ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI FISICA NUCLEARE - ISTITUTO NAZIONALE DI

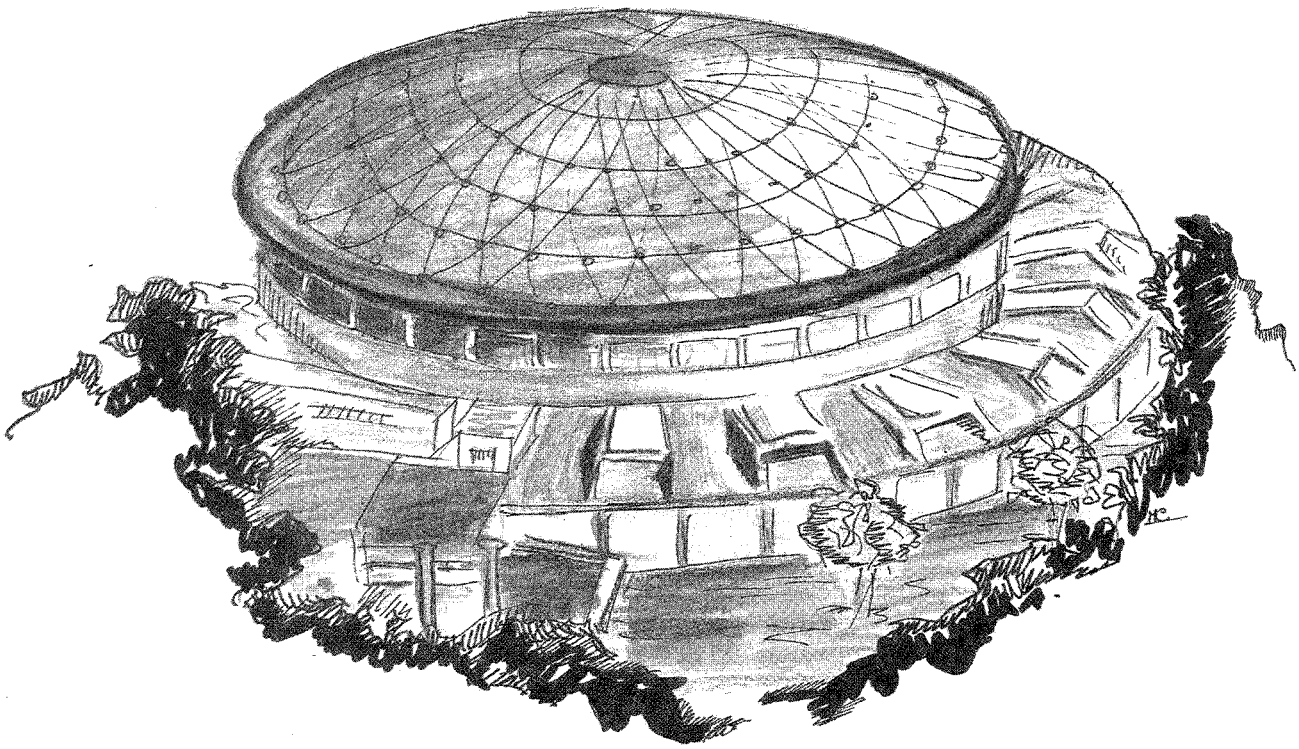
Laboratori Nazionali di Frascati

Sobmitted to IEEE Jr. of Selected Areas in Commun.

LNF-89/011(P)
1 Marzo 1989

M.L. Ferrer, E. Pace, G. Mirabelli, E. Valente, F. Fluckiger, G. Heiman:

GIFT: A MULTIPLE GATEWAY FOR FILE TRANSFER, ACCESS, AND MANAGEMENT



Servizio Documentazione
dei Laboratori Nazionali di Frascati
P.O. Box, 13 - 00044 Frascati (Italy)

INFN - Laboratori Nazionali di Frascati
Servizio Documentazione

LNF-89/011(P)
1 Marzo 1989

GIFT: A MULTIPLE GATEWAY FOR FILE TRANSFER, ACCESS, AND MANAGEMENT

M.L. Ferrer, E. Pace
INFN - Laboratori Nazionali di Frascati, P.O. Box 13, 00044 Frascati, Italy

G. Mirabelli, E. Valente
INFN - Sezione di Roma, Piazzale Aldo Moro 2, 00185 Roma, Italy

F. Fluckiger, G. Heiman
CERN - P.O. Box 1211, 23 Geneva (Switzerland)

ABSTRACT

A multi-protocol-converter allowing file access, transfer, and management and remote job entry between different network protocols is presented. The gateway architecture and the protocol conversion model, mediated by a File System, are described. It will be shown that this approach greatly reduces the complexity of the multiprotocol conversion problem. Some examples of the gateway implementation are given. The gateway, entirely designed and developed by an international collaboration, has been in real production since 1985.

1. - INTRODUCTION

At the beginning of the 80's, a number of networks were set up either between homogeneous computers using a communication protocol produced by manufacturers or between etherogeneous computers using a home-made protocol. At that time, it was already possible to put a computer on more than one network at the same time thereby allowing interconnections between different kinds of networks. Moreover, international standards for remote

terminal access were well defined, and the standard organizations were working in order to define electronic mail applications. On the contrary, the definition of standard documents concerning file transfer, access, and management and remote job entry applications was at its initial stage. Before the completion of the standard definitions, the scientific and academic community (in particular the High Energy Physics community) needed an interim solution to connect existing protocols for file transfer and access. For this purpose, the authors started to study the feasibility of a software gateway system.

