BCPL task via addresses in globals. This "sandwich" procedure is
also used to create the run-time library itself, since it is rather like a program without a
START procedure.

During the course of development of the network software it has become clear that quite
frequently a particular set of procedures are common to all of a subset of tasks. An obvious
possibility is to insert the procedures into the run-time library, with their addresses in globals
in the 0-100 range. However, in some specific cases, this would not be ideal, since not only
would there soon be no further available globals in this range, but also on occasions the
complete subset of tasks is not required on a particular Modcomp, and so it would be wasteful
to include these procedures. To solve this problem, the concept of a subsidiary library was
created. Such a subsidiary library may be shared between a particular set of tasks, using a
commonly-agreed set of globals in the region above 100 as a linkage mechanism.

A further interesting and original solution to a particular problem has been the treatment
of what are called processes. The particular problem is that a single task may be required to
handle a number of concurrent processes. Any process may at times be held up waiting for a
particular event, but the task must not be held up since that would block other processes. A
typical example is the symbiont task which deals with simultaneous input/output on several
CERN data-links. The solution involves assigning a process block to each process. The
process block, identified by the process number, contains the address of an extension stack
particular to the process, the current stack pointer within the stack and the numerical limits of
the set of globals (in the range >100) to be saved in the extension stack when the process is in a wait status. For the task itself special routines are provided to start, hold, restart or stop a process. The overall design is such that a normal task can be turned into a multi-process task with a minimum of effort.

A relatively trivial, but potentially dangerous, problem concerns the definition of the values of symbolic constants. Many of these constants are required by both assembly language routines and BCPL tasks, and even by the dump reconstruction program, so that it is important that their values for assembly language and BCPL code remain the same. The solution is to define these constants, either absolutely or via macros, in a small assembly language module. When any values change or new constants are added, the module is assembled, thus producing an output listing which contains all of the constant names and their absolute values. In addition to being printed, this output listing is input as data by a utility program (written as usual, in BCPL) which produces a file of "equate" values and a corresponding file of manifests. These files can then be called in by assembly language or BCPL programs respectively to get the values.
CHAPTER VI

COMMUNICATION SUBNETWORK SOFTWARE

One of the positive factors in the choice of Modcomp computers as switching nodes was the availability of a communications-oriented operating system called MAXCOM [5]. The official description of this showed it to consist basically of a task scheduler which activates or interrupts tasks according to associated task priorities and work queues, plus various utility system services, interrupt handlers for a variety of peripherals, and a basic operator control task.

The detailed examination of MAXCOM, which is all written in assembly language, showed it to be a well-designed system. However, there were a number of program implementation errors and a lack of strict adherence to defined standards of assembler language programming. After attempting a dialogue with Modcomp, it became clear that the best course for CERN was to abandon all hopes of the necessary changes, bug fixes etc. being done by Modcomp, and concentrate instead on adapting MAXCOM to CERN's particular requirements. In retrospect this was a wise decision, which has also been made by other users of MAXCOM.

VI.1 Basic system

A working MAXCOM system consists of MAXCOM itself (scheduler, utilities etc.), plus a varying number of tasks and device handlers. The device handlers are interrupt-driven and re-entrant so as to be able to handle multiple devices when necessary. Communication between a pair of tasks, or a task and a handler, is carried out by creating small arrays describing the objective of the communication, and then linking these arrays on the work queue of the relevant task or device. The arrays are called "queue vehicles", whilst those particular ones which represent an input/output request to a handler are called "input/output vehicles".

One of the first things to be realised was that it is most convenient to be able to dynamically create, use, and then destroy queue vehicles. However, the typical size of queue vehicle is 10 words, but the standard system possibilities of MAXCOM only allowed dynamic allocation/deallocation of buffers in multiples of 64 words. In the interests of efficiency, the buffer allocation scheme was rewritten to allow for buffers of any arbitrary length. Extra features were added to allow deallocation of part of an allocated buffer, and also to allow complete deallocation of all buffers belonging to a particular task. It has also proved useful to distinguish between buffers whose lifetime is likely to be short (not more than a few seconds at most) from those whose lifetime is long. The complete description of this dynamic buffer scheme is to be found in NPN 44.

At much the same time, it was realised that, in certain cases, a task may not wish to proceed until some particular event happens. The success of a request for a buffer could be one such event, completion of a particular input/output request another. The solution of putting the task into a loop obviously fails if the desired event requires action from a task of
lower priority, since such a task would never be permitted to execute. Therefore a scheme was implemented whereby a task could suspend itself when waiting for such events, and be restarted on completion of the specified event.

Use of MAXCOM soon revealed that the device handlers as provided by MAXCOM were not particularly well-coded or bug-free. In some cases the functional definition of the handler was inadequate, due either to CERN-requested modifications to hardware or simply for reasons of efficiency. Whatever the reason, it has turned out that all the handlers have been modified to a degree ranging from minor error correction to complete re-write. In addition, handlers have had to be defined and written from scratch for CERN data-links, and lately for a Tektronix 4027 colour display.

By far the most useful additions to MAXCOM have been those aimed at assisting in debugging software systems. Since such features might well be applicable to other systems it is worth treating these additions in detail.

i) On a Modcomp II the memory can be divided by software into alternate protected (privileged) and non-protected (non-privileged) regions. Software in non-privileged regions is not allowed write-access to protected regions, nor may it do system-oriented instructions concerned with input/output, interrupt handling or protect boundary setting. Therefore, by putting into protected region MAXCOM itself, the device handlers and the various constants and arrays which define the hardware and software configuration, the likelihood of a disastrous memory overwrite is reduced. In practice such overwrites have been extremely rare.

ii) In MAXCOM itself a number of checks were added to trap cases which should not arise. In addition, the original MAXCOM action taken upon detection of any error situation (which was an immediate halt) was changed to be the output of an explanatory message following by a halt or continuation according to the gravity of the situation.

iii) In the very early days, the only means of debugging was via the Modcomp console. As a complement to this, a console dump utility was included in MAXCOM, i.e. within a protected region of memory.

iv) As soon as it was feasible to get a dump to a large computer, using floppy disk as intermediary, a program to analyse a dump became a desirable item. To make this program possible it was necessary to include in MAXCOM, in a fixed set of locations, various critical values and pointers to tables.

v) When a system does crash it is vital to know the most recent history of events. For this purpose a trace buffer was included in MAXCOM. Any software, at any interrupt level, may add any information of interest at any time. The buffer is filled in a circular fashion, so that new information overwrites the oldest information. The information is grouped into up to 15 types (e.g. handler, task or system; IBM, CDC or CERN data-link connection), with each type having a corresponding console switch which can be used to suppress trace entries for that type when necessary. This latter suppression is very useful when looking for particular errors (e.g. in a handler), whilst the full trace concept is absolutely indispensable for general debugging.

vi) For on-line debugging of a running system, a special debug task was written. It has many useful facilities, including the ability to examine and change memory, software-implemented break points, examination of the current state of any task, print-out of all or part of the trace buffer etc. In addition, it can be used in a stand-alone mode after a system crash (except for serious memory overwrite cases) to examine core buffer ownership, search for particular word patterns etc. This task is particularly useful when at an early stage of testing of new software systems, since at this stage crashes are frequent. The task is described fully in NPN 32.

A further development which has proved essential to the Modcomp software system has
been that of making use of pseudo-devices. In MAXCOM terminology these are called "symbionts", and are simply a software procedure whereby a hypothetical hardware device (plus an equally hypothetical handler) are simulated by a software task. The task itself may make input/output calls to real devices. The most important application of symbionts, which will be explained later, is their use to make the central node control task of a Modcomp treat all links to other computers (CDC, IBM, Modcomps, user computers) in an identical way regardless of the particular hardware involved.

Although the concept of symbionts was included in MAXCOM, their intensive use soon revealed shortcomings and oversights in the implementation. Considerable software effort was needed to correct this, and, in particular, to allow one task to simultaneously handle multiple pseudo-devices, to abort input/output request or to make various utility routines normally used by handlers (timeouts, input/output queue manipulation etc.) also available to symbiont tasks. As a result of this software effort, the concept and use of symbionts has now been thoroughly integrated into MAXCOM.

VI.2 Software overview

This section is intended as a concise overview of the software of the network nodes. A complete detailed description is given in NPN 39.

All of the network nodes have a certain amount of common software which must be resident in the node at all times. Some nodes have particular resident software in order either to support particular hardware (such as a connection to a large host computer) or to allow performance of a particular function (such as entering interesting events on a log file stored on the IBM). There are also tasks which are non-resident (i.e. not in the resident software) but may be loaded into free memory and then executed on some or all of the nodes.

Even though different nodes may have different software, it is desirable to standardise as much as possible. For this, the obvious approach is to consider the hypothetical case of a node which has all possible software and hardware. In such a case, all different tasks would have different task numbers and all different devices different device numbers. The standardisation then comes by fixing these task and device numbers over the whole set of nodes. Of course, this means that, on any individual node, some task numbers and device numbers will correspond to a non-existent task or device. Thus, there has to be software to trap calls to non-existent tasks or devices, and the task and device tables are longer than strictly necessary. However, this is far outweighed by the benefits of standardisation.

Another standardisation is concerned with the treatment of different hardware connections to other computers. These various connections are normally treated at the interrupt level by assembly language handlers, each with different specifications. In order to hide these differences from the higher-level tasks it suffices to define a pseudo-device with a standard specification. Input/output or control requests to this device are treated by symbiont tasks of which there is one per device type. The symbiont tasks are coded in BCPL. In the long-term, a gain in efficiency could be obtained by combining a real handler with the corresponding symbiont task into a single, more complex, handler. Such a step would only be undertaken when specifications are completely frozen, and even then only if there is a demand for increased throughput which cannot be satisfied in any other way.

Another feature for which a symbiont has proved invaluable is the "remote console" feature. This is too complex to explain fully here, but is described in NPN 62 and in [13]. By means of it, any task can send messages to any console in the network, or even to any other task in any network node. In addition, any node can eavesdrop on the local console communications of a different node. Typical uses might be the control of a node whose local console is not working, or a dialogue with a distant node for the purposes of executing test programs. However, there are many other uses which have been devised. In practice, the
remote console symbiont works by sending individual self-contained packets, commonly called datagrams, around the subnetwork. Since there is a small, but finite, chance of losing such packets, it is essential not to use the remote console feature in cases where loss of a packet would be catastrophic.

There are numerous checks to ensure that the software (and hardware) is functioning correctly. Any error condition is classified as serious or non-serious. A non-serious error only merits a warning message on the console. However, a serious error always provokes an error message followed by a halt, where the error message is written only onto the local console by a small self-contained routine within the main MAXCOM resident (i.e. in protected memory). The action to be taken then is reloading by a local operator from a floppy disc. More sophisticated methods are under development to provide loading from the IBM either under operator control, automatically by a neighbouring node or even on request from the dying node itself.

There are some utility tasks which could in principle run in any one (or perhaps several) nodes. In some cases, tasks in all nodes may wish to communicate with these utility tasks. In order to remove the necessity of knowing the real location of a utility task the notion of a "virtual host" is used. For this, a "virtual host number", in the hexadecimal range F0 to FF, corresponds to the desired utility task. All nodes other than that or those on which the utility task is implemented use normal network routing to send information to the virtual host. When this information finally arrives at the chosen virtual host the software recognises that it contains within itself the required task. The virtual host concept is currently used for the control centre (FF), reserve control centre (FE) and message logging (F8) tasks, and for utility services available to subscribers of the network.

VI.3 Principal tasks

This section describes the function and capabilities of the principal tasks which are used in the Modcomp nodes. In this context principal is to mean the tasks which are of particular importance to the operation of the network. Note that a more detailed description is given in an Appendix B to this report.

Tasks may be used for control of pseudo-devices (symbiont tasks) or be independent of them (standard tasks). Some tasks may only be required on a subset of Modcomps (for hardware or operational reasons) whilst others might, in principle, be usable on all Modcomps. Some tasks, needed for a given Modcomp, will always be loaded in that system (resident tasks), whereas others (non-resident tasks) may be loaded only when required. Finally, some tasks may be automatically started up by the system at initial execution time, whilst others will wait for work to arrive for them. The task which is central to every Modcomp is the Network Control task (TNC), which supervises all connections to other computers, calculates and issues routing information and keeps statistics on packet transfers, errors etc. It will respond to commands or information requests coming from a variety of sources, directing its reply as requested. A full description exists in NPN 59.

For packet transfers into or out of a node, TNC always sees the various types of connections to adjacent computers as a standard network link. This is achieved by the use of symbiont tasks. According to whether the underlying hardware is a CERN data-link, a CDC channel coupler or an IBM channel coupler, the symbiont task has the name SPI/SPO, SCB or SIB. Note that for the CERN data-links, which are already full duplex, there is actually an input (SPI) and an output (SPO) symbiont task.

The TNC task is capable of dealing with all the error reports which the symbiont tasks may generate when trying to input or output packets. However, in the vast majority of cases there is no error, and also most incoming packets are to be sent out again on a different link.
This procedure is speeded up by having a simple packet switching task (TSW) which controls packet throughput when possible, but in any problem case simply passes control to TNC.

The console interpreter symbiont task (SCI) is one of several high level tasks which can create packets and send them to TSW for forwarding through the network to a different node. Its main use is to allow distribution through the network of simple messages which can be generated or interpreted via BCPL input/output statements. A full description appears in NPN 62 and additional detail in [13].

The control centre supervisory task (TCC) receives information from all nodes and presents the information on a Tektronix 4027 colour graphics screen. It also may send packets around the network to perform control and monitoring functions.

All tasks may send messages to a special logging task (LOG) whenever they have anything important to record. For this, they use the console interpreter to direct the message to the correct virtual host in which the log task is running. LOG then time-stamps each message and adds it to the end of a message file on the IBM.

Non-resident tasks may be loaded using a combination of three tasks, LDT, NRT and PUL. NRT finds core space and does address relocation, whilst PUL obtains the desired file from the IBM. LDT interprets the file-name information and controls data exchange for simple file transfer and downline loading.

Both PUL and LOG, as well as a utility task (UTY) for file transfer between floppy disk and the IBM, communicate with the IBM using a set of routines in a special shared library. These routines allow for the creation of logical links to the IBM via a simplified version of a Transport Manager (STM), which has no error recovery. A full-scale Transport Manager (TMG) does exist but is not normally run for reasons of memory space.

General operator interaction with the system is done via the standard operator control task (OCT). A specialised task (BCL) also exists for the purpose of debugging.

There are many other tasks available, most of which are non-resident. In particular, many hardware diagnostics exist, thus allowing investigation of possible hardware faults on-line. This is an important aid towards a maximum up-time for the network.
CHAPTER VII

HOST CONNECTIONS

The term 'host' is used for those CERNET computers which offer a service to other computers. This service is implemented via one or more programs which run regularly in the host computer and are willing to converse with any authorised user program in any other CERNET computer. In particular any host computer normally runs a 'File Manager' which is ready at any time to converse via a logical link with any program in any computer in order to transfer complete or partial files.

It is likely that, in the near future, File Manager programs may be written for computers other than the large CDC/IBM mainframes of the computer centre. Such programs might then run on one or more computers of each of the standard minicomputer types, so as to offer a particular service to other similar minicomputers.

For the purposes of this report, the only hosts considered are the CDC (6500 and 170/720) and IBM (370/168 and 3032). Each host has a File Manager capable of accessing both permanent files and job input/output file queues, and of informing a connected user program of the existence, or otherwise, of files of certain specified types. The IBM can offer some further services as explained in the next section.

VII.1 IBM

The IBM mainframes provide in general a wide range of services for users of the computer centre. This is also true of the IBM's role as CERNET host.

In the following subsections, there is a detailed explanation of the way in which the CERNET connection to the IBM has been implemented and made accessible to users.

VII.1.1 Basic aims of the CERNET connection

The fundamental aim of the connection of the IBM machines to CERNET was to provide program-to-program communication, i.e. to enable a program running in an IBM computer to exchange information with one or many other programs running in any of the CERNET connected computers. This aim was to be achieved with a minimum of effort and resources for the user, just by calling a set of standard subroutines. In particular, programs using the CERNET connection should not require any privileges, e.g. running in supervisor mode, or being 'authorised' in any way.

VII.1.2 Factors that determined design and implementation

The most important factor determining the design was, of course, the available hardware. The decision had been taken earlier to implement the packet switching service of CERNET on Modcomp II computers, for which the manufacturer was able to supply a Modcomp to IBM/360 channel interface, Model 1950 [14].

31
This microprogrammed device, in conjunction with software running on the Modcomp CPU, simulates an IBM control unit serving up to 255 peripheral devices.

A very tempting idea, therefore, was to simulate a couple of IBM peripherals on a Modcomp computer and to let programs on the IBM read from and write to these simulated devices whenever they wanted to communicate with CERNET. Thus the Transport Manager service for the IBM connection, which implements the end-to-end protocol, and takes care of message assembly, disassembly and flow control, would be put on the Modcomp. This, however, turned out to be impossible, as the necessary buffering of messages would almost immediately use up all available core memory on the Modcomp.

So it was decided to implement the CERNET Transport Manager as a 'started task' on the IBM computer. All communication with CERNET is channelled through this task, which therefore needs efficient communication with the front-end Modcomp and efficient information exchange with user programs running on the IBM computer. Both these requirements should, in principle, be met by operating system services. The operating system under which the CERNET connection had to be implemented, was IBM's most recent one, MVS. Though it provides for efficient communication with peripheral devices, thus making a high speed data flow between the front-end Modcomp and the CERNET Transport Manager on the IBM possible, one of its key features is to protect different programs against each other by nicely 'caging' them in their own virtual address space.

However, some of this virtual address space (CSA: Common Service Area) is common to all programs, thereby providing an area for communication. But, despite the fact that IBM surely have realised the need for these inter-address space communications, there is no service, i.e. no standard set of supervisor calls, provided by MVS for such an information exchange.

Fortunately, at the time of the design of the CERNET/IBM connection, the decision had been taken to run the STANFORD terminal handling/text editing system WYLBUR/MILTEN on our IBM computer. The adaptation of this system to run under MVS, which was done by the RAND Corporation, also required inter-address space communication, which was implemented by a RAND software package called WINDOWS. This package uses a system of control blocks and buffers in the CSA to establish unidirectional communication 'windows' between different virtual address spaces. It thus provides the necessary means for communication between the CERNET Transport Manager and user programs on the IBM.

The choice of the implementation language was mainly determined by the need to keep overheads as small as possible and the ability to use operating system services easily and freely. On the IBM computer, this inevitably restricts the choice to PL/S and Assembler. PL/S, however, is not available from IBM, so the coding was done in Assembler.

Another important principle that guided the design and the implementation was to reject all ideas or proposals requiring modifications of the operating system.

**VII.1.3 An overall view of the Transport Manager**

The following is intended to be a very simplified description of the Transport Manager implemented on the IBM machines.

The CERNET Transport Manager runs under MVS on an IBM computer as a single, event-driven task.

The events that cause the Transport Manager to take some action, are:

- completion of an outstanding READ from Modcomp.
- completion of an outstanding WRITE to Modcomp.
- a user program requesting a Transport Manager service.
- a user program signalling its termination.
- receipt of a regular timing signal every second
- the console operator requesting a Transport Manager service.

32
Upon completion of an outstanding READ, the Transport Manager checks whether the operation completed successfully or not. Successful completion means that a packet has been read from the front-end Modcomp. This packet is then treated according to its header contents i.e. according to packet type and/or data type. For details the reader is referred to NPN 16 which describes the end-to-end protocol. Unsuccessful completion of the outstanding read leads to a complete shutdown of the connection to the front-end Modcomp with a corresponding message appearing on the IBM operator's console.

Essentially the same action is taken for unsuccessful completion of a WRITE operation. In case of a successful completion, a check is done to see whether a reply is expected: if so, a timeout value will be initialized.

As already mentioned, the 'WINDOWS' package is used for communication between user programs and the Transport Manager. There are two kinds of information exchange:
- requests to and replies from the Transport Manager.
- data to and from a partner program via a logical link.

Accordingly, there are two types of windows:
- two static windows which 'belong' to the Transport Manager, TMREQUEST (the odd spelling is due to IBM's restricting names to eight characters), and TMREPLY. Both are 'multiple' windows seen from the Transport Manager, i.e. many programs can write into TMREQUEST and the Transport Manager can write back to many programs reading from TMREPLY.
- dynamic windows for data transmission which are created when a logical link is established. A parameter supplied by the user program determines how many pairs of windows (one for each transfer direction) are to be created when the link is established. This parameter determines the maximum number of outstanding transfers in either direction.

When a user program requests a Transport Manager service, it writes a request in coded form to the TMREQUEST window and waits for the reply. The Transport Manager is notified that this event has taken place. It decodes and satisfies the request, whenever possible, and then writes an appropriate response to TMREPLY. The user program then continues according to the reply returned by the Transport Manager.

Data transmission is somewhat complicated by the fact that CERNET error recovery procedures work on the message level rather than on the packet level. To allow for retransmission, entire messages have to be kept at the sending end until the acknowledgement from the remote receiver arrives. In our case, this is accomplished by holding the window buffer until the acknowledgement indicates a successful SEND operation.

A very handy feature of the WINDOWS system is that it signals a closure of a window by one communication partner to the other partner. So, whenever a user program closes its 'pane' to the Transport Manager windows TMREQUEST or TMREPLY, either voluntarily (because it no longer wants to communicate via CERNET), or involuntarily (by the WINDOW CLEANUP procedure which is automatically invoked at step termination), the Transport Manager is notified of this event and can perform the necessary link closures and cleanup functions.

A supervisory function of the Transport Manager is to ensure that certain things happen within finite time intervals, e.g. a remote Transport Manager responds in time. For this purpose the Transport Manager is activated at regular intervals of 1 second real-time. It then looks at all 'items' it finds on its 'timeout queue', decrements delay values and takes appropriate action when a delay expires.

Whenever the Transport Manager is activated for any reason, it also checks if there has been a command issued by the console operator. The console operator can request various displays of users, links and flags and can ask the Transport Manager to stop. In the latter case the Transport Manager will close all open links and ABEND all user programs using CERNET at that moment, before it terminates.
VII.1.4 Some details of the Transport Manager implementation

In the following some details of the Transport Manager implementation are given which may be helpful for people faced with similar problems.

The communication with the front-end Modcomp is achieved by simulating two non-standard peripheral devices on the Modcomp, one for input, one for output, which of course have to be included in the system generation. The only difficulty then is to enable the Modcomp to act as an 'active' device whenever it wants the IBM to read from it. The idea of using the 'attention' mechanism of terminal-like devices was quickly abandoned (being too complicated) in favour of what may be called the 'delayed device-end technique'. When the Transport Manager is started on the IBM, it immediately issues a read command to the corresponding Modcomp device. The Modcomp responds to this in its initialization phase by sending a pseudo packet, i.e. a number of bytes smaller than any possible CERNET packet would have, followed by a "channel-end", but not by a "device-end" status.

A special channel-end appendage routine recognizes the fact that the packet received is too short to be meaningful and immediately re-issues the read command without signalling a read completion to the Transport Manager. This re-issued read operation is, however, held up in the IBM by the input/output supervisor which, by inspection of the device status, recognizes that the device-end is still pending. Not until it has a real packet to send to the IBM does the Modcomp issue a device-end. The reception of a device-end will make the outstanding read operative, the packet will be read into the IBM and will again be followed only by a channel-end to liberate the channel, but not by a device-end. This time the test for sufficient size in the channel-end appendage routine is passed successfully and the Transport Manager is told that a read has completed. The next read operation issued by the Transport Manager will then be executed immediately or queued, depending on whether the Modcomp did or did not send a device-end in the meantime.

For transfers in the opposite direction, it is assumed that a buffer big enough to hold a maximum size packet is initially available in the Modcomp so that the very first WRITE can be executed immediately. Upon termination of a WRITE to Modcomp operation, the Modcomp will send channel-end and delay the device-end until a new buffer is ready to receive the next packet. In this way, the transfer operations from IBM to Modcomp are adapted to the slower of the two partners, whichever that happens to be.

Both READ and WRITE use the EXCPVR macro instead of the standard EXCP. EXCPVR is, on the one hand, more efficient than EXCP because it relies on the fact that the caller will supply channel programs which have correctly been translated from virtual to real addressing, but on the other hand it puts the burden of page fixing and channel program translation on the caller. In our special case this burden is rather light. A packet will never exceed 2046 bytes. So a buffer pool can easily be arranged with buffers of 2048 bytes each, two per virtual page of 4096 bytes, such that no buffer ever crosses a page boundary. Page fixing and channel program translation is then a trivial job, which is done in an appendage routine (PGFX and SIO appendage) of 48 bytes total size.

The appendage routines must be specified in the data control block (DCB) used for the input/output operation and are searched for in system libraries at OPEN time, if they are not yet loaded. In order to avoid linkage of these special purpose appendages into system data sets every time a new version of the operating system is generated, a trick is used: before the two DCBs related to Modcomp input/output are opened, i.e. at Transport Manager initialization, 'LOAD' supervisor calls are issued to load the appendages into the job pack area. At Transport Manager termination the corresponding DELETE calls are issued.

No attempts whatsoever are made to recover from a crash, because this is always regarded as a catastrophic failure. So far, the software has been sufficiently reliable to permit this policy. In the rare cases of crashes, a restart of the Transport Manager is needed.
An invaluable aid for debugging not only CERNET software on the IBM but also for testing and debugging remote Transport Managers has been the built-in trace and snap facility. Every significant action taken by the Transport Manager causes insertion of a 16 byte entry into a circular trace stack (4 bytes of text identifying the trace call, 8 bytes of information depending on the particular call and a 4 byte time stamp). The current contents of the trace stack and/or the current memory contents can be printed at any time by using an appropriate operator command.

**VII.1.5 The Transport Manager user interface package**

This subroutine package implements the communication between a user program and the Transport Manager. It codes user program requests and decodes and acts upon Transport Manager responses. The underlying principle of this part of CERNET software was to protect the CERNET-IBM connection from errors in user programs.

This led to a very rigorous checking of user supplied parameters. Also, certain operations must follow each other in a meaningful sequence, e.g. a wait for a transfer completion is only accepted when a transfer is really outstanding etc.

Another feature enhancing the protection of the Transport Manager is the fact that messages are assembled from packets in the user program’s buffer not by the Transport Manager but by the interface package upon reception of the ‘receive complete’ status. So an erroneous buffer address has no consequences for the Transport Manager’s functioning.

For details of the calling sequences the reader is referred to NPN 90.

**VII.1.6 An overall view of the File Manager**

The File Manager is a program running permanently (as a started task) on the IBM computer. It is a good example of a typical ‘network service’, a program that deals with requests it receives from remote user programs via CERNET.

The File Manager provides access to data sets on IBM machines according to the specification of the file access protocol (FAP) in NPN 4. A detailed description of its external specification, in particular IBM-specific information, is given in NPN 72.

It is event driven, very much like the Transport Manager, but its structure is slightly more complicated. It consists of a main task and a number of subtasks to which the main task delegates work for certain commands. There are subtasks that are statically attached to the main task, and others that are created when needed.

This multitasking structure is necessary because some of the File Manager commands result in very time consuming processes, e.g. file migration by the Hierarchical Storage Manager HSM, and it would have been prohibitive to have the File Manager waiting for the completion of such a process locking out all other File Manager users.

The events that trigger actions of the File Manager are:

- a CERNET event (see below)
- a subtask termination
- expiry of a fixed time interval
- an operator’s request for service

A CERNET event can be:

- a new logical link has been established
- a RECEIVE operation has completed
- a SEND operation has completed
- a logical link has closed

When a new logical link is established, the File Manager gets working space and builds
tables and control blocks for this new partner. It then issues a first receive to get the caller's first command.

When a RECEIVE completes, the File Manager decodes the command found in the message header and branches off to the appropriate subroutine to treat the command. Messages received without a header are always interpreted as a PUT command.

When a SEND completes, the File Manager, depending on whether the caller is doing a high speed read or not, branches to the GET subroutine and sends the next block to the caller or issues a RECEIVE for the next command from the caller.

When a link closes, the File Manager closes any open file for this customer, and finally returns this caller's working space to the system. This process has to be delayed if any subtask has been started for this customer because subtasks rely on the presence of certain control blocks while they are working.

Upon subtask termination, the File Manager checks if the logical link for which the subtask was working is still open. If so, it completes the treatment of the command and sends the result back to the caller; if not it completes the cleanup process.

When a predefined time delay has elapsed, the File Manager checks the operator request flag.

At any time, the operator may request a display of the File Manager's current activity, an immediate termination with all open links being closed at once, or a shutdown procedure where current customers are still served but new requests are refused and termination occurs after the closure of the last open link.

Like the Transport Manager, the File Manager contains a trace facility with a circular trace buffer, the current contents of which are printed upon operator's request. In addition to this, the operator may also request a dump of the File Manager's core memory contents on a printer.

As far as details of the commands obeyed by the File Manager are concerned, the reader is referred to NPN 72.

**VII.1.7 The file access user interface package**

The file access user interface package available on the IBM was written according to the proposed specifications in NPN 4. It takes into account the specific functions and features of the File Manager implementations on CDC and IBM. Thus it can also be used for local tests of transfer utilities and/or the File Manager on the IBM.