After a first phase, the following general principles for file transfer and access for a gateway were stated:

1. No changes of the applicative programs on the end-nodes in order to reduce the software support needed and also to facilitate the end-users who require no special interface while accessing remote files through the gateway.

2. No "store and forward" feature on the gateway, but "on fly" conversion, allowing the user to control the whole transfer operation.

3. Software modifications on the gateway machine only at the application level, without changes of the involved protocols.

A first implementation of a such gateway had been already realized by the authors using a PDP† (1) (2) machine running the RSX operating system and interconnecting INFNET (the INFN network based on DECnet, the network protocol by DEC) and CERNET (the CERN network, based on a home-made network protocol). A second gateway allowing multiprotocol conversion, which is the argument of the present paper, has been realized by the authors, in the framework of an international collaboration, utilizing a MicroVAX II running the VMS operating system. This gateway, which is known as GIFT (General Internetwork File Transfer) (3), was developed starting from 1982 as a project of scientific and academic communities related to the CERN activities. It can actually perform translation between the following protocols: CBS-Blue Book, CERNET, DECNET, RHF and TCP/IP. New protocols can be

†DEC, RSX, VMS, PDP, MicroVAX, DECnet are trademarks of Digital Equipment Corporation

included in the near future, and in particular the ISO/OSI FTAM is one highly requested, in order to allow a smooth migration from the actual protocol towards the already defined standards.

This gateway will be described using the OSI relay model at the application level. It will be shown that a protocol conversion mediated by a File System allows multiprotocol conversions greatly reducing its complexity.

The OSI terminology (4)-(6) and in particular the concepts defined into the ISO 8571 documents (7)-(10), about the FTAM (File Transfer, Access, and Management) model will be used. This model defines a File Service and specifies a file protocol available within the application layer of the OSI Reference Model. It provides sufficient facilities to support file transfer, and establishes a framework for file access and file management. It also defines several concepts, particularly the two File Service Users, "Initiator" (FSUI) and "Responder" (FSUR), emphasing the asymmetry of the dialogue between them.

In Section II some concepts about the GIFT application relay will be presented, using these OSI concepts to describe the entities involved in any file transfer operation between computers. The multigateway implementation on a VAX/VMS operating system will be described in Sections III, IV and V. Section III describes how a multiprotocol converter can be realized by trapping in the native File System the calls to the file access routines and by dispatching them to the different slots or libraries which realize, through the FSUI routines, the access to a remote file on a different network. Section IV will present how such a system dispatcher has been realized, while Section V will describe the slot architecture and the protocol conversion problems giving some implementation examples.

2. - APPLICATION RELAY FOR FILE TRANSFER AND FILE ACCESS

Recent papers concerning formal description of the protocol converters (11)-(16) describe several approaches in order to realize a relay below the application level, the highest one defined

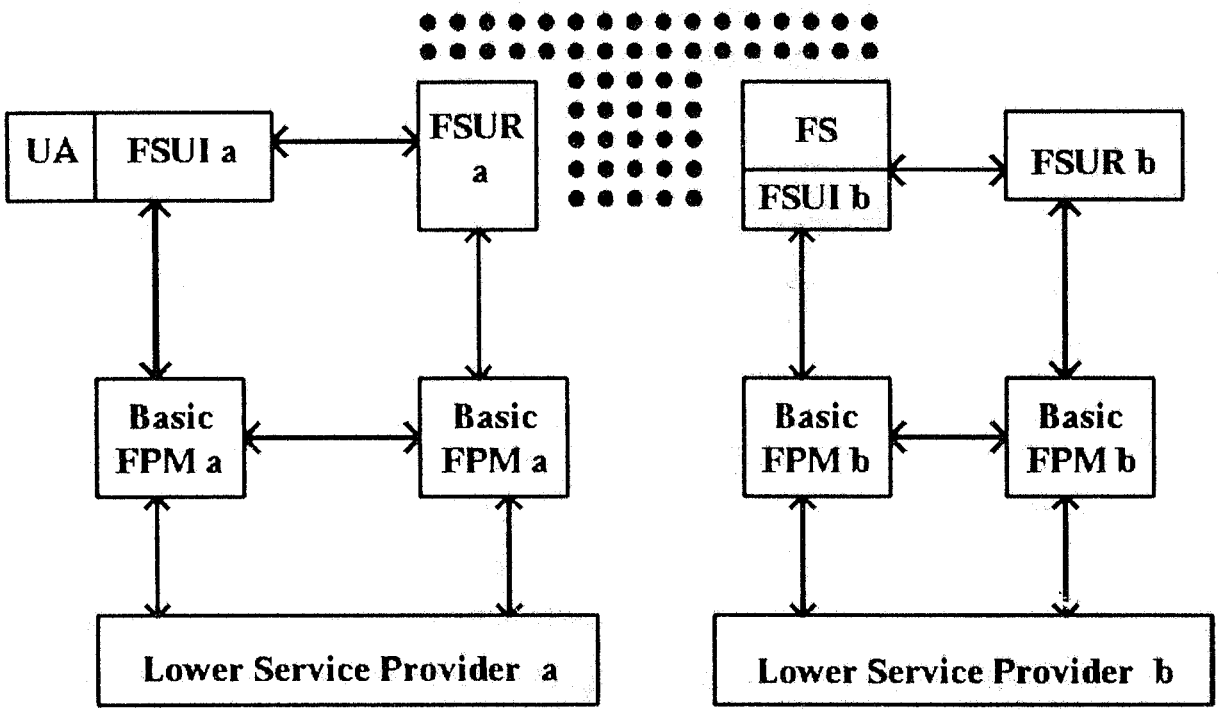


Figure 1. The application relay model mediated by a File System

in the ISO/OSI reference model. These approaches require therefore compatible utilities at the higher levels for all the machines accessing the gateway.

An alternative application level relay model, based on the concept of File System, is proposed by the authors. According to a usual scheme, in order to access a remote file located inside the same network, a user invokes a local FSUI utility, that activates a remote FSUR utility through a specific network protocol. FSUR accesses the file using File System (FS) primitives. An application relay for file transfer between networks **a** and **b** can be realized extending the previous scheme as shown in figure 1. The calls to the local File System primitives from the $FSUR_{\mathbf{a}}$ utility in the gateway, which is located in a common node of the two networks, are converted into a set of $FSUI_{\mathbf{b}}$ primitives.

An operating system supplying a unique set of primitives in order to access both local and remote files, if used in the gateway, would optimize and extend the capabilities of a such converter. The FSUR utility does not need to be modified and all the protocol conversion work concerns can be realized inside the local File System. Moreover, if this machine supports different FSUR utilities for different network protocols a multiple converter towards a specific

network protocol b is automaticaly realized. The complexity of this converter increases linearly with the number N of protocols involved. Conversely the realization of a multiprotocol converter defining a Protocol Converter Machine for each pair of networks involved presents a complexity increasing as N(N-1)/2.

3. - MULTIGATEWAY IMPLEMENTATION

The VMS operating system offers the user a unique set of services, Record Management Services (RMS), in order to process and manage both local and remote files and their contents (17). For this reason it has been chosen as the starting point to implement GIFT.

A user program can access files by calling a set of VMS system services. For each system service call, the operating system invokes the special instruction CHME #code (Change Mode to Executive). Such instruction, using the argument as the entry point of a dispatcher table, executes the relative RMS routine (18). The VMS system services implemented in GIFT are listed in Table 1.

For each service call, the user can provide a number of arguments to the RMS routines, using one or more control blocks. This capability assures a large flexibility when file and record attributes are defined. Four different kinds of control blocks are available: FAB, NAM, RAB, and XAB's.

RMS is set up by a common part, the Kernel, plus two different parts, for accessing local and remote files respectively. The Kernel allocates/deallocates dynamic memory needed for internal working areas, checks the input attributes in order to assure a consistence between them and the required services, checks the correctness of the sequence of the called services and provides the output arguments to the user program.

In order to access remote files through the DEC network, the RMS engages a dialogue with the remote FSUR, using the Data Access Protocol (DAP), through a set of network services called NT routines, briefly described in Appendix A, that realize all the required information

Table 1. RMS SERVICES implemented in GIFT

SYS$CLOSE	terminates file processing and disconnects all record streams
SYS$CONNECT	establishes a record stream by associating a RAB with an open file
SYS$CREATE	creates and opens a new file of any organization
SYS$DISCONNECT	terminates a record stream by disconnecting a RAB from an open file
SYS$DISPLAY	returns the attributes of an open file to the user program
SYS$ERASE	deletes a file and removes its directory entry
SYS$GET	retrieves a record from a file
SYS$OPEN	opens an existing file and initiates file processing
SYS$PARSE	parses a file specification
SYS$PUT	writes a new record to a file
SYS$READ	retrivies a specified number of bytes from a file, beginning on block boundaries
SYS$SEARCH	searches a directory, or possibly multiple directories, for a file name
SYS$WAIT	awaits the completion of an asynchronous record operation
SYS$WRITE	writes a specified number of bytes to a file, beginning on block boundary

management. The sequence and the contents of the messages depend on the requested remote files operation and on their attributes.

The GIFT application relay is set up by modifying the standard RMS, keeping its Kernel and the network service routines calling sequence with their input/output arguments. It retains also the connection between the VMS system services and the RMS routines.

The GIFT implementation is realized as a shareable library which contains the modified RMS modules and defines their entry points using the so called transfer vector mechanism. A separate routine (the GIFT dispatcher module) defines a dispatcher table that allows the connection between the VMS system services and these entry points, according to a new argument that identifies the network to be invoked. A different version of this shareable library exists for each network. A user program running in the relay machine, and in particular a $FSUR_{\mathbf{a}}$ utility, requiring access to the network **b**, will be linked including the GIFT dispatcher module and the relative shareable library defining the network **b** entry points.

In order to facilitate the relay management, it was decided to create a dispatcher and a different library (slot) for each network protocol (Fig. 2), instead of implementing a dispatcher inside an unique shareable library. Each slot consists of the Kernel and of a second part that is specific for each network protocol involved. The interface between the Kernel and the network is accomplished again by the NT routines. The original sequence of DAP protocol messages in each NT routine is converted, through a set of implementation dependent GIFT routines, in a sequence of calls to the specific network **b** File Service primitives, which are indicated in Fig. 2 as $FSUI_{\mathbf{b}}$. Any GIFT routine must return to the caller the outputs required, obtained from previous or actual network services invoked, or giving default values or unsupported error code. A brief description of the GIFT routines is given in Appendix B.

Another kind of solution was examined that eliminates the DAP protocol message mapping, converting any NT routine into one or more service routines of the network **b**. This solution would require a wide knowledge about the RMS internal areas and the NT routines structure for any slot construction. Moreover a RMS utility modified in this way could hardly conserve the generality of the original one.