There again the design principle was to detect errors as early as possible, i.e. at the user program level, to avoid exposure of vital parts of the CERNET software to hazardous parameter combinations and to avoid unnecessary actions at the File Manager level for cases which are obviously illegal.

Thus, user-supplied parameters are not only checked for being meaningful, but consistency checks are also done on the sequence of operations etc.

Whenever any of these tests fails, the user program is passed back a status value signalling the error that has been detected. He can use one of the standard functions of the package to retrieve an explanatory message in plain English.

**VII.1.8 Miscellaneous CERNET jobs running on IBM machines**

There are two jobs to be mentioned in particular: ECHO and CHIMP.

ECHO is a started task running permanently on every IBM machine with a CERNET connection. Its name reflects its function. Any remote machine may establish any number of logical links to this process, send messages and receive the same messages back. ECHO is not only used to test hardware and software up to the level of the end-to-end protocol but it is also used by monitoring tasks in minicomputers probing the availability of CERNET
connections at regular intervals.

CHIMP, the CERNET High speed Inter-Mainframe Package, is a started task running permanently on only one of the two IBM machines sharing the spool data sets and queues. Its purpose is to transfer job submit files and print files between CDC and IBM in the computer centre.

VII.2 CDC

The central CDC computers offer a CERNET file transfer service through a File Manager running in the CDC Cyber 170/720. There is no direct connection between CERNET and the CDC 7600, and the connection to the CDC 6500 is used only for back-up and testing. Files travel between the CDC Cyber 170/720 and the other CDC computers using the manufacturer's hardware and software.

The CERNET service is provided by a self-contained network interface program called NETWORK. NETWORK includes a File Manager, a Transport Manager and some diagnostic aids. NETWORK is the only program in the CDC computers to transfer data through CERNET; there is not yet the possibility of program to program communication service for other programs.

VII.2.1 Hardware connection

The CDC 170/720 and 6500 are connected to the Modcomp computers of the network through CDC 6681 data channel converters and Modcomp 5943 satellite couplers [15].

The 5943 is a special version of the Modcomp 1941 standard satellite coupler. Like the 1941 it provides a common status register for synchronization, and allows data to be transferred in either direction with repacking to compensate for the 60-bit words and 12-bit channels of the CDC computers. It differs in that it allows extra modes of packing/unpacking between 16-bit Modcomp words and 60-bit CDC words.

VII.2.2 Software components

The principal components of CERNET software in the CDC computers are:

i) The network interface program NETWORK and its associated peripheral processor programs UMC and 1UN. UMC drives one or more 5943 satellite couplers. 1UN starts NETWORK, in response to an operator command.

ii) The satellite coupler diagnostic test program MCDIAG and its associated peripheral processor program UMD. MCDIAG, UMD and the corresponding Modcomp task DCB constitute an on-line test for the 5943. There is also an off-line test program UMS, which runs in a peripheral processor under Control Data's system maintenance monitor SMM.

iii) Some utility programs. RDNET and WRNET translate blocked files between CERNET format and CDC format. NETCON controls the use of data sample file space, which is used to hold long files of short lifetime. NETPPD prints the peripheral memory dumps created by UMC. NETSEF prints UMC's system engineering file messages, which record satellite coupler input/output errors.

NETWORK and MCDIAG are written in SYMPL (Control Data's system implementation language) and COMPASS (assembly language). The utility programs are written in COMPASS, MORTRAN (an extension of FORTRAN), and PASCAL. The peripheral processor programs are written in COMPASS.

The programs run in the CDC Cyber 170/720 and the CDC 6500, under the NOS/BE 1.3 operating system. There are, in addition, versions of RDNET and WRNET for the 7600, running under 7000 SCOPE 2.1.4.
VII.2.3 The network interface program

The components of the network interface program NETWORK are:

A command interpreter.
A display formatter.
The Transport Manager.
A bounce process.
An echo process.
The File Manager process.
A File Manager test process.

The command interpreter and display formatter provide the interface with the operator of the CDC Cyber 170/720. Some commands are obeyed by NETWORK directly, others are passed to the satellite coupler driver UMC. The console display shows coupler and process traffic, and the most recent log messages. The operator may ask to see a list of the currently valid operator commands, or a list of network subscriber computers, in place of the traffic display.

The Transport Manager provides the CERNET data transport service for the four processes. A process may request a logical link to another process, accept a logical link requested by another process, send and receive data through a logical link, and close a logical link. Data are sent as messages consisting of a message header and a message body, where the message header is short and the message body may be long (spreading over several packets). Any of the four modes of data repacking provided by the 5943 satellite coupler may be applied to the message body; the message header is always treated as 8-bit data.

The bounce and echo processes are diagnostic aids. Bounce generates messages of random data and length, sends them to an echo process elsewhere in the network, receives the echoed messages, and checks that they are correct. Echo is the counterpart of bounce, and returns any message it receives.

The File Manager process provides the file transport service. It allows user programs in other subscriber computers in the network to read, write and delete permanent files, including data sample files, and to send and receive job and print files. In all cases the initiative for a file transfer comes from elsewhere, typically from one of the central IBM computers. CERNET cannot be used to transfer a file directly between two CDC computers.

The File Manager test process is a very simple test of the CDC File Manager, using a fixed repertoire of test scripts to exercise the most common File Manager operations. More serious testing depends on File Manager test programs running in other (non CDC) computers.

NETWORK contains a number of diagnostic tools. The bounce and echo processes have already been mentioned; they are used as a confidence test for CERNET hardware, and to check the availability of computers on the network.

NETWORK's log file contains a record of satellite coupler, logical link, and file open and close operations, and errors, together with traffic and error counters. It may also contain packet and/or message traces, which the operator can select for a single logical link, for all links, or (in the case of the message trace) for all links to a given process. The packet trace consists of one line per packet input or output, showing the contents of the packet header. The message trace consists of one or more lines per message input or output, showing the message header on the first line and, optionally, the beginning of the message body on subsequent lines. Both traces carry time stamps and the addresses of the packets or messages shown.

The operator can also ask NETWORK to print the contents of parts of its memory (central memory or extended core storage), to start and stop program counter sampling (used to determine which parts of NETWORK use the central processor most), and to introduce
artificial errors into its packet output stream (simulating the effect of packets being duplicated or lost by CERNET).

For software testing, NETWORK can be run by itself without UMC or CERNET. In this mode packets are passed directly from the packet output queue to the packet input queue.

Satellite coupler input/output errors are reported by the coupler driver program UMC, in the system dayfile and (more fully) in the system engineering file, and in the form of error counters to NETWORK. There are utility programs (HI and NETSEF) for retrieving the error messages from the dayfile and the engineering file. UMC recovers from transient input/output errors without operator intervention.

VII.2.4 Satellite coupler testing

The 5943 satellite couplers can be tested during CERNET operation, using the bounce and echo processes described above. For fault diagnosis, more precise tools are needed. The main diagnostic tool is MCDIAG, together with its associated peripheral processor program UMD and the Modcomp task DCB. When MCDIAG is run, the coupler under test is taken out of service, but not the associated CDC and Modcomp computers.

MCDIAG presents the operator with a repertoire of tests of gradually increasing complexity, together with some pseudo-tests whose function is to return additional information (memory and trace dumps) after a test has failed. The test repertoire starts very simply, testing flags, status and interrupts, first from the CDC side alone and then from both sides; then come data transfer tests, first in one direction at a time, then in both directions with comparison of the data returned against the data sent.

For extreme faults, there are two off-line diagnostic tests: UMS, which runs under CDC’s system maintenance monitor and tests functions and status from the CDC side only, and a similar program for the Modcomp side only. These two programs are very rarely used.

VII.2.5 Further information

The operation of the network interface program NETWORK is described in NPN 54. The external specification of the File Manager is contained in NPN 28, supplemented by NPN 33.

The diagnostic test program MCDIAG and its corresponding Modcomp task DCB are described in NPNs 85 and 86. The skeleton test program UMS is described in NPN 87.

The file conversion programs RDNET and WRNET are described in NPN 61. The data sample file control program NETCON is described in NPN 73.

VII.3 File Manager operations

The CDC and IBM File Managers allow users to read, write and delete permanent files, to submit files to job and output queues, and to retrieve job output. All files are organized sequentially; on IBM they may be members of partitioned data sets. The IBM File Manager provides additional facilities: for examining the file catalogue, the directory of a partitioned data set, the job and output queues; for job control; and for moving data sets to and from mass storage.

Permanent files may be of three types: standard permanent files, data sample files, and archive files. Data sample files are large files of short lifetime, such as samples of experimental data due for immediate analysis. Archive files are used to hold back-up copies of minicomputer files on IBM mass storage.

Several different data formats are allowed, and appropriate translations between host and network formats are provided, usually by the File Managers themselves. On CDC, some translations are performed by utility programs.

The principal data formats are: text, blocked binary, and unblocked binary. Lines of text are represented within CERNET as records of ASCII characters, with strings of consecutive
spaces replaced by escape sequences; the records are blocked together to form CERNET messages. Short binary records may be sent through CERNET in blocked binary format, with several records per CERNET message. Longer binary records travel through CERNET unblocked, each record occupying one CERNET message.

On CDC, text is represented in display code with zero byte line terminators (record type Z) or in ASCII; blocked binary data is held as variable length blocked records (record type W); and unblocked binary data is held as operating system logical records (record type S). The File Manager reads and writes display code text and system logical records directly. Utility programs translate to and from the other formats.

On IBM, text is held as fixed length blocked records of EBCDIC characters (record format FB) or in WYLBUR compressed (EDIT) format; blocked binary data is held as variable length blocked records (record format VBS); and unblocked binary data is held as undefined records (record format U). The File Manager reads and writes these formats directly.

The additional operations provided by the IBM File Manager fall into four groups: catalogue and directory information, date and time, HSM services, and job control. A user can read an extract from the file catalogue (describing a particular data set or containing an index to other data sets), or the list of members of a partitioned data set. The date and time command returns the date and time of day as a character string. The HSM services command provides access to the Hierarchical Storage Manager (HSM) which looks after files on mass storage. The following HSM operations are implemented: migrate (a disk file to mass storage), recall (a file from mass storage to disk), delete (a migrated file), backup (copy a disk file to mass storage), recover (a disk file from a backup copy), and recatalogue (a migrated file which has been deleted from the file catalogue). The job control command allows a user to locate a job or jobs, to release job output for printing, to delete job output, to cancel an executing job, to read the names of the output data sets of a job, and to read the output queue for a specific destination.

File Manager operations are described fully in NPNs 28 and 33 (for CDC) and in NPN 72 (for IBM). The file access protocol is defined in NPN 4.

VII.4 Real-time programs (IBM)

One of the original aims of CERNET was to permit program to program communication in real-time, with the specific object of allowing a minicomputer to offload data for CPU-intensive processing onto a more powerful computer. The IBM computers at the computer centre were the first to be used as the powerful computer in such a dialogue, although with the passage of time it is likely that others may come to provide such a service.

The user view of such real-time program to program communication is dealt with in Chapter VIII. For the moment what is considered is the way in which the requirements for real-time programming make the execution of such a program in the IBM different from that of standard batch-style or other programs.

Given the existence of the Transport Manager User Interface Package described earlier in this chapter, it is relatively simple to write a program which executes in the IBM and converses with a partner program in some other computer. It is equally simple to modify an existing data analysis program to take input from a partner program, rather than from disk or tape, and to return results to the partner. However, in the operational environment, running such programs in real-time poses problems.

When the user program executes on the IBM, it will be competing for resources with other programs, including the IBM Transport Manager itself, Wybur, the IBM system and others. The resources which it requires are, in general, simply CPU time in a short concentrated amount, since its normal function is to process the incoming data message from the partner program and to return an answer to it.

40
The solution adopted is the logical one of defining a particular job class for these real-time jobs. The characteristics for this job class are then optimised for such short intensive data analysis. For the case of IBM, there is a comprehensive set of parameters available for job class definition at system generation time.

Another potential problem is the number of such programs which may execute in parallel. If there were to be too many, then the total demand might have a considerable effect on the other services. Fortunately, so far, there has been no widespread demand for this facility, so that it has sufficed to have at most two initiators for this class of jobs on the IBM. If more were to be required, then one possibility would be to move services (i.e. initiators) between the two IBM mainframes. Currently most services (Wylbur, CERNET File Manager, real-time job initiators etc.) are run on one particular IBM (the 3032).

In summary, therefore, real-time programs are run regularly on the IBM but the load of such jobs has been very light so far. However, there are potential applications which may give rise to a growth in the near future.
CHAPTER VIII

USER CONNECTIONS

This chapter gives some details of the way in which the user computers are connected to CERNET and the software programs and packages which enable a user program to communicate with a program in a different computer. The way in which the various user computers have come to use CERNET is described in Chapter XII.

VIII.1 The portable Transport Manager

Given the variety of different user computers, it is an obvious advantage if as much as possible of the basic software required can be easily transported from one user computer to another. This is particularly true for the implementation of the end-to-end protocol software, which must always exist and obey the rules laid down by the protocol.

The concept of what is known as the "portable Transport Manager" (portable TM) is relatively simple. It is assumed that any reasonable minicomputer operating system can provide a multi-tasking environment, and that some reasonable form of inter-task communication is possible. The portable TM is then a task to which other tasks can make requests and send or receive data messages.

The simplest way to envisage the portable TM is as a central kernel surrounded by an outer layer. The kernel is machine-independent and implements the protocol as laid down in NPN 16. The outer layer is made up of routines which are system-dependent.

The outcome of the above concept and philosophy is the portable TM and the associated User Interface Package. This latter was agreed between all concerned and described in NPN 40, so that all Transport Managers could adopt it as a standard. The portable TM was then defined with a clear separation of the kernel and the system-dependent routines. The definition of system-dependence was anything which could arguably be so on any of the variety of minicomputer operating systems.

What has since resulted is therefore a portable TM for which the kernel is completely system-independent to the extent that it only depends on the word length being at least 16 bits. The handling of the data within packets is completely in the domain of the system-dependent routines. The routing and protocol headers are inserted or extracted via other system-dependent routines before the kernel can handle them. The kernel calls a system-dependent routine on any occasion at which some action might be required on some minicomputer. The fact that some of these calls may be unnecessary on a particular mini is unimportant, since it is easily handled by making the associated system-dependent routine a dummy.

The portable TM is described in NPN 48. A brief description of its kernel and system-dependent routines follows.
VIII.1.1 The portable TM kernel

The kernel is that part of the TM which implements the end-to-end protocol. It is normally waiting to be activated for one of several reasons, namely a new request from any other task, the input or output of a packet or a timeout event.

A request from any other task enters the kernel via a system-dependent routine which ensures that it is in a standard format. This format is simply an array of values defining the request and is called a Request Block. The standard action is to verify that this Request Block contains a meaningful request and then to initiate the required action as soon as possible. Information about the action and its success or failure are put into a status word inside the Request Block and sent back to the task via other system-dependent routines.