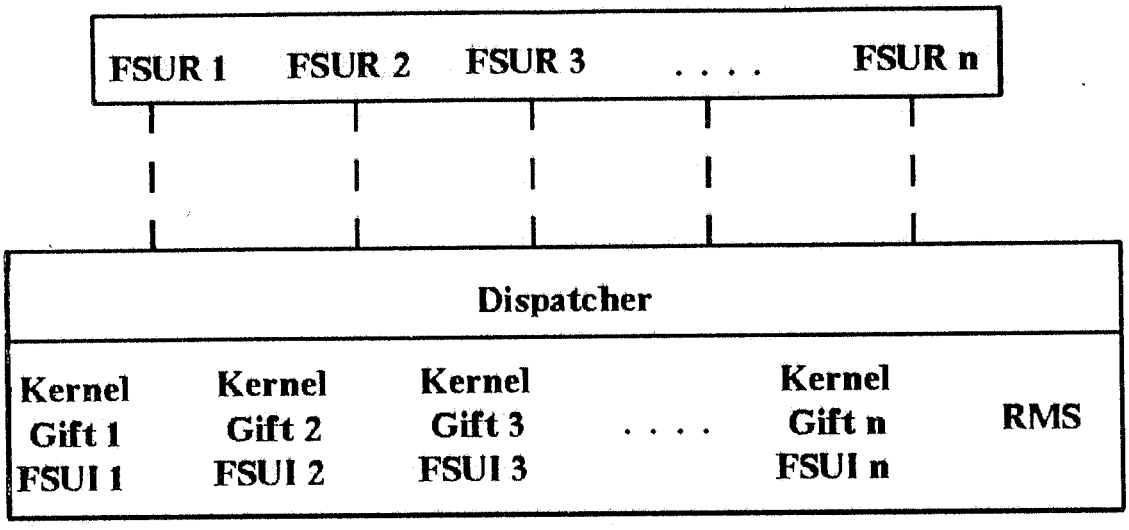


Figure 2. The GIFT multigateway system architecture

All the internal data areas defined in the original RMS remain unchanged except the Network Work Area (NWA). Each slot modifies this area according to the specific network requirements. The NWA area contains the DAP control block and gives storage for node and quoted string, scratch buffers used for file specification string parsing, record/block fragmentation, transmission and reception buffers, etc. The DAP area, that contains some fields as they were defined in the original RMS, realizes the communication between the NT routines and the particular network implementation.

4. - SYSTEM SERVICES DISPATCHER

Figure 3 shows an example of the GIFT dispatcher control flow that realizes the mapping between the VMS system services and the modified RMS routines for a given network. The left side of the figure presents the user program and the GIFT TRAP (dispatcher) routine.

Any program running on the GIFT machine and requiring access to another network for file operations, must introduce a call to the SYS$NETWORK routine, in order to move the network identifier in the FAB and/or RAB RMS control blocks. The access to the file is accomplished by calling the original system service primitives.

The GIFT TRAP module retrieves the network parameter from these control blocks and according to its value dispatches the VMS system service call to the appropriate NET$service module defined in the slot. In particular, the GIFT TRAP module also allows the jump to the original RMS primitives so that files on the relay machine are also accessible.

The right side in figure 3 shows the transfer vector definitions for two different slots that in any case are constructed starting from the Kernel (KER) routines. Unsupported services in the specific network are returned as a standard RMS error code.

At link time, the GIFT TRAP module and the slot transfer vector address must be included. This architecture allows the modification of the GIFT resident libraries without relinking of the user program.

The NET$service primitives, reproducing the original VMS structure, run in executive mode, calling the System Service SYS$CMEXEC. In this application only syncronous operations are allowed, for that reason the control returns to the user program only when the I/O operation has been completed. The dynamic memory management is defined separately from the original RMS thus allowing a large independence on subsequent operating system upgrades introduced by DEC.

5. - PROTOCOL CONVERSION

As previously said, each GIFT slot is constructed from the common Kernel and an implementation dependent part interfaced to the Kernel through a set of GIFT routines. The input/output arguments of these routines are defined as DAP area fields that reflect the RMS file structure. All the protocol conversion operations must be realized inside these GIFT routines, by mapping these fields, whenever possible into other equivalent fields of the slot specific network. Frequently, output arguments (e.g. creation date in an open operation) are not supplied by the network invoked. In this case, defaults could be given in order to cover up this lack of information. When the mapping is ambiguous, other information is required in order to solve this ambiguity. This information must be supplied by the Initiator and passed transparently to the GIFT routines. That is realized in GIFT using the so called foreign filename specification which is accepted by RMS, extended to include implementation dependent switches and

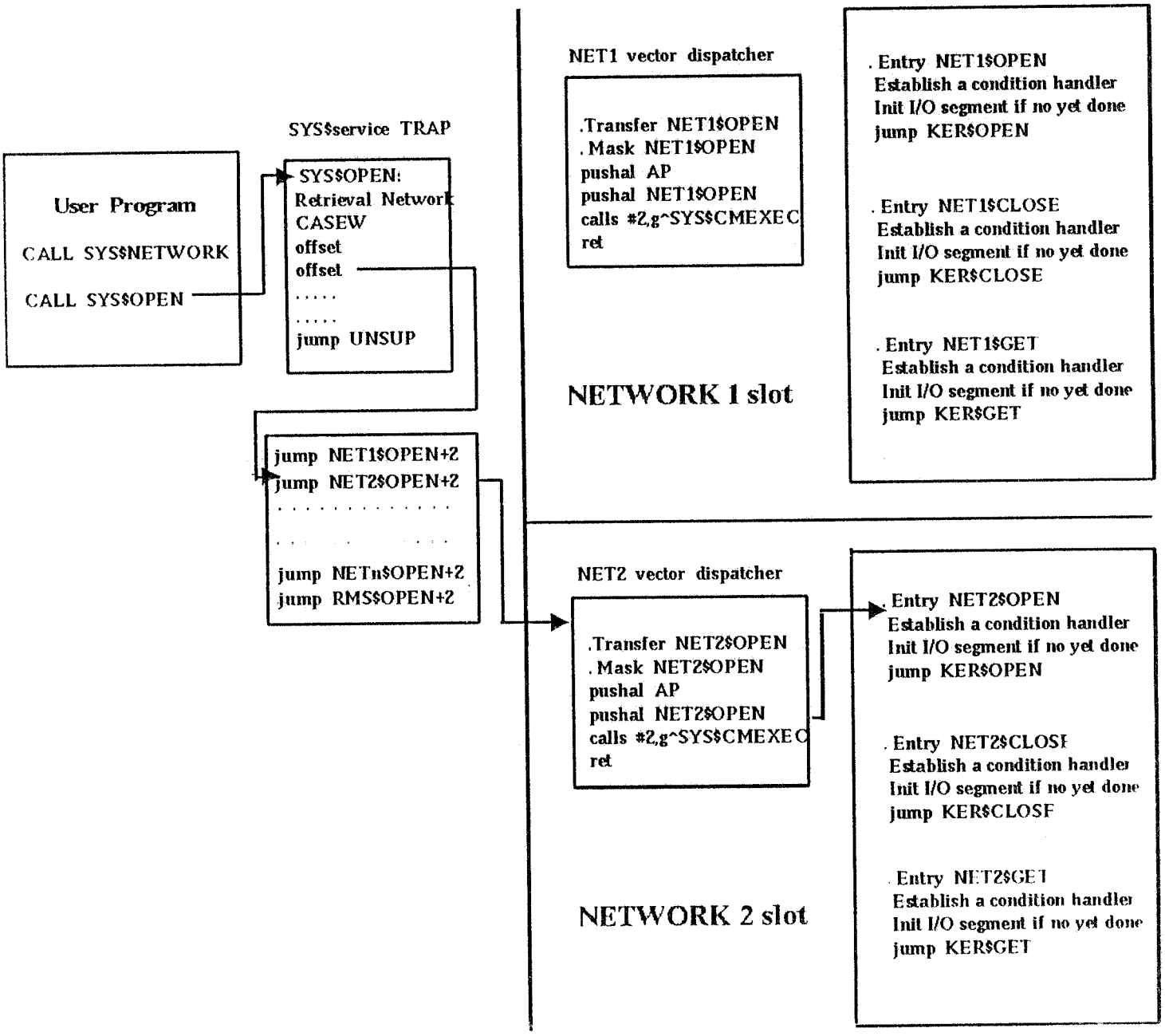


Figure 3. GIFT System Dispatcher

modifiers. These switches allow not only better specification of the file and record attributes, record format and access mode, but also the implementation of other remote network specific functions such as, for example in the CERNET slot, Job retrieval, Queue management (fetch, lists, delete entry), Job submission and Printing spooling, remarkably extending the capabilities of such a converter and adding new features to the original RMS, that is only able to realize Job submission or Printing spooling operations into generic queues.

Differences in the file record attributes require a record by record conversion inside the GIFT routines. Similarly, differences on block size or access mode (block or record access) require block packing/unpacking and record conversion.

In the following, the general features of GIFT will be clarified by discussing the example given in figure 4 which shows how the CERNET slot implements the operation of opening for retrieval a file resident on an IBM computer running the MVS operating system. The CERNET protocol was chosen for the example because it offers many differences with respect to the RMS, when representing file structure and attributes. The implementation dependent part is defined by the GIFT routines code and by the sequence of calls to the CERNET specific NFservices routines (19) which are briefly described in Table 2.

The SYS$OPEN service parses the filename string and sets pointers to the different components into the FWA area fields. In particular, the node specification including account information is set in the FWA$Q_NODE descriptor. The NT$ACCESS and NT$OPEN routines are called in order to create a link with the remote node and to open the specified file.

NT$ACCESS builds from FWA$Q_NODE a Network Connect Block (NCB), which contains the remote node identification, to be used as input to the GIFT$ACCESS routine. In the considered example (Fig. 4), the NCB is given by the string: *IBM"username password"::* The username field in the NCB is interpreted as containing the accounting information to be used when opening the link (group and user), and the password is interpreted as the IBM keyword. The GIFT$ACCESS routine finally creates a link via the NFSTRT routine with the

```
SYS$OPEN        (Input: filename=IBM"group.user key"::block:[name1]member)
    → NT$ACCESS
        → GIFT$ACCESS          (Input: NCB=IBM"group.user key"::)
            → NFINIT
            → NFSET    (GR=group,US=user,KY=key)
            → NFSTRT   (NODE=IBM)

    → NT$OPEN
        → GIFT$RECV_CNF        (Output: SYSCAP)
        → GIFT$GET_FILESPEC    (Input: FILE=block:[name1]member)
        → GIFT$OPEN
            (Input:    DataType=block,DS=name1,MB=member,
            Output:    RL)
                → NFSET    (DS=name1,MB=member)
                → NFOPEN   (DataType=block,RL)
        → GIFT$RECV_ATT
            (Input:    RL,DataType=block
            Output:    DEV=disk_device,ORG=sequential,BLS=default,
                       ALQ1=default,DEQ1=default,FSZ=0,EBK=default,
                       FFB=0,LRL=0,FOP1=0,RFM=fixed,
                       RAT=none,MRS obtained from RL)
        → GIFT$RECV_TIM
            (Output:   CDT=default,RDT=default,EDT=default,
                       RVN=default,BDT=default)
        → GIFT$RECV_PRO
            (Output:   OWNER=default,PROSYS=default,
                       PROOWN=default,PROGRP=default,
                       PROWLD=default)
        → GIFT$RECV_NAM
            (Output:   NAMESPEC=block:[name1]member)
```