For packet input, the kernel is normally willing to receive a single packet at a time. It must then deal with it, including the transfer of data or information to the correct user task, before permitting input of another packet. In practice, input packets are always read into a memory buffer (fixed location or dynamic) and any message data given to the user task by a memory copy. Packet output is simply the reverse.

The kernel keeps track of all of the user tasks known to it, by means of arrays known as User Control Blocks (UCBs). These contain information given by the user task when it makes itself known to the TM for the first time. They are chained together via their first word in order to make it easy to check on all the known user tasks.

Similarly, the kernel keeps track of all of the logical links known to it, by means of arrays known as Link Control Blocks (LCBs). These keep track of the current state of each logical link, the list of input/output requests for the link, message and fragment numbers, name of remote machine etc. Information about the links of any particular user task are also chained together.

The whole of the kernel is written in BCPL, in order to be completely portable. At its highest level, it is simply a loop in which system-dependent routines are used to wait for an event (packet input/output, user task request or timeout expiration), deal with the event and initiate any new packet input/output. There is provision to call a special system-dependent routine if the kernel has no active logical links, thus allowing for roll-out onto disk for the TM.

VIII.1.2 System-dependent routines

The system-dependent routines are of necessity callable from the BCPL kernel. However, many of them are frequently coded in a language nearer to assembly language, in order to make use of privileged features of the local operating system. There are three categories of these routines.

The first category consists of the routines which deal with the interaction with the user task. This includes reception of requests from the task, which are then correctly formatted into request blocks, and the return of responses to the task. There is also the copying of information to and from the user task's data areas. This can be simple or complicated according as to whether there are or are not system routines able to copy to and from the user task's memory space without the explicit co-operation of the user task.

The implementation of this first category also involves preventing, permitting or even forcing of the user task into a suspended state, and restarting a suspended task when necessary. This is obviously very system-dependent, and very frequently depends upon whether the copying to and from the user task's data areas is possible without user task collaboration. In practice any suspension might be done either directly by the user task itself (upon request of the TM) or by the system-dependent routines of the TM (using special privileged system routines).

The second category of system-dependent routines contains those concerned with packet input/output. These routines have to report the success or failure of packet transfer operations,
clean up when the kernel indicates that it has finished treating packets, and set up new input requests when necessary. The routines are obviously different for different hardware configurations, in particular for PDP-11 computers which may be connected directly via CAMAC interfaces or indirectly via the OMNET gateway.

The third category consists of utility routines. The request to set a timer belongs to this category, as does the request to wait for any new event. The kernel can indicate that it currently has no outstanding activities to a routine which may be able to cause suspension or rollout of the TM. There are initial setting-up routines which may allocate space for packets, link or user control blocks, debug trace buffers etc. and which may hand back identification information. Finally, event tracing and dumping should be partially in the domain of this category.

VIII.2 PDP-11 connections

Direct connection to CERNET is available for PDP-11 computers running under the RSX-11M operating system and equipped with a CERNET CAMAC link.

The aim was to optimize the CERNET software as far as space and time were concerned, in machines having at most 256 Kbytes of memory and very often devoted to crucial real-time data-acquisition tasks.

The following considerations were taken into account. The standard way to operate on CAMAC modules (ESONE CAMAC calls [10]), though satisfactory in most applications, cannot be used in this case, because it issues one system directive (QIO in the RSX jargon) for each CAMAC operation, introducing an overhead of approximately 2 milliseconds. Thus it seemed preferable to drive the CERNET CAMAC modules through software embedded in the executive (and called "driver" in RSX terminology). In such a way, with a single system directive it was possible to perform complex operations on the CAMAC modules: the driver included the full line protocol.

The use of the portable Transport Manager was at this point possible, but it carried with it several drawbacks. One was its size, quite large for such small machines. Another was the double copying of messages: from the hardware to an internal buffer (more space again), and from the buffer to the user space. Furthermore, the portable TM being implemented as a program, big system overheads were encountered when scheduling it, and in the communication between the user program and the portable TM itself.

It was then decided to try to include the full end-to-end protocol implementation into an RSX driver, much as an exercise to produce a convenient and reliable way of interlinking small computers with a direct point-to-point connection. The result was successful and the driver (called NW) has been adopted for all the PDPs-11 connected to CERNET via CAMAC links.

The NW driver is compatible at the level of QIO with the OMNET implementation: utility and user programs can run on both types of CERNET links, with no change.

Though the NW driver operates directly on the CERNET CAMAC modules, it relies on the CAMAC driver (CA) for more complex operations such as LAM handling or DMA transfers. The communication between the two drivers goes via system routines and is very fast.

Processing of the directives by the NW driver is rather un-conventional, in the sense that it does not make use of the system queues for the requests blocks, but it keeps its own data base.

A utility program NWR has been provided, which allows mounting and dismounting the link (i.e. enabling and disabling the link activity), and printing the link status.

The CAMAC modules can be put anywhere in the CAMAC system and their position can be changed at any time, the configuration being given to the NW driver dynamically at the time of mounting.
VIII.3 VAX-11 connections

The implementation of the end-to-end protocol on the VAX-11 looks very much like the one on the PDP-11. In the case of the VAX-11, arguments about time and space savings are no longer vital; still, the justification of having a CERNET driver instead of the portable Transport Manager arises from the fact that the BCPL system for the VAX-11 was not yet fully tested at the time the first connection had to be made, and that the experience gained with the PDP-11 gave a push in this direction.

There is a single CAMAC driver on the VAX-11, handling both the genuine CAMAC operations and the CERNET ones. The different functions are kept well separated in different modules, but still the communication between modules belonging to the same driver is easier than in the PDP-11, where two drivers must co-operate.

The VAX-11 driver has a number of new features. It can drive many physical links; only one of them may be connected to CERNET, the others being point-to-point direct connections to other computers. Some of the links may be used for program-to-program communication within the host VAX-11 itself: the driver makes a memory-to-memory copy without acting upon the hardware. All this structure is transparent to the user: the non-CERNET links are assigned a name, and can be addressed by using their name in place of the remote machine name. All links use the full end-to-end protocol, as the true CERNET ones.

A utility program has been provided, with enhanced capabilities as compared to the corresponding program on the PDP-11 (NWR). It is fully conversational, menu driven, and can be used to configure and mount the link system, or conversely dismount it. It can display the status of each link. The program also has built in an extended testing capability: it can bounce messages and check them back, or reflect messages to any partner (including itself). All the transmission parameters can be easily set as desired.

VIII.4 Nord-10 connections

NORD Computers operating under SINTRAN III are directly connected to CERNET using the standard CAMAC modules either via the CC-NORD10 single crate interface or the FISCHER system crate interface.

Presently, about ten Nord computers are linked to CERNET and use the standard CERNET software. Apart from the software support and software development computers, these systems are generally used by experimental groups for data acquisition and control. In addition, a number of special projects using NORD computers are linked to CERNET (Graphics Station, Text Processing, etc).

VIII.4.1 The portable Transport Manager (TM)

The portable Transport Manager is a standard real-time program (task) running under the NORD operating system — SINTRAN III. The system-independent and the system-dependent part of the TM are written in BCPL. Standard Nord system routines and library routines are called via a library of special BCPL interfacing routines.

VIII.4.2 The Transport Manager User Interface Package

The User Interface to the Transport Manager is a library of routines written in Fortran and callable from any of the standard languages supported on the NORD computers (viz. Fortran, Pascal, Basic, NPL). These routines transmit user requests to (and receive replies from) the TM using the Sintran system internal message devices. As this mechanism is rather slow, user messages (data) are copied directly to and from the user virtual space (making use of the alternate page index table addressing) by the TM. The data is copied into a 1K word buffer fixed in the TM and then transferred to the CERNET Camac module by DMA (and similarly for input).
VIII.4.3 The File Access User Interface Package

The user interface to the remote file system is also implemented as a set of Fortran callable routines, which are written in Fortran and call the above-mentioned standard interface routines to the TM. In this way, it has been possible to use the same file access library on the NORD, HP, PDP-11 and VAX-11 computers.

VIII.4.4 File Access and Job Submission

A general interactive program (CERNET-File-Tran) makes available on the NORD computer the facilities offered by the remote File Managers (IBM, CDC). For example, the options common to the IBM and CDC are

SEND-TEXT SEND-DATA JOB-SUBMIT PRINT-FILE
READ-TEXT READ-DATA JOB-FETCH PURGE-FILE

and for IBM only

SEND-WYLBUR-TEXT JOB-CONTROL LIST-FILES
READ-WYLBUR-TEXT HSM-SERVICES LIST-DIRECTORY.

When called, the program first presents the network status, the availability of remote hosts and details of local open links.

User communication with this program is in the style of other Sintran subsystems:

- available options for the chosen host are shown by typing 'HELP'
- commands may be abbreviated
- default values are presented in square brackets and selected by carriage returns

In general, a user must enter his account parameters once for a remote host. These parameters may then be saved on his local NORD file base and used as defaults. (Otherwise the defaults of NORD user System are used.)

VIII.5 HP21mx connections

In CERN’s minicomputer support policy, the HPs are used as small single user systems. The 60 Kilobytes of memory available to the user is not enough to run the full CERNET software, i.e. Transport Manager and user interface packages. Only the HP development computer (2 megabytes of memory) is connected to CERNET and able to use the full range of CERNET-provided services. Other small HPs will be connected simply in order to make use of a new range of services for simple file transfer (see subsection IX.2.2) for which the full Transport Manager is not required.

VIII.5.1 Applications

CERNET utilities provide a number of facilities to integrate the development system with CERN’s central computers. These include file transfer and job submission, output spooling to computer central printers, save and restore of user disc space, data base report generations.

VIII.5.2 The Transport Manager and User Interface Package

The Portable Transport Manager used on the HP computers is written in BCPL, requires 38 Kilobytes of memory and runs under the RTEIVB operating system. TMG is cross-compiled on the IBM and then transferred to the HP development system. In order to save space, a non-standard condensed BCPL library has been included in TMG.

The interface between the user’s requests and TMG is implemented as a set of FORTRAN routines NTLIB. Buffers and requests are communicated between TMG and the user through EMA (extended memory access). This mechanism is very slow, limits the message size to 1 KW and allows only the transmission of records. The transmission of unblocked records of 6 KW used for disk backup on the IBM file base requires a special interface implemented by a program called TRKMG.
The interface between TMG and the CAMAC is done in the system-dependent part of TMG which uses the standard CAMAC calls. A real-time program CAMG handles the CAMAC interrupts.

Finally, TMG (a real-time program in the time list queue) simulates the TMG timeouts.

**VIII.5.3 The File Access User Interface Package**

The file access user interface package is a set of FORTRAN routines implementing the file access protocol. Since it uses the HP EMA buffers, FUJIP is a portable module. Furthermore, the message buffers are referred to not by addresses but by indexes in the EMA.

**VIII.5.4 General background program HPMG**

A 38 Kilobyte general background program HPMG allows up to 3 users to access the IBM and CDC facilities. Each user gets a copy of an interactive program called UMG which communicates through HP CLASS input/output requests with HPMG.

**VIII.5.5 The interactive program UMG**

The syntax of the commands implementing the CERNET operations is similar to the WYLBUR syntax: USE (get a file), SAVE (save a file), FETCH (fetch a job output), LOCATE (locate job), SHOW (show ddnames, time, directory, ...), MIGRATE, BACKUP, RECALL, RECOVER (IBM MSS functions).

The commands may be abbreviated to the first letters making them unambiguous. The options, if not specified in the command are prompted for by UMG. The characters `#` and `?` may be used to specify default values (if any) or to ask for a short explanation of the possible options which can be specified in the command. The command STATUS gives status the network host machines and the list of open logical links.

The UMG commands can also be put in COMMAND FILES. A set of command files for the most commonly used services is available.

**VIII.6 Connection of OMNET**

The network of PDP-11s referred to as OMNET [2] evolved from the tree-structured network surrounding the CII 10070. Initially, all program support was supplied by the CII 10070, and all transfers were either to or from this machine.

As the sophistication of the connected PDP-11s increased, so did the complexity of OMNET, which developed into a tree-structure. In spite of this, the basic characteristics of the network were unchanged: a network with fixed routes, transferring datagrams of virtually unlimited size.

Since OMNET represented a population of users already conversant with network procedures, and since, with the retirement of the CII 10070, these users were in search of a host, it was decided that high priority should be given to connecting OMNET to CERNET. The user has the same facilities as the directly connected PDP-11 user described in section VIII.2, though the implementation of the end-to-end protocol software is very different.

At the time, the CAMAC link was not available, and the operating system used within OMNET ('SMO') does not have CAMAC drivers. The connection to CERNET was therefore made by means of a special interface that connects directly to the PDP-11 Unibus. Its characteristics are essentially identical to those of the link hardware on the Modcomp. A special driver for this interface was written under SMO to send and receive packets under the CERNET line-protocol. It then remained to provide a mechanism for communicating across the boundary between the two networks.

It was decided to do this as transparently as possible, by considering each network as a single machine from the point of view of the other network and mapping OMNET datagrams
onto CERNET packets:

*One machine* in OMNET was equipped with a CERNET link, and referred to as 'The Gateway'. This gateway machine represents the path between OMNET and CERNET, and provides multiplexing of all transfers between CERNET and OMNET. This multiplexing is provided by means of the addressing scheme used in CERNET: A source or destination address is 16 bits long. The top 8 bits represent a 'machine', and the lower 8 bits are considered to represent an entity within the machine and are not used for routing by the communication subnetwork. For gateway transfers, therefore, the top byte of the address contains the CERNET address of the gateway, and the lower byte holds the OMNET number (limited to 8 bits) of the machine addressed within OMNET.

A *packet* arriving from CERNET into OMNET has appended to it an OMNET header completed by insertion of the destination machine number from the CERNET destination word (lower byte), and the whole is then forwarded through OMNET. Similarly, a packet destined from OMNET to CERNET carries the full CERNET packet in its data part until it arrives at the OMNET machine serving as gateway. This machine then strips off the OMNET header and forwards the remainder down the link to CERNET.

In order to provide back-up and to spread the load on the gateway, this scheme was extended to allow two OMNET machines to be used as gateways. This means that from the point of view of CERNET, OMNET has become two independent machines, with different addresses. Within OMNET, the routing towards CERNET is provided for each machine by one of two gateways. Systems using modified routing tables exist for each gateway to allow through-routing to the other, in case of trouble with one gateway connection. A case study of this gateway is given in [16].
CHAPTER IX

NETWORK SERVICES AND FACILITIES

It is easy to suppose that all that is required of a communication network is the ability to allow communication between any two subscriber computers. However, in practice, there are a number of necessary additions which make a network easy to operate and to monitor. Furthermore, there are some services which have been incorporated into the CERNET nodes in order to provide either extra facilities or simplifications for those subscriber computers which have limited resources (hardware or software).