Fig. 4. Open for file retrieval on a CERNET IBM

Table 2. CERNET file services routines

NFCLOS	close a file. Options allow the user to specify whether the file is to be kept or to be deleted
NFINIT	first call that the user should make before starting file access operations.
NFGET	reads the next record of a sequential or partitioned file into the user's record buffer.
NFOPEN	to open a file. An NFOPEN call must normally precede all other operations on a file.
NFSET	generalized mechanism for specifying various options and parameters to the user interface software.
NFSTOP	to close the logical link between the subscriber task and the host file manager.
NFSTRT	to specify the remote host file system. It is also responsible for creating the connection with the CERNET responder, giving the necessary accounting information.

IBM CERNET node, previously setting in the CERNET work buffer, the group, user, and keyword parameters via NFSET routines. The status code, converted from CERNET code into a RMS standard code, is reported back to the SYS$OPEN routine.

The NT$OPEN routine calls the GIFT$RECV_CNF routine, that simulates a partner using a File System with the capabilities listed in Table 3. These capabilities define which kind of information will be exchanged between the NT and the GIFT routines. This general feature of GIFT allows a large flexibility in building any GIFT slot.

Calling the GIFT$GET_FILESPEC routine, the filename string is converted into the new network protocol syntax. Both the RMS native and foreign filename specification can be used through GIFT. The native filename specification (e.g.*device:[directory]filename.type;version*) allows the use of wildcards in order to produce multiple file transfers. The foreign filename specification does not allow multiple file transfers but specifiers can be appended allowing a better mapping between the differences in functions and attributes used in the two networks involved in the conversion.

In any case, the GIFT$GET_FILESPEC routine maps each filename field into the IBM filename specification fields. If native specification is used, the routine extracts the different fields from the FWA area because the SYS$OPEN sets this area after parsing the filename string. For example, the following filename specification, *[name1.name2.name3]*.type*
is interpreted in the MVS operating system as requesting access to all the members of the *name1.name2.name3.type* partitioned dataset. If foreign file specification is used, the routine receives transparently the initial string and parses it to obtain both the IBM filename and the specifiers indicating special IBM functions or file attributes. For example, the following filename specification, *"name1.name2.name3.type#member1/UNDEFINED/RL:3600"*
is interpreted as requesting access to a given partitioned dataset member that is stored as fixed record length containing 3600 bytes.

The CERNET File Access Protocol (FAP) foresees a number of different DataTypes to be specified by the user, allowing various kinds of files to be transported and interpreted amongst different machines. This DataType normally has to be specified by the user, and it is essential to repeat the same code when storing and then retrieveing a file, otherwise the file structure may be completely lost.

Moreover, three different categories of specifiers exist in the GIFT CERNET slot:

Table 3. IBM-CERNET System capabilities

Partner node supports:

- Allocation of space at file creation
- Sequential file organization
- Sequential file access (file transfer mode)
- Append records to end-of-file
- Command file submission/execution
- Display of file attributes on request
- Directory list operation
- Date and Time XAB message
- File Protection XAB messsage
- Spool file on close FOP option
- Execute command file on close FOP option
- Delete file on close FOP option
- Wildcard operations (excluding directory)
- Name message

---

* The MODIFIER, which essentially maps the CERNET DataType concept.
* The SERVICE, to instruct the GIFT software to initiate a Remote Job Entry operation (Job submission, Job control and Job retrieval) or Printing Spooling operation.

* The SWITCH, to further specify a request. This is especially useful for services, which allow a wide spectrum of special options to be specified.

A few statements can be added to further clarify the relation and interaction of modifiers, switches and services specifications.

- A modifier implies a number of default switches. For this reason, to open a file to read, the specification of the modifier is normally sufficient. In order to extend the use of the RMS native file specifications, the required modifier can be put inside the device field, as in the following example

  *IBM"user.group key"::UNDEF:[name1.name2]**

- If no modifier is specified at all and the file is not in default format, a retry strategy is implemented in the CERNET slot based on the error response from the IBM, that could possibly be an indication of the record format and file structure.
- In general the use of switches should be more common with a Service specification that almost always requires additional information to be specified, while a genuine file transfer request is in general perfectly defined by the modifier alone.

Following the example in Figure 4, where the filename native specification without the use of wildcard was given, the NT$OPEN routine calls a set of GIFT routines, dependent from the remote file system capabilities, in order to exchange the informations required. The GIFT$OPEN routine maps the file attributes and opens the specified file on IBM, by calling the NFSET routine to set the Dataset (DS) and the Member (MB) names into the CERNET work buffer, and the NFOPEN to finally open the file. The NFOPEN routine uses the DataType argument in order to define how the IBM file must be opened. The Record Length (RL) is returned as output information. This value will be used by the GIFT$RECV_ATT routine in order to calculate the RMS Maximum Record Size (MRS).

The GIFT$RECV_ATT returns the device and file attributes. In the actual implementation, the device characteristics returned corresponds to a disk device. Concerning the file attributes, three different categories must be defined: file organization, file dimension and record structure. In the actual implementation, the only file organization supported is the sequential one. No information is returned by the CERNET protocol about the file dimension

when a partitioned dataset is accessed. For this reason, the relative arguments are set to zero indicating an unknown value. The record format is defined as a function of the DataType argument and in this example assumes the value corresponding to a fixed record format and no record attributes are defined in the RAT argument.

The GIFT$RECV_TIM and GIFT$RECV_PRO give default time and protection attributes. Finally the GIFT$RECV_NAM maps the filename in order to construct the resulting filename string that will be returned to the caller in a native or foreign specification.

Concluding this Section, it is important to note how the described approach in implementing the protocol conversion inside the CERNET slot resolves both the large differences between file structure and file attribute definitions, used by the RMS and the IBM file system respectively, and moreover the limitations on the CERNET protocol when transporting these file attributes.

6. - CONCLUSIONS

A multiprotocol conversion model and its implementation for file transfer, access, and management, realized on a VAX/VMS computer has been presented. The system architecture implementation has been discussed: it is set up by a central dispatcher with a slot for each protocol involved. This model has allowed the easy inclusion of network protocols that are largely used in the European scientific and academic community, always retaining the functions implemented by each protocol separately. Examing the first ISODE FTAM implementation, which only allows file transfer for the time being, but new profiles will be available in the next future, the authors are convinced that it is perfectly possible to include an FTAM slot inside GIFT, allowing a smooth migration to the ISO/OSI standard within the European scientific and academic community.

ACKNOWLEDGEMENT

The authors wish to thank the DEC Italy for providing the documentation and the RMS sources. Moreover, they wish to thank W. Blake from the JANET Network Team and S.C. Weston from the Rutherford Appleton Laboratory, for the BlueBook slot work, and finally T.

Moene, W. Poppe, P. Wielinga and W. van der Scheun from the SARA community for the TCP/IP and RHF slots work.

APPENDIX A

NETWORK SERVICES IMPLEMENTED BY THE NT MODULES

* **NWA setup and release :**
  - NT$NWA_INIT: Allocates space for a NWA control block and then initializes selected fields.
  - NT$NWA_FREE: Deallocates the NWA control block after all QIO have been processed.
* **Network access/deaccess :**
  - NT$ACCESS: Creates a logical link between this process and the FSUR.
  - NT$ASSIGN: Assigns a channel to the network device.
  - NT$DEACCESS: Destroys a logical link.
  - NT$RET_DEV_CHAR: Returns the true device characteristic information to the user's FAB.
* **Network search files :**
  - NT$SEARCH: communicates with the FSUR at the remote node to obtain a list of directory entries that match a given (wildcard) file specification.
* **Network open file :**
  - NT$OPEN: communicates with the FSUR at the remote node to open the specified sequential, relative, or indexed file.
* **Network create file :**
  - NT$CREATE: communicates with the FSUR at the remote node to create the specified sequential, relative, or indexed file.
* **Network connect stream :**
  - NT$CONNECT: communicates with the FSUR at the remote node to establish a record stream for the specified file.

* **Network block I/O** : communicates with the FSUR at the remote node to perform read and write block I/O operations.
  - NT$READ: to read the specified blocks.
  - NT$WRITE: to write the specified blocks.
* **Network get/put record** : communicates with the FSUR at the remote node to perform get and put records operations.
  - NT$GET: to get (retrieve) the specified record of a sequential, relative, or indexed file. This routine supports both DAP file transfer and record transfer modes.
  - NT$PUT: to put (store) the specified record of a sequential, relative, or indexed file. This routine supports both DAP file transfer and record transfer modes.
* **Network disconnect stream** :
  - NT$DISCONNECT: communicates with the FSUR at the remote node to disconnect the specified record stream.
* **Network close file** :
  - NT$CLOSE: communicates with the FSUR at the remote node to close the specified file. Optionally, a file may be deleted, executed, printed, submitted (executed then deleted), or spooled (printed then deleted) on close at the remote node.
* **Network erase file** :
  - NT$ERASE: communicates with the FSUR at the remote node to erase (delete) the specified file.

APPENDIX B

GIFT ROUTINES

* GIFT$ACCESS: creates a logical link.
* GIFT$CLOSE: closes the specified file and process selected FOP options.
  - Implicit Inputs:

    DAP$L_FOP1, file options field
* GIFT$CREATE: creates the specified sequential, relative, or indexed file.

- Implicit Inputs:

  DAP$B_ORG, file organization

  DAP$B_RFM, record format

  DAP$W_MRS, max record size allowable

  DAP$L_ALQ1, allocation quantity field

  DAP$B_FSZ, record header size

  DAP$W_DEQ1, default extension quantity field

  DAP$L_FOP1, file options field

  DAP$W_LRL, longest record's length

* GIFT$DEACCESS: destroys a logical link.

* GIFT$ERASE: erases (deletes) the specified file.

* GIFT$GET: gets (retrieves) the specified record of a sequential, relative, or indexed file. This routine supports both file transfer and record transfer modes.

  - Implicit Inputs:

    DAP$B_RAC, record access mode

    DAP$L_VBN, next record pointer (relative)

* GIFT$GET_FILESPEC: builds a filespec (less primary node name) from its constituent parts in FWA area, set by the SYS$PARSE routine.

* GIFT$OPEN: opens the specified sequential, relative, or indexed file.

  - Implicit Inputs:

    DAP$B_ORG, file organization

    DAP$B_RFM, record format

    DAP$B_RAT, record attributes

    DAP$W_MRS, max record size allowable

    DAP$B_FSZ, record header size

    DAP$W_DEQ1, default extension quantity field

    DAP$L_FOP1, file options field

* GIFT$PUT: puts (stores) the specified record of a sequential, relative, or indexed file. This routine supports both file transfer and record transfer modes.