Many of the above-mentioned services are implemented by software in one, several or all of the CERNET nodes. If not in all, then they are always addressed as virtual hosts, for which the real node(s) concerned may be changed from one to another without affecting the manner in which the service is used.

The rest of this chapter goes into more detail on the various services and facilities.

IX.1 The probe packet

The communication subnetwork keeps a periodic check on the availability of the subscribers. The basic mechanism for this check is by sending a 'probe' packet and waiting for the corresponding 'probe reply' packet from the subscriber. If no valid answer is received within 10 seconds, the user is deemed unavailable, all link output purged, and the routing algorithm informed of unreachability. A probe packet is then issued every 10 seconds until a valid reply is received, at which time the link is once again usable for network communication.

Since the probe packet is sent at regular intervals, it is also used as a carrier of information between network and subscriber: information concerning the availability of the main network hosts is carried in the packet, coded according to the setting of bits in a word of the packet.

IX.2 Services provided to user computers

IX.2.1 Name service

The first network service to be provided arose from the fact that every subscriber computer has a number associated with it. This number is required to be in the source field of every packet emanating from the subscriber computer. However, the requirement of having identical standard software in subscriber computers of the same type makes it desirable for the subscriber computer not to need to know its own number. Therefore, the CERNET nodes automatically fill in an empty source (and also empty destination) field of every incoming packet with the source number. This is possible since every node knows the number of all its directly connected subscriber computers.

A consequence of the above philosophy on the assigned numbers of the various
subscribers and hosts is the problem of initial link establishment. Originally, any subscriber computer wishing to open a link had to know the number of the destination subscriber computer Transport Manager. However, this meant keeping a table of numbers in all subscribers. A recent improvement involves keeping this table in a file on the IBM and including in a CERNET virtual host (see section VI.2) a task to read this file. The task, called the Name Server, is then available to all subscribers. All that the subscriber has to do is to address a request link packet to the Name Server and include in it the name of the required subscriber/host. The Name Server then converts the name to the correct number and forwards the packet. The requester can then find the correct number from the returning request-accept. In case of an illegal name, the Name Server converts the request-link packet into a close-link packet (with close-reason “unknown destination”) and returns it to the sender.

**IX.2.2 Simple File Transfer**

The most recent service to be included is primarily aimed at making it possible to connect a user to CERNET and carry out file transfer without having a full Transport Manager in the subscriber machine. This service was originally intended as a bootstrap loading/dumping service, but was extended to allow generalised file transfer. The basic concept is that the CERNET node to which the subscriber is connected will take care of the protocols necessary for file transfer to or from the designated host computer File Manager. Transfer between this node and the simple subscriber is then just a matter of sending blocks (not packets) via the line protocol. The service is restricted in the sense that the blocks (records) of the file cannot exceed about 2000 bytes, and that the line between the node and the subscriber is unavailable for anything other than the one simple file transfer whilst it is in progress. However, the gain is that CERNET can be used with a very simple program in the subscriber, and bootstrap loading can be simple and fast.
CHAPTER X
OPERATIONAL AIDS

X.1 The Control Centre

The increasing complexity of the network and the physical distribution of nodes made overall control progressively harder. For this reason, a service was designed to centralise information about network status and activity, so as to offer a global display function including an alarm facility for critical events.

The human interface to this service is provided by a colour graphics display which presents overall monitoring information and acts as the interactive terminal for centralized control.

The basic components whose status is managed by the monitoring service are computers (nodes and subscribers) and links. The status of nodes and subscribers is implied by the status of the links to them and the accessibility as held in the local routing tables.

Nodes and links can be in one of three states:

Operable : The node is accessible.
            The link is Open or Opening.
Inoperable : The node is known to have failed.
            The link is Closed.
Unknown : The node is inaccessible for an unknown reason.
            No information about the link is available.

These states are represented by distinct colours on the display:
• Green for operable (green+red for "Opening")
• Red for inoperable
• Blue for unknown

X.1.1 Display facilities

There are three different formats in which the accumulated information can be presented. In the "main image", a picture of the general network topology is presented. Nodes are represented as discs, links as lines either between nodes or towards subscribers represented by their machine numbers (figure 5). Critical subscribers, for which the central operations group is responsible, such as important users or hosts are shown against a background which indicates accessibility using the colour codes given above.

In the sample main image the circles represent the node computers with their hexadecimal identifiers. Each two-character identifier around the edge represents one subscriber connection. The F-identifiers at the centre represent virtual machines with control tasks and do not correspond to real computers linked in the configuration.

The location of the nodes is as follows:
In the Computer Centre (all Modcomp Classics) 11 and 12 for user connections, 13 and 14 for backup CDC and IBM connections, 17 and 18 for main CDC and IBM connections. In other areas (all Modcomp II/45) 15 and 16 the North experimental area, 1B and 1C the West experimental area, 1D and 1E the EP division area, 19 and 1A for DD division hardware and software development.

The IBM 370/168 and 3032 hosts are subscribers 52 and 51 respectively. The CDC Cyber 720 and 730 hosts are 41 and 42 respectively.

Since the full network picture can be confusing, function keys allow selection for display of a subset of subscribers. Where more detail is required about a given node, the “secondary image” can be chosen. There is one secondary image per node and it gives more detailed information as to the status of all links to that node, as well throughput statistics (figure 6).

A third display possibility, the “history image” provides a recent history of all critical elements of the network. It presents a bar-chart, colour coded as already defined. Each column represents a chronological record of events on a critical element (node or subscriber), and one row is issued for each change of state of any of these elements. The time at which each event occurred is also recorded on the chart.
**X.1.2 Monitoring facilities**

Once a given image has been chosen, it will remain on the screen. When some change occurs to the state of the network, the current picture will be updated if such a change affects it. Whether or not this is the case, a global count of unacknowledged changes will be incremented and is continuously shown at the top of the screen. This is displayed in the 'change colour', purple. A change to the status of a displayed link is indicated in the main image, by appending the character C in the change colour to the link; in the secondary image, the corresponding link information is given in the change colour. A function key is provided for the operator to acknowledge changes. This causes the change mark to disappear, the link information to be rewritten in the currently appropriate colour and the change count to be decremented.

Changes from Open to Opening state of critical elements — i.e. internode links or links to critical subscribers — are declared as "warnings". For each warning, the corresponding link has a W appended to it. A global warning count is updated and displayed at the top of the screen. The W and the warning count both flash. Simultaneously, a warning light on the central operator's console is triggered to flash. The warnings become changes when the corresponding links reopen, or can be cancelled individually by the operator. When no outstanding warnings remain, the operator's light stops flashing.
X.1.3 Control facilities

Since the basic design of the network is to distribute as far as possible all the functions, it is only through the control centre that an overall view is possible. For this reason, all commands are usually issued at the control centre display by means of the remote console facility (see section VI.2). The ensuing status changes can then be seen immediately reflected in the display of component states.

One additional operational procedure is required. This is the interchanging of facilities between two hosts. Conceptually, this is the same as interchanging the identification numbers of the two machines. A command (SWAP N1 N2) can be given to the control centre, so that the relevant change of name commands can be given to all nodes concerned. In addition, the information about such topology changes is updated within the control centre tables, so that, if a node is reloaded, the change of name command can be re-issued automatically.

X.1.4 Monitoring methods

In order to provide as far as possible up-to-the-minute information on link states, the node control task sends information to the control centre to inform it of each change of link status. Since the communication subnet offers a datagram service, there is no assurance that this information will arrive. There is however, because of the network's high reliability, a very high probability.

In order to ensure that the control centre does not suffer from lost information, commands are issued periodically to each node in turn requesting node status. In addition, the local routing algorithm is conditioned to inform the control centre task if any node becomes unreachable. If this happens, the status of the node and all its connected links becomes Unknown.

Once the node becomes reachable again, the request for link status is issued immediately. This mixed solution in which immediate automatic updates are expected, as well as periodic responses, involves inevitable redundancies. In so doing, it provides a simple, powerful and reliable means for the acquisition of complete network status information. More details are given in [17].

X.2 Message logging and analysis

In order to use both the local and remote facilities in an economical way, a centralized service, in the form of a logging task running in a virtual host distinct from the monitoring service, is necessary.

Several virtual hosts for logging are concurrently connected to the network, only one being accessible at any time. The distributed control services spontaneously send the text messages describing important events such as node reloading, link failure, hardware errors, etc. The logging service then appends to the received message the time of reception and stores this information on the IBM with the help of the File Access and the end-to-end protocols.

Since messages to be logged generally tend to arrive in bunches, the log task keeps the log-file open for a certain time after writing a message. This reduces the probability of very frequent requests to open and close the file, which would put an extra strain on the facilities in the logging node and the host File Manager.

The file which is written in this way is then available as a text file that may be listed, selectively scanned and edited by the standard processors within the host machine.
X.3 Remote console

Since analysts and engineers are not always close to a network node when they want to carry out a test, the facility of accessing the nodes remotely has been provided.

A logical link can be established to a specific task running in a virtual host. The first message indicates a password and the number of the CERNET node for which this logical link should simulate the console. Until the termination command is given all input sent to the virtual test via the logical link will be forwarded to the chosen node as if it were console input, and all output issued on that chosen node will be returned to the virtual host and then sent back along the logical link. Thus, the program which set up the logical link can act as if it were the physical console of the chosen node.

A Fortran program has been written in the service PDP-11 connected to Cernet, to use this facility. Since access to this computer is available via the local digital exchange from any terminal on site, network control and testing can also be carried out through any terminal.
CHAPTER XI
OPERATIONS

XI.1 Coordination

CERNET, from the operations group viewpoint, is a part of the facilities managed by the Computer Centre. These facilities comprise three principal areas of activity, namely CDC, IBM and Remote, the first two dealing with the mainframes and their peripherals and the third with all aspects of teleprocessing which includes CERNET. One important feature of CERNET was that its operation should be fully absorbed into the standard procedures of the Computer Centre, since, in addition to its specialised function of transmitting experimental data, it has the more general task of integrating the available host services across the mainframes. Figure 7 shows the items that need to be considered for successful operation and how they are interrelated.

![Diagram of Operational Co-ordination Scheme]

Figure 7: Operational Co-ordination Scheme

As the overall size of the computing facilities under the management of the operations group increased, the possibilities for errors of omission became considerable. For this reason, it was decided that a computerised management information system was indispensable as a
replacement for the present paper-chase. Most of the information relating to CERNET is in a form such that it can easily be incorporated into this system which is currently being developed.

Event logging is automated by means of a scheme for passing selected categories of messages originating in the communications subnet to a file in the IBM database. This file can be subsequently examined, using a Wylbur Exec File procedure, for messages relating to a specific system or class of error.

The complexity and extent of CERNET is such that an operator has neither the time nor the knowledge to resolve problems effectively without the aid of a very sophisticated system interface. The control centre with its associated colour display informs the operator of each significant change of state throughout the system and also whether an action is required on his part. Such a situation usually means that he must localise on the display the failing piece of hardware and contact the on-call person from the network team. Exceptionally he will, himself, deal with minor failures where software has to be reloaded, physical paths switched and routing tables altered, but where possible the Network Control Task will do this automatically as required. The control centre also gives the operator the recent history of the overall system in the form of a histogram of each component. It would be difficult to over-emphasise the dramatic advantage that the control centre as implemented in CERNET has over the conventional alphanumeric gobbledegook console output with which operators are still expected to interact, even on the most advanced systems.

Normally, an operator cannot expect to obtain technical help for CERNET problems outside working hours. For host mainframe failures, he is adequately supported and the experience to date with the communications subnet indicates a low probability of serious problems that would need the immediate attention of an engineer or analyst. His principal difficulty will be how to respond to requests from a large user community to deal with problems the origins of which will often lie in the user's own system. Nevertheless, it is the operator's task to satisfy himself that there is no failure in the network and to then convince the user that this is the case. To this end, he needs a good knowledge of the overall system, simple diagnostic tools both for the CERNET communications subsystem and the CERNET hardware and software on the user system, together with effective check-out procedures.

CERNET has, almost from its inception, been scheduled around the clock (apart from small breaks for reconfiguration at intervals) and this schedule includes the host machines and the associated network OMNET.

Such a schedule necessarily means that software development is carried out in parallel with the service, which entails a risk of partial failure. This risk is minimised by the overall redundancy. In addition, the built-in safety features ensure system integrity.

CERNET is continuously monitored for error conditions, for the availability of its paths and for the individual use of it by the subscribers. This enables intrinsic system component weaknesses to be located and corrected, to ensure that the service provided conforms to user expectations and to try to ensure that the limited services available are sensibly allocated to those users who have most need of them.

**XI.2 Performance statistics**

Table 1 summarises the principle statistics on Cernet since the beginning of the service. The availability is as seen by a selected important subscriber using one of the large host machines. The chosen user is the European Muon Collaboration and the host is the IBM, the connection passing via both Omnet and Cernet. The behaviour over the network is reasonably uniform so this measurement is representative of total system performance.

The lost time is shared between the external associated systems such as Omnet or the
IBM mainframe, and the failures in Cernet itself. This internal lost time is then divided into hardware and software. The weekly traffic is the data plus the overhead and is expressed in millions of packets and thousands of logical links.

<table>
<thead>
<tr>
<th>QUARTER</th>
<th>SCHEDULED HOURS WEEKLY</th>
<th>TOTAL ACHIEVED</th>
<th>LOST TIME</th>
<th>WEEKLY TRAFFIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXTER.</td>
<td>HARD.</td>
</tr>
<tr>
<td>1978/1</td>
<td>39.6h</td>
<td>68.3%</td>
<td>22.3%</td>
<td>4.2%</td>
</tr>
<tr>
<td>1978/2</td>
<td>38.3h</td>
<td>95.5%</td>
<td>0.9%</td>
<td>1.7%</td>
</tr>
<tr>
<td>1978/3</td>
<td>115.4h</td>
<td>96.4%</td>
<td>2.3%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1978/4</td>
<td>131.8h</td>
<td>97.8%</td>
<td>1.7%</td>
<td>0.2%</td>
</tr>
<tr>
<td>1979/1</td>
<td>145.0h</td>
<td>97.8%</td>
<td>1.3%</td>
<td>0.3%</td>
</tr>
<tr>
<td>1979/2</td>
<td>161.8h</td>
<td>98.0%</td>
<td>1.7%</td>
<td>1.0%</td>
</tr>
<tr>
<td>1979/3</td>
<td>161.6h</td>
<td>98.6%</td>
<td>1.1%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1979/4</td>
<td>155.0h</td>
<td>98.6%</td>
<td>1.3%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1980/1</td>
<td>157.9h</td>
<td>98.9%</td>
<td>1.0%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1980/2</td>
<td>159.1h</td>
<td>98.6%</td>
<td>1.2%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1980/3</td>
<td>161.4h</td>
<td>98.7%</td>
<td>1.0%</td>
<td>0.1%</td>
</tr>
<tr>
<td>1980/4</td>
<td>160.5h</td>
<td>99.1%</td>
<td>0.6%</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

Table 1: Quarterly performance

XI.3 Operational weaknesses

The environmental problems which are encountered in operating a variety of equipments spread over Cern are an ongoing cause for concern. The site is continuously undergoing major constructional changes causing many scheduled and accidental interruptions to power supplies and air-conditioning. The impact of these incidents is minimised by scheduling of services, by uninterruptible supplies and by redundancy but much remains to be done. In many cases the physical environment provided for the user machines is inadequate, a factor over which the
CERNET team has little control.

Diagnosis and rectification of faults within the communications subnetwork itself is a well established and effective procedure. It is exceptional if a problem is not despatched during the next working day. However, when the fault lies in the user machine or in his interface with the network the procedures break down. Efforts to date to create standard test software and establish simple diagnostic procedures, to be followed routinely by users when faults arise, have not been successful.

To keep proper statistical information on a network is a non-trivial task. To record and analyse the history of individual machines which are not easily accessible and do not possess sophisticated operating systems with management functions is not feasible without special tools. In principle most of these tools now exist and efforts continue to see that they are fully exploited.

i) The Cernet Transport Manager produces System Management Function Records detailing each logical link that is opened. These are analysed and provide an exact view of all utilisation of the network that involves the IBM. No similar facility exists for the CDC utilisation, although the dayfile holds some information.

ii) The Log Task which runs in one of the nodes receives a variety of information which is chiefly used by the engineers to trace faults. This information is passed on to the IBM. Some relatively modest developments of this function could produce an effective method of recording the history of the network components.

iii) Inevitably it is necessary to make use of operator logs, local machine console output, etc... to complete the automatically recorded information. Some progress has been made in constructing an operational data base into which this can be entered and from which information on related systems can be retrieved.
CHAPTER XII

NETWORK USAGE

The primary goal of CERNET was originally meant to be the transfer of raw data from the data acquisition minicomputers of the various physics experiments to the computer centre, and the return of results of the data analysis programs. In practice, the development of CERNET as a general purpose communications network has permitted many other types of use. This chapter is intended to give a survey of the various applications, and the way in which later types of use have developed and now represent the main bulk of network usage.

The changing use of CERNET has had a considerable effect on the computer centre itself. This centre is now seen as a collection of services, rather than computers, and the end users are accessing these services according to their needs. The services include guaranteed file storage or archiving, a variety of output devices (laser printer, microfiche, photo-typesetter etc.) and a set of input/output stations distributed around the laboratory. In many cases, the user of the services is not aware that CERNET is involved.

XII.1 On-line analysis

The concept of on-line analysis was basic to the design of CERNET as a vehicle for program to program communication between subscriber computers. It involves a program in a user minicomputer conducting a dialogue with a program in the central IBM computer.

The program in the IBM is basically the one normally used to analyse the experimental data. However, the input of the raw data for a single event is taken from a CERNET logical link and given to the program in the minicomputer rather than from local backing store. Likewise, some relevant part of the result of this analysis is returned to the minicomputer.

The typical use in the minicomputer involves, therefore, sending events to the IBM and receiving back results. The decision on when to send events and how to present the returned results is made at the minicomputer end. There is, of course, a basic requirement that the event be processed and the results returned within a timescale of a few seconds at most. This has already been discussed in Chapter VII.

The first users of this facility were also the first experiment to connect directly to CERNET, namely the European Muon Collaboration (EMC). Their mode of operation is that the data for individual events is treated as far as possible on the local minicomputer and then displayed in graphic form on a storage tube display. Any event which looks particularly interesting can then be sent to the IBM for analysis. The returned results, in this case particle trajectories, are then superimposed on the initial image for examination.

Although historically very important, this type of use of CERNET is now a minuscule part of the total use. There are, in fact, other applications in which a user has written a program in the IBM which talks to a minicomputer, but these are generally similar to the File Manager in that the objective is file transfer in one direction or the other.

60
XII.2 Data sampling

Another concept basic to CERNET was that of sending data samples to the central computers. The samples were to be collected on a single file during a period of time, after which the normal event analysis program could be initiated in the central computer.

It is essential to note that the concept of data samples does not, in general, imply centralised data recording. Instead, it means that the data acquisition program in the minicomputer may use any free time to send a percentage of events to the central computer, but with the standard data recording medium remaining as local magnetic tape. CERNET is not designed to accept the total data rates of several simultaneous experiments.

From the point of view of the minicomputer, a data sample may be created by selecting a small percentage (typically 5%) of events and modifying the data acquisition program to send these events to the File Manager of the desired central computer. When sufficient have been sent to make up a meaningful sample, transferring is stopped and the analysis program initiated in the central computer. This latter step may be done in various ways, including the sending of a job via CERNET.

An alternative method in the minicomputer is to run the normal data acquisition program for long enough to have a meaningful sample on magnetic tape, and then to run a separate program to transfer all events of this tape. This has the advantage of not requiring modifications to the data acquisition program, but may involve stopping data acquisition unless the two things can be done in parallel, either by sophisticated programming or on separate minicomputers.

Seen from the central computers, this is simply one of many types of incoming file transfer handled by the File Manager. However, the characteristics of the file involved are that it may be fairly large, but in general it will have a short lifetime. The existence of a special file type for such files has already been mentioned in Chapter VII.

Because of the similarity to other types of file transfer, it is not easy to be sure of the volume of such traffic. The available figures show it to be at most a few percent of total traffic. In addition the percentage is continually decreasing due to the increasing volume of other traffic.

XII.3 Individual file transfers

The transfer of individual files between minicomputers and the central computers, in particular the IBM, began to grow in volume as soon as standard file transfer programs were provided for the standard minicomputer types and operating systems. By 1978, they accounted for about three quarters of the traffic into or out of the IBM, at which time there were typically about 1500 files per week transferred, involving of the order of 40 megabytes of data. Since then, the number of files per week has only increased slowly, but the average file size has more than doubled, so that the corresponding weekly traffic has reached the 100 megabyte range.

The first use of standardised file transfer involved text files. A frequent reason was the popularity of the Wylibur system on the IBM for the manipulation of text files for which the master copy was to be held in the filing system of a minicomputer. However, it is not unknown for the reverse to happen, i.e. a minicomputer user to recall a text file from the central computers in order to manipulate it with the local text editor.

File transfer is also useful when the computer-centre hosts are used to run cross-assemblers, cross-compilers and link loaders for minicomputers. The adoption by CERN of a universal object module format (CUFOM) [12] has made this a popular means of minicomputer support. The file transferred from the computer centre to the minicomputer can be a relocatable or absolute object module, which may be loaded on the fly or stored on disk.

File transfer between minicomputer and mainframe may also take place because of limitations of disk space on the minicomputer. This is effectively using the mainframe as a
backup store. Invariably, the mainframe used is the IBM, since it has the mass store attached plus the Hierarchical Storage Manager (HSM) program to migrate files between mass store and disk.

If the local minicomputer has a hard-copy output device (normally a printer) another use involves programming the mini as a remote output station. For this purpose a program in the mini must regularly poll the computer centre hosts, i.e. the File Managers, in order to know whether there are any output files which have been created by a normal host job. The flagging of such files is done by means of job control statements. The File Managers are able to give information as to whether any such files exist. If so, the minicomputer program may ask for the files, one by one, and print them on the local printer. This is particularly useful for the general purpose minicomputers, including one which is dedicated to handling all plotter output and another for photo-typesetting output.

It is clear from the above that the reasons for file transfer between minicomputers and the computer centre hosts are many and various. In the future, it is certain that some particularly powerful and well-equipped minicomputers will be provided with File Managers, thus becoming CERNET hosts, in order to provide a service, in general for the benefit of other similar, but less well-equipped, minis.

XII.4 Disk backup

A standard problem with minicomputers is the precautions to be taken in order to safeguard the disk database. The traditional method is to dump the whole data base periodically onto magnetic tape, where the period might typically be once per week. A sophisticated scheme will then call for daily dumping of all files changed since the previous dump. The problem is to find people willing to do this, since the most convenient time is usually when no-one is using the computer. Even if this is done then the tapes have to be controlled, labelled etc. by someone on the spot. Also, recuperation of the data base in case of error requires someone with the necessary knowledge to be on hand.

Many of the above problems exist in large computer centres. However, a large computer centre necessarily has the manpower to do the tape mounting, dismounting, administration and so on. In addition, the centre can make use of the IBM mass store itself.

It therefore was not long before some users of minicomputers with important data bases considered using the IBM as a backup data base for their own. Again, the standard method is to send a complete disk image to the IBM via CERNET at regular intervals. The format for the data transfer can be chosen to maximise efficiency and minimise transfer time, since the image can simply be a bit stream with no separation of files.

Once the above procedure has been established then it can easily be scheduled for night-time execution. In some cases, it may even be possible to schedule it automatically, so that human presence is no longer required.

It is still true that a periodic incremental dumping of changed files is a useful complement. In addition, the minicomputer must have the necessary programs to reconstruct the data base if necessary. Furthermore, the recall of individual files from the IBM via some utility program can be useful. However, such programs need to exist whatever method is chosen. The big advantages of using the IBM thus remain as automatic backup, without human intervention, at a convenient time of day.

XII.5 Inter-mainframe traffic

The volume of traffic represented by file transfers between the IBM and the CDC is currently the fastest growing load on CERNET. This stems from the fact that each mainframe has some facilities not available on the other one, but of interest to all users. In particular, the IBM has a laser printer, a mass-store and, very recently, a microfiche output device; the CDC
has a collection of remote input/output stations distributed all over the CERN site. In addition, with the existence of large collaboration experiment groups, it is not infrequent to have programs which may execute on either the CDC or the IBM, and so it becomes necessary to move both programs and data between the two systems. In fact the amount of data transferred between the two mainframes via CERNET has grown to become the majority of total CERNET traffic.

When the inter-mainframe service began, the first main use was the transfer of files from IBM to CDC, in particular the output files of IBM jobs to be printed at a suitable CDC Remote input/output station. A later addition was for the submission of jobs to the CDC 7600 from a Wyibur terminal. By the end of 1980, this direction of traffic was accounting for about 5400 such files per week, totalling approximately 275 megabytes. This probably corresponds to a sustained rate of about 1.3 kilobytes/second during most of the standard working hours.

The basic method for transferring these files to the CDC (and also for files in the reverse direction) is an IBM program which is scheduled to run every few minutes. The program interrogates the File Manager on the IBM to see if there are any files flagged for transfer. If necessary, it then creates a suitable logical link for the transfer of one or more such files.

For the transfer in the reverse direction, the object is, in general, to retrieve output files. These may be destined to be returned to Wyibur users, to be printed on the IBM laser printer or on microfiche, or to be executed as a job on the IBM. The method is similar, with the obvious difference that it is the CDC File Manager which is interrogated by the IBM program. The controlling program has been made different in order to permit transfers in both directions simultaneously. By the end of 1980, this type of traffic had risen to a weekly total of some 500 megabytes.

For the average user wanting to transfer a single file of a standard type, there are ready-prepared jobs which are invoked by catalogued procedures after the user has specified the relevant parameters (file name on CDC and IBM, type, direction of transfer etc.).

Finally, it may happen that a user either has a slightly non-standard file or wishes the file transfer to be only a part of a complete job execution on the IBM. In this case, there are utility programs which he can execute via suitable JCL statements in the job. This use is quite convenient for obtaining files from the CDC.

XII.6 Miscellaneous

Some developments in the use of CERNET do not belong to any category previously described. Essentially, they all involve a program on the IBM making use of CERNET to talk to one or several programs elsewhere using the Transport Manager UIP.

One of the first of these was what is called the "Wyibur bridge". In terms of software, this is actually an addition to the Milten subsystem to allow access from CERNET to Wyibur. The use is for terminals hard-wired to minicomputers which themselves are connected to CERNET. The minicomputer is then programmed to transmit commands from the terminal to Wyibur via CERNET, and to return output to the terminal, i.e. it is a terminal concentrator. This application is increasing in popularity, both for the hard-wired terminals and for the times when there are no free connections for INDEX terminals via the IBM 3705 multiplexer.

In this context, it should be noted that an alternative development for PDP-11 computers, called mini-Wyibur, offers similar possibilities. The difference is that mini-Wyibur is a program which duplicates many of the standard Wyibur commands on the local PDP-11. However, the filebase from which the files are used or saved may be either the local filebase or that of the IBM. Access to the IBM files is made via CERNET and the IBM File Manager. This concept could be applied more generally if so desired.

Doubtless, as time goes on, new uses will appear. One already in advanced state of
development is not dissimilar to that just described. It is concerned with the 168E project, in which several 168E special-purpose microcomputers which emulate part of the IBM 168 instruction set will take over the CPU-intensive processing for physics data for one or two experiments. The data will be held in the IBM and passed, event by event, under the control of an IBM program via CERNET to a PDP-11. The PDP-11 will then distribute the events for processing to any convenient 168E and eventually return the results for each event to the IBM. It is clear that the IBM program is another special-purpose application for which the standard File Manager is not equipped.
CHAPTER XIII

RELIABILITY, PERFORMANCE AND THROUGHPUT

The judgement of the success or failure of a project such as CERNET must ultimately be based on the two factors, reliability and performance. In addition, they are two highly-interdependent factors, neither of which can be sacrificed to satisfy the other.

The reliability aspect has been tackled in a number of ways, the main ones of which are back-up hardware, high-level language software and good diagnostic and recovery aids. These of necessity slow down the performance as measured by the theoretical hardware data transfer rate versus the actual measured overall performance. However, the hardware data transfer rate is sufficiently high that the end result is still quite adequate for all the needs originally foreseen and for others which have arisen since.

The figures given in the following sections are based upon an analysis of the system log output of the two IBM computers, on which information is inserted concerning all CERNET activity on the IBM, and of the CERNET node console outputs. These latter are used to produce a regular operation report including performance and reliability statistics. The operation report assumes availability and reliability to mean that of the main IBM service as seen from the viewpoint of the OMNET gateway.

XIII.1 Reliability

The criterion of ‘reliability’ for a multi-node, multi-host, multi-user system such as CERNET is extremely difficult to establish. Any individual user may have a clear idea of reliability as the percentage of success he has in connecting to a program on another subscriber computer and conducting a meaningful dialogue without harmful interruption. However this does not give an overall definition of reliability.

Even the above simple user definition is variable, in the sense that the user’s view may vary according to the extent to which he is informed of scheduled unavailability of the computer to which he wishes to connect.

For a very large number of users the reliability is essentially the availability of the prime IBM service, including the File Manager, via CERNET. In principle, this is near to 100% except for occasional scheduled periods when some special action requires both IBM 370/168 and 3032 to be removed from normal service, or when a major reconfiguration of the CERNET communications subnetwork is taking place. Such happenings might involve a scheduled down-period of up to 4 hours but would cause at most one interruption per month.

For those users requiring access to the CDC File Manager the availability is much less, since the CDC service on the front-end (MFA, the CYBER 170/720) is not scheduled to be 100% available.

Even for the IBM, there are a number of short scheduled unavailability periods, which reduce the availability by one to two hours per week. Typically, these might involve an Initial
Program Load (IPL) to transfer services between the two IBM computers.

An examination of the statistics for a 30-week period just prior to the completion of this report shows that on average the main IBM service was scheduled to be available to users via the OMNET gateway for 160 hours 20 minutes per week, and was actually available for 158 hours and 30 minutes per week, i.e. 98.82% of the scheduled hours. Of the lost 1.18% a small part (0.16%) is due to CERNET node software problems, a miniscule part (0.04%) to CERNET data-link hardware problems. The main part of the loss (0.98%) is due to unscheduled problems outside the domain of CERNET node hardware and software, i.e. mainly due to IBM problems.

It can be seen that reliability of the CERNET data-link hardware is extremely high (99.96%). A practical observation is also that a data-link is either working perfectly, with a zero error rate, or indicating regular errors which are reported by the driving software and dealt with rapidly. Even a failing data-link has enough in-built checks that data integrity is maintained; in general, the failing link finally becomes completely unusable and the CERNET communications subnetwork reacts accordingly.

XIII.2 Performance

As with reliability, there are many differing ways to measure performance. To the person who has initiated data transfer between two computers a simple measure is the data rate, i.e. the amount of data per unit of time. However, this rate may depend on a large number of factors. Some are fixed, such as the efficiency and sophistication of the software packages in the two communicating subscriber computers, and the number of CERNET nodes on the chosen route. Others are variable, in the sense that they may change with time. Important examples of the latter are the load on the two subscriber computers and the priority accorded to each communicating process within its own computer system. The type (e.g. ASCII or binary) and size of each unit of data transferred also have an important role.

One fixed measure of performance is the throughput of a single CERNET node computer in terms of packets per second. Due to the very high speed of the hardware this throughput is essentially a function of software handling time only, i.e. it does not vary with packet size. Measurements give the throughput as about 70 packets per second for a Modcomp II and 100 packets per second for a Modcomp Classic.

In the extreme case, hypothetically possible, where virtually every packet carries a full quota of data, the 100 packets per second would give a data rate approaching 200 kilobytes per second. Such an extreme case would require very long multi-packet messages and very sophisticated user software.

At the other end of the scale, data could go as single partially-filled packets each of which requires many control packets. In fact, the end-to-end protocol can permit the ratio of control (i.e. non-data) packets to data packets to rise to 7 to 1.

Faced with all these factors the only way to get consistent figures is to run standard test programs at non-busy times. This results in optimum times for data transfer of particular types.

Two types of test program exist. In one, a subscriber creates random data and sends it to another subscriber, who may either return it or throw it away. The two options of this first type of test program may be called BOUNCE/ECHO or SOURCE/SINK respectively. In the second type, a test program in a subscriber computer creates random data to send to a file on a chosen host, via the host File Manager, and then reads back the file.

The various figures given below refer to data transfers in which the amount of data per message varies from 2 kilobytes (one packet per message) to 6 kilobytes (3 packets per message).

A typical BOUNCE/ECHO (or SOURCE/SINK) from a PDP-11 via OMNET to the
IBM may be from 15 to 30 kilobytes per second. From a Nord-10 or HP-21mx the rate is a little slower, due to internal communication speeds in the Nord-10 or HP-21mx. From a directly connected PDP-11 the rate may be appreciably higher (20 to 40 kilobytes) and from a directly-connected VAX-11/780 it may be the double (30 to 60 kilobytes).

BOUNCE/ECHO between the IBM and the CDC, via a single node, may typically vary from 35 to 70 kilobytes/second. File transfer from the IBM to and from the CDC File Manager can, in the limit, also reach 35 to 70 kilobytes per second, using a very optimised test program. However, a normal rate is 30 to 60 kilobytes per second.

An interesting SOURCE/SINK figure is that for transfer between two adjacent CERNET nodes, each having a full version of this portable Transport Manager and a pipelined mode of data transfer (i.e. a continuous flow of messages) a rate of almost 50 kilobytes/second has been achieved for single-packet (2 kilobyte) messages. This also demonstrates that making a Transport Manager ‘portable’ does not necessarily make it slow.

XIII.3 Throughput

By “throughput” is meant the total traffic passing through CERNET. To a first approximation this is equivalent to the traffic into or out of the IBM, even though this excludes communication from minicomputer subscribers to the CDC.

The early usage of CERNET was oriented towards file transfer between IBM and minicomputers, and it accounted for virtually all of the traffic. At the end of 1978 it consisted of about 1500 files per week, with a slight majority for files sent to the IBM. The growth since then has been slow but steady, so that by late 1980 the number of files had approximately doubled. Also more emphasis has been placed on retrieval of files from the IBM.

In 1979 CERNET began to be used for inter-mainframe traffic (IBM-CDC). The growth of such traffic has been spectacular. By the end of 1980 such traffic has risen to be over 60% of the total messages into or out of the IBM, i.e. approximately 2 messages in every 3 are between IBM and CDC. In fact, since such messages are longer than the average, the percentage of real data certainly exceeds 75% of the total real data being moved into or out of the IBM.

In terms of logical links seen by the IBM the statistics have to be interpreted correctly. There are a number of services which are of a polling type, i.e. which involve opening a link at regular intervals. The link may or may not be used for transfer of real data. Currently such links amount to about 30% of the 15000 or so logical links opened per week.

A very recent growth area is the "virtual terminal", in which CERNET is used to allow a terminal of a user computer to act as a terminal of the IBM terminal system Wylbur. A single logical link then corresponds to a single logged-in session. The current number of such links per week has exceeded 1000 and is still increasing. Of course, in terms of the number of logical links or amount of data transferred this use is still relatively small, but it does indicate how new uses can become popular.

In general, it can be said that the CERNET traffic is approximately doubling each year. The figures per week as of late 1980 show about 15000 logical links per week, with a total message input to the IBM of about 300000 and an output of 250000, i.e. a total of 1 message (either in or out) every second.
CHAPTER XIV

FUTURE DEVELOPMENTS

The main development work for CERNET has now been completed and the network is an established feature of the computing facilities of the Laboratory. Extensions, in the sense of connections of more subscriber computers and nodes, will be made as and when the need arises, but the emphasis will be one of operation of a stable system for the next few years.

This is not to say, however, that no further improvements will be made to the network. There are several areas where development will be continued, as part of the routine improvements to an operational system. These are, at present, in network software, in link interfaces, and in connections to other networks.

XIV.1 Software

The Modcomp Classics have been used so far essentially as rather faster Modcomp II/45s in the network. The additional facilities which they offer, especially virtual memory mode, will gradually be brought into use leading to an increase in the number of possible data-link connections to the network.

The monitoring facilities of the control centre are being used to collect information on network performance, in particular, to identify bottlenecks and analyse traffic patterns. This data will be used to indicate the appropriate software improvements in subscriber and node computers to increase the throughput of the network. Spectacular increases in the performance are not to be expected, but increases of a few tens of percent may be made over the next few years.

The asymmetry between the CDC and the IBM Transport Managers will be removed, making the CDC also an “active” host, such that user programs running in the CDC system can initiate network activity.

Most probably, other hosts from amongst the CERN preferred minicomputer types will be added to CERNET in 1981 or 1982, the most likely first candidate being the PDP-11 or VAX-11.

XIV.2 Data-Link Interfaces

The requirement for a user computer to be equipped with CAMAC and to be sufficiently large to allocate enough memory for the full Transport Manager and User Interface Package is an important limitation on the computers which can become standard CERNET subscribers. Whilst this is understandable when seen in the light of the initial design aims of CERNET to support minicomputers with CAMAC in physics experiments, it becomes an undesirable restriction when seen in the light of the more general purpose network into which CERNET is evolving. This “configuration overhead” is too expensive for some users of small minicomputers who could reasonably benefit from a CERNET connection.
Work is therefore going on to alleviate these problems, and two approaches are being made. Firstly, to use a microprocessor CAMAC module to perform all the Transport Manager functions, thus removing the memory overhead from a subscriber computer which already has a CAMAC interface. Secondly, to develop a microprocessor interface card which can be easily adapted to the preferred types of minicomputer at CERN, and which will permit a CERNET data-link interface to carry out at least the line protocol function, and perhaps the full end-to-end protocol.

The provision of a much slower, but also cheaper, CERNET data-link interface, probably using the HDLC standard, will also be followed up for users of small minicomputers or microcomputers who do not require the high speed links.

XIV.3 Connection to other networks

The initial design of CERNET was made before the CCITT X-25 standard [8] was established. Consequently, CERNET has its own protocols and these are not directly compatible with X-25 which will be used in the National PTT packet switching networks coming into operation over the next few years in the Member States.

A gateway, or "protocol converter system", is therefore being developed, using a microprocessor CAMAC module to provide the X-25 interface levels 1, 2 and 3. This interface will be driven by a CERNET node which will carry out the necessary protocol conversions to allow computers in Laboratories in the Member States to be connectable as CERNET subscriber computers, via the PTT public packet switching networks.

The gateway is also being designed in such a way that future network developments at CERN which may use the X-25 protocol may be readily connected to CERNET.

The use of the CAMAC system with microprocessor modules for X-25 and for the Transport Manager opens up the possibility of being able to build up X-25 packet-assembly (PAD) services and other facilities into CERNET in a very flexible way which we plan to exploit in the future.

Finally, with the advent of Local Area Networks which will certainly appear at CERN in the near future, provision will have to be made for these to have access, if required, to their nearest CERNET node.
REFERENCES

1) Ball, D. et al,
   FOCUS — A remote access file handling system on-line to a
   CDC 6000 series computer,

2) Parkman, C.F. and Lee, J.G.
   OMNET— High Speed Data Communication for PDP-11 computers
   CERN Yellow Report 79-11, 1979

3) Modular Computer Systems Inc.
   Modcomp Classic Central Processor Reference Manual
   Publication No. 210-140000-000, 1978

4) Modular Computer Systems Inc.
   Modcomp II Central Processor Reference Manual
   Publication No. 210-102000-000, 1975

5) Modular Computer Systems Inc.
   Maxcom Communications Executive Reference Manual
   Publication No. 210-600310-000, 1975

6) EURATOM Reports
   EUR 4100, 1972
   EUR 4600, 1972
   EUR 6100, 1973

7) Fluckiger, F. and Piney, C.
   Comparison between X25 and CERNET Architectures
   CERN Data Handling Division Report DD/80/18, 1980

8) I.T.U. Geneva, Switzerland
   Draft Revised CCITT Recommendation X.25
   COM VII — No.489, 1980

9) Bruins, T and van de Kerk, P.J.
   Asynchronous Serial data transmission at 2.5 mbps.
   CERN Data Handling Division Report DD/75/24, 1975

10) EURATOM report
    Subroutines for CAMAC

11) Richards, M.
    The BCPL Programming Manual
    The Computer Laboratory, University of Cambridge,

12) Montuelle, J.
    CUFOM, the CERN Universal Format for object modules

13) Fjellheim, R.A.
    A Message Distribution Technique and its Application to
    Network Control.
    Software Practice and Experience, 9 (1979) p 499.

14) Modular Computer Systems Inc.
    Model 1950 Modcomp-to-IBM 360/370 Interface
    Publication No. 225-200127-001, 1975

15) Modular Computer Systems Inc.
    Modcomp-to-CDC Satellite Coupler Model 5943, CERN special,
    Technical Specification (1976) and
Modcomp-to-CDC Satellite Coupler Model 1941

16) Piney, C. et al

17) Piney, C. and Fluckiger, F.
Principles of Control in a Distributed Network CERN Data Handling Division Report DD/80/4, 1980
APPENDIX A: LIST OF NETWORK PROJECT NOTES (NPNS)

The following is a list of titles of the current Network Project Notes, listed by their NPN number. Copies are available on request, as long as stocks last, from the Data Handling Division Secretariat, CERN, CH-1211 GENEVE 23, Switzerland.

2 Network Virtual Terminal C. Adams, B. Segal
7 Tasks and Timeouts C. Piney, M. Gerard
8 Modcomp Assembly Language coding conventions M. Gerard
9 Loading and Dumping on the Modcomps H. Davies, C. Piney
10 CERN modifications to MAXCOM M. Gerard
11 MAXCOM System Generation M. Gerard, C. Piney
12 Modcomp II System Protect Feature usage M. Gerard
13 The Line Protocol H. E. Davies, C. Piney
14 A 9600 Baud Line Protocol for the Network C. Piney
16 End-to-End Protocol The Network Team
23 Modcomp Teletype Dump program M. K. Downie
24 Modcomp-CDC Satellite Coupler Handler (CERN version) M. Gerard, M. K. Downie
25 Specification of Modcomp Links J. Joosten, R. Pieters
27 General information on jobs and Files at the CDC Centre H. E. Davies
28 Access to Files at the CDC Centre through CERNET J. Blake
29 Symbionts under MAXCOM C. Piney
30 The Line-Protocol implementation C. Piney

73
<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCL—an Interactive Debugging program for BCPL under MAXCOM</td>
<td>M.Richards, C.Piney, M.Gerard</td>
</tr>
<tr>
<td>Job submission to the CDC Centre through CERNET</td>
<td>J.Blake</td>
</tr>
<tr>
<td>PDP11 CERNET Data Link Interface</td>
<td>C.Parkman</td>
</tr>
<tr>
<td>CERN Changes to MAXCOM REX calls</td>
<td>M.Gerard, C.Piney</td>
</tr>
<tr>
<td>The transmission system for the Network Links</td>
<td>J.Joosten</td>
</tr>
<tr>
<td>Engineers Diagnostic Programs ED1, ED2, ED3</td>
<td>J.Joosten</td>
</tr>
<tr>
<td>Network Stage 1 Modcomp software</td>
<td>M.Gerard, C.Piney, R.Fjellheim</td>
</tr>
<tr>
<td>Specifications for the Interface to the future version of the BCPL Portable Transport Manager</td>
<td>C.Bloch, M.Gerard</td>
</tr>
<tr>
<td>BCPL Tasks under MAXCOM</td>
<td>M.Gerard, C.Piney</td>
</tr>
<tr>
<td>Allocation of contiguous core by MAXCOM</td>
<td>M.K.Downie, M.Gerard</td>
</tr>
<tr>
<td>The MVS version of the Modcomp 1950 Interface Diagnostic Program</td>
<td>D.Wiegandt</td>
</tr>
<tr>
<td>Network Computers, Topology and Routing</td>
<td>M.Gerard</td>
</tr>
<tr>
<td>The CERNET Portable Transport Manager</td>
<td>J.Nerdal, N.Jacobs, C.Bloch, M.Gerard</td>
</tr>
<tr>
<td>Event tracing in Network Nodes</td>
<td>M.Gerard</td>
</tr>
<tr>
<td>Network Node Link Control Tables</td>
<td>M.Gerard, R.Fjellheim, C.Piney</td>
</tr>
<tr>
<td>MAXCOM Task Pseudo-Console Switch Register</td>
<td>M.Gerard, C.Piney</td>
</tr>
<tr>
<td>MAXCOM Operator Control Task</td>
<td>M.Gerard, C.Piney</td>
</tr>
<tr>
<td>No.</td>
<td>Title</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>54</td>
<td>Using the CDC Network Interface Program</td>
</tr>
<tr>
<td>55</td>
<td>MAXCOM Non-Resident Tasks</td>
</tr>
<tr>
<td>56</td>
<td>Network traffic statistics</td>
</tr>
<tr>
<td>57</td>
<td>Network Control-Overview</td>
</tr>
<tr>
<td>58</td>
<td>FORTRAN Functions as interface to CERNET in the IBM-370</td>
</tr>
<tr>
<td>59</td>
<td>Local Node Control</td>
</tr>
<tr>
<td>60</td>
<td>CDC Line Protocol Symbiont (SCB)</td>
</tr>
<tr>
<td>61</td>
<td>File Conversion programs RDNET and WRNET</td>
</tr>
<tr>
<td>62</td>
<td>Remote Console</td>
</tr>
<tr>
<td>63</td>
<td>The SPY Task</td>
</tr>
<tr>
<td>64</td>
<td>Fill-Rom Control and Status for Modcomps</td>
</tr>
<tr>
<td>65</td>
<td>CERN Datalink 'Fill' Procedure</td>
</tr>
<tr>
<td>66</td>
<td>CERNET Remote File Access: Test software for PDP-11</td>
</tr>
<tr>
<td>67</td>
<td>FORTRAN functions as interface to CERNET in the Nord-10</td>
</tr>
<tr>
<td>68</td>
<td>Modifications to 6000 SCOPE for CERNET</td>
</tr>
<tr>
<td>69</td>
<td>Packet Routing</td>
</tr>
<tr>
<td>70</td>
<td>CERNET Remote File Access: User Interface calls for PL-11 and MACRO-11</td>
</tr>
<tr>
<td>71</td>
<td>A CERNET File Transfer Utility for the PDP-11</td>
</tr>
<tr>
<td>72</td>
<td>Access to IBM Data Sets through CERNET</td>
</tr>
</tbody>
</table>
73 Program NETCON for control of Permanent File space with ID=NETWORK
   D.Wiskott

74 RSX-11M Implementation of the CERNET Transport Manager
   J.I.Nerdal, M.K.Downie

75 Recent additions to the IBM File Manager
   D.Wiegandt

76 CERNET Network CAMAC Link
   J.Anthonioz-Blanc, J.Joosten

77 On-Line translation of CERNET character files in MFA
   J.Blake

78 A Virtual Terminal Protocol for CERNET
   R.P.Watts

79 The PDP-11 BOUNCE /ECHO Program
   M.K.Downie

80 A File Utility for CERNET
   R.Fjellheim

81 Terminal Identifiers for CERNET users on MFA
   J.Blake, I.McLaren

82 Error Conditions in CERNET usage
   M.Gerard

83 Proposed CERNET Message System
   C.Piney

84 CERNET test software
   B.W.Evershed

85 The Modcomp to CDC Satellite Coupler Diagnostic Test Program
   J.Blake

86 The Modcomp to CDC Satellite Coupler Diagnostic Task DCB
   J.Blake

87 UMS, a Modcomp to CDC Satellite Coupler Test Skeleton
   J.Blake

88 Simple File Transfer and Bootstrapping
   C.Piney

89 Modifications to the BCPL run-time system for the NWACP Task
   M.K.Downie

90 The new set of Transport Manager Interface Routines (NT-Routines) on the IBM
   D.Wiegandt
91 Extensions to the CERNET File Manager in MFA

J.Blake
APPENDIX B: PRINCIPAL MODCOMP TASKS

Herein is a more detailed description of the various tasks which make up the systems which execute on the Modcomp nodes.

Unless otherwise stated, all tasks are coded in BCPL.

TNC (Task for Network Control) Standard task available on all systems, always resident, started automatically at system start-up.

TNC is the basic task for node control and packet switching. It maintains information on the state of all communication links for the node. By exchanging routing information with adjacent nodes, it maintains tables which enable it to decide upon the correct route for any given destination. It can accept additional input from the console (local or remote) modifying the routing tables, the link tables, etc. It will answer requests for information about throughput, state of links etc., where the question may come from the local console, a remote console or a task such as the control centre (TCC).

TNC is fully described in NPN 59.

TSW (Task for Simple Packet-Switching) Standard task available on all systems, always resident.

Most of the packets which are received by a node are destined for immediate forward transmission according to the destination. TSW is the task which normally controls receipt and forwarding, using the routing information as set up by the local network control task TNC. In any special case, such as error conditions, packet addressed to this node, temporary memory overload etc., the processing is simply handed over to TNC.

TCC (Task for Control Centre) Standard task only available on systems with a T4O27 control centre display, can be resident or non-resident. If resident it is automatically started at system start-up.

The control centre being intended as the visualisation of the state of CERNET, plus the ability to enter commands, on a colour display (Tektronix T4O27), the job of TCC is to collect information and display it. TCC is able to interrogate the network control task (TNC) of any node in order to find link status, throughput etc. The available forms of display include the whole network or particular subsets, focussing upon a particular node or giving a history of previous events (e.g. change of state of nodes). To allow easy choice of display function, many of the keys of the T4O27 are software programmed as function keys.

SCI (Symbiont for Console Information) Symbiont task available on all systems, always resident.

This symbiont is used to implement what is known as the 'Remote Console' facility. A full description appears in NPN 62.

The function of SCI is to intercept task requests for input or output on device 0 (zero), which is considered as the local console in standard MAXCOM. By means of 'prefix sets' messages may be routed to one or more other nodes, where they may be printed or directed to a named task. An escape character allows the bypassing of these prefix sets, thus permitting tasks to send messages to a specified node either for printing or as input to a partner task.

SCB (Symbiont for CDC Transfers in Both Directions) Symbiont task only available on systems with a CDC coupler, always resident.

This symbiont task enables the high-level tasks (TNC, TSW) to treat each of the one or more links to CDC couplers as a pair of standard network intelligent devices, one input and
one output. Since the drive is physically a single half-duplex channel connection are treated both pseudo-device queues (the input queue and the output queue) in parallel. Actual calls to the device are made via the CDC coupler handler.

As with other devices, the symbiont can be told to cease use of the CDC coupler. Such is the case either for a CDC connection held as a reserve or for an on-line diagnostic task (DCB) which itself calls the same handler directly.

SCB is capable of handling multiple CDC couplers. To do so, it uses the concept of processes as mentioned in section VII.2.

**SIB (Symbiont for IBM Transfers in Both Directions)** Symbiont task only available on systems with an IBM coupler, always resident.

This symbiont task is the IBM equivalent of the CDC symbiont SCB explained already. It enables the high level tasks (TNC, TSW) to treat a single half-duplex channel connection to an IBM as a pair of standard network intelligent devices, one for input and one for output. Like the CDC symbiont, it treats both pseudo-device queues in parallel, using calls to the real IBM coupler handler.

**SPI, SPO (Symbionts for CERNET data-link Input/Output)** Symbiont tasks available on all systems, always resident.

These symbionts enable the high level tasks to treat the CERNET data-links as a pair of standard network intelligent devices, one input and one output. They are separate tasks because the CERNET data-links are already full-duplex. They are treated as a separate input and output device, each with its own handler software. In practice, there are a certain number of routines common to both, which are put into a subsidiary library common to both symbiont tasks.

SPI and SPO are the most striking examples of the treatment of multiple links by single BCPL programs. To do this they both use the concept of processes as described in section VII.2.

A particular feature of SPO is the implementation of down-line loading, for which SPO obtains the down-line load file by conducting a dialogue with LDT and the PUL task.

**NRT (Task for Loading Non-Resident Tasks)** Standard task available on all systems, always resident.

When an attempt is made to activate a non-resident task, the MAXCOM system sets a task status flag bit and activates task NRT. It is then the job of NRT to ascertain the required memory space, reserve the space, obtain a relocatable object module for the required task, relocate it into the reserved space, fill in the correct task start, end and transfer addresses into the task control block and then activate the required task.

To obtain the object module, and information about it, NRT interacts with the PUL task, which actually fetches the desired file from the specified host.

**PUL (Pull Object Module File from Host)** Standard task available on all systems, always resident.

In order to provide non-resident task loading or down-line loading of absolute object modules into adjacent nodes, it has to be possible to obtain a desired object module from a specified host. PUL is the task which performs this function. PUL may act in conjunction with NRT for non-resident task loading, or with SPO and LDT for down-line loading. Its conversation with the host, currently always the IBM, goes to the File Manager. For this purpose, PUL uses the shared library of File Manager user interface package (SUIP) also used by UTY and LOG.
**LDT (Load/Dump Task)** Standard task available on all systems, always resident.

This task carries out the basic steps required for downline loading, dumping and simple file transfer. It receives and analyses a 'file description block' from the target machine and then opens the required files. The dialogue with the target, PUL and the File Manager on the host is then orchestrated by this task, which also looks after tidying-up after error and closing correctly on completion.

**TMG (Transport Manager Task)** Standard task only available on selected systems, non-resident.

TMG is the implementation of the machine-independent Transport Manager (described in NPN 48) for the Modcomps. It provides a testbed for proposed modifications to this Transport Manager, as well as a way of communicating with other CERNET-connected computers. However, this latter use is currently the function of the symbiont STM, since there is not the requirement yet for the sophisticated error-recovery procedures of TMG.

**STM (Symbiont for Simple Transport Manager)** Symbiont task available on all systems, always resident.

The requirements of a CERNET node as regards communication with subscribers or hosts are normally very straightforward. In general there is no need for multiple logical links from any one task, nor is there any need for sophisticated error recovery, since transfers are of relatively short files and the whole transfer can be repeated in case of error. Therefore, STM enables a node task to talk to any other network-connected computer, using CERNET end-to-end protocol. For the calling task, STM operates as a pseudo-device capable of opening/closing a link and sending/receiving complete messages.

**OCT (Operator Control Task)** Standard task, available on all systems, always resident.

This is a modified version of the original assembly language task provided by Modcomp. It is used to input date and time, activate, hold or resume tasks, control the pseudo-console register and purge excessive console output of individual tasks which may temporarily get too verbose. It has other functions, such as changing task priorities (very rarely done) or inspecting or modifying memory locations (normally done more easily via the BCL task).

**BCL (BCPL Debug Task)** Standard task available on all systems, normally as non-resident. If resident it is automatically started at system start-up.

The various possibilities with this task are described in NPN 32. It is particularly useful when testing new versions of software, or looking for difficult bugs. The provision of software-implemented break points is extremely useful in such cases.

**LOG (Task to Log Selected Messages on a Central Log File)** Standard task only available on selected systems, resident on one selected system and non-resident on others. If resident, it is automatically started at system start-up.

The LOG task is ready to receive messages from anywhere in the network. Any message received will then be copied to a log-file held on the IBM, accompanied by information giving the date, time, sending machine and sending task name. If the IBM is unavailable, the message goes onto the local console log, and, when the IBM again becomes, available a separate message indicates the fact that some messages went to the local console.

It is the responsibility of individual tasks to decide what messages are important enough to go to the log file. They then send these messages to the LOG task on a virtual host (F8). This virtual host is normally assigned to a fixed real node on which LOG is resident. In case of problems, LOG can be loaded dynamically into selected other nodes and will automatically
handle the change of assignment of virtual host.

*UTILITY (Utility Task)* Standard task only available on selected systems, normally non-resident

This utility task is used for the transfer of core load object files from the IBM onto floppy disk, or the transfer of dump files from a floppy disk to the IBM. It allows the user, via console input, to specify options concerning IBM file identification and location of data on the floppy disk. It shares the subsidiary library (called SUIP — Simple User Interface Package) with any other tasks liable to access files on the IBM.

*EFF (Efficiency-measurement Task)* Standard task available on all systems as a non-resident task.

The efficiency of a running system is basically measured as the ratio between the time spent by the system in its idling task and the time spent in useful work. EFF measures this ratio by comparing over given periods of time the amount of time spent in the idle loop against the real time. EFF can keep a running average of efficiency and can print out results as often as required.

*SPY (Task to Spy on System Resource Usage)* Standard task available on all systems, non-resident.

This task is able to spy on the CPU utilisation of a running MAXCOM system by inspecting, at regular intervals, the program counter address at the previous highest-level interrupt (the clock interrupt). It produces a histogram of percentage usage. The histogram may represent a certain region of memory, divided into equal sub-regions according to user input. In this case, the user may choose to consider all tasks or one specific task. Alternatively, the histogram may simply be the relative use of the system by the various tasks, thus indicating which tasks use the CPU most frequently.

Such a task is invaluable for the improvement of task and system performance.

*DCB (Diagnostic Test of CDC Coupler)* Standard task only available on systems with a CDC coupler, always non-resident.

The original version of this task was coded in assembly language, since at the time of coding BCPL was not available. Since then, a BCPL version has replaced the original version.

The task is used in conjunction with a corresponding test program which executes in the connected CDC (Cyber 170/720 or 6500 or both). The CDC program is the master, and may be directed from the CDC console or via a command file. DCB, itself, merely responds to requests for particular tests, and will issue console messages in case of failure. Options include continuation after failure, continuation of other coupler test (if running) after failure on one coupler, or halt on error. The latter is of course infrequently used, since the Modcomp would normally be continuing to service other network links in parallel.

The task actually communicates with the CDC via calls to the standard CDC coupler handler. In order to guard against interference with calls from the SCB symbiont task, the normal network use of the CDC link has to be suspended. However, this is a small price to pay for an on-line diagnostic.

A full description of DCB is given in NPN 86.

*ED1, ED2, ED3, ED4, ED5 (Engineers Diagnostic Tests)* Standard tasks available on all systems, non-resident.

These assembly language tasks are on-line diagnostics used to check operation of a particular CERN data-link in a Modcomp. As with DCB, the relevant link must be disabled for normal network usage. However, the tasks then take absolute control of the link to be
tested, including resetting the interrupt address location.

Features of particular interest are the software disabling of the system protect feature, so that input/output instructions may be obeyed, and the use of pseudo-console switch register to simulate the physical console switches. By these means, any link in the network can be tested by an engineer situated at any network node.