- Implicit Inputs:

  DAP$B_RAC, record access mode

  DAP$L_ROP, record options field

  DAP$L_RECNUM1, record number field

* GIFT$READ: reads the specified blocks and copies them into the BDB buffer.

  - Implicit Inputs:

    DAP$B_RAC, record access mode

    DAP$L_VBN, next record pointer (relative)

* GIFT$RECV_ATT: simulates attributes message received.

  - Implicit Outputs:

    DAP$L_DEV, device characteristics field

    DAP$B_ORG, file organization

    DAP$W_BLS, block size field

    DAP$L_ALQ1, allocation quantity field

    DAP$W_DEQ1, default extension quantity field

    DAP$B_FSZ, record header size

    DAP$L_EBK, end-of-file block number field

    DAP$W_FFB, first free byte in eof block

    DAP$W_LRL, longest record's length

    DAP$L_FOP1, file options field

    DAP$B_RFM, record format

    DAP$B_RAT, record attributes

    DAP$W_MRS, max record size allowable

* GIFT$RECV_CNF: simulates a received configuration message.

  - Implicit Outputs:

    DAP$Q_SYSCAP system capabilities field

* GIFT$RECV_NAM: simulates a received name message.

  - Implicit Inputs:

    DAP$B_ACCFUNC, access function field set as

DAP$K_DIR_LIST, if doing directory-list

DAP$K_OPEN, if doing open

DAP$K_CREATE, if doing create

- Implicit Outputs:

DAP$Q_NAMESPEC descriptor pointing to the name field

* GIFT$RECV_PRO: simulates protection message received.

- Implicit Outputs:

DAP$Q_OWNER, descriptor pointing to the file owner field

DAP$W_PROSYS, system protection field

DAP$W_PROOWN, owner protection field

DAP$W_PROGRP, group protection field

DAP$W_PROWLD world protection field

* GIFT$RECV_STS: simulates status message received. This message is received from partner if not in file transfer mode and returns record file address of the first block accessed.

- Implicit Outputs:

DAP$W_RFA, record file address

* GIFT$RECV_TIM: simulates a time message received.

- Implicit Outputs:

DAP$Q_CDT, creation date and time field

DAP$Q_RDT, revision date and time field

DAP$Q_EDT, expiration date and time field

DAP$W_RVN, revision number field

DAP$Q_BDT, backup date and time field

* GIFT$SEARCH: obtains a list of directory entries that match a given (wildcard) file specification.

- Implicit Inputs:

DAP$B_FAC, file access

* GIFT$SEND_CTL: Simulates a Control message sent to partner.

- Implicit Inputs:

DAP$B_CTLFUNC, control function field set as

DAP$K_PUT_WRITE, if doing Put/Write

DAP$K_GET_READ, if doing Get/Read

DAP$K_CONNECT, if doing Stream Connect

* GIFT$WRITE: writes the specified blocks from the BDB buffer.

  - Implicit Inputs:

    DAP$B_RAC, record access mode

    DAP$L_VBN, next record pointer (relative)

REFERENCES

(1) M. L. Ferrer, G. Mirabelli, C. Stanescu and E. Valente,"Conversione di Protocollo di livello superiore a quattro tra reti DECnet e CERNET", *Atti III DECUS Italia*, 1982, p.151.

(2) M. L. Ferrer, G. Mirabelli and E. Valente,"Gateway INFNET - CERNET performance", *LNF-83/3(R)*, 1983.

(3) M. L. Ferrer, F. Fluckiger, G. Heiman, G. Mirabelli and E. Valente, "GIFT: an HEP project for file transfer", *Computing in High Energy Physics*, L.O. Hertzberger and W. Hoogland Ed., Elsevier Science Publishers B.V., North Holland, 1986, pp. 189-192.

(4) "Information processing systems - OSI - Basic reference model", ISO 7498, 1986.

(5) "Information processing systems - OSI - Service conventions", ISO/TR 8509.

(6) C. A. Vissiers and L. Logrippo,"The importance of the service concept in the design of data communication protocols", *Protocol Specification, Testing and Verification, V*, M. Diaz Ed., Elsevier Sciences Publishers B.V., North Holland, IFIP, 1986, pp. 3-17.

(7) "Information processing systems - OSI - File transfer, access and management - Part 1. Introduction", ISO 8571-1, 1987(E).

(8) "Information processing systems - OSI - File transfer, access and management - Part 2. Vitual filestore", ISO 8571-2, 1987(E).

(9) "Information processing systems - OSI - File transfer, access and management - Part 3. File service definition", ISO 8571-3, 1987(E).

(10) "Information processing systems - OSI - File transfer, access and management - Part 4. File protocol specification", ISO 8571-4, 1987(E).

(11) V. G. Cerf and P. T. Kirstein,"Issue in packet-network interconnection ", *Proceeding of IEEE*, Vol. 66, November 1978.

(12) H. X. Zeng, and D. Rayner,"Gateway testing techniques", *Protocol Specification, Testing and Verification,IV*, Y. Yemini, R. Strom and S. Yemini, Ed., Elsevier Sciences Publishers B.V., North Holland, IFIP, 1985, pp. 637-655.

(13) P. E. Jr. Green,"Protocol conversion", *IEEE Trans. Commun.*, Vol. COM-34, No. 3, March 1986, pp. 257-268.

(14) I. Groenbaek,"Conversion between the TCP and ISO transport protocols as a method of achieving interoperability between data communications systems", *IEEE Journ. on Select. Areas Commun.*, Vol. SAC-4, pp. 288-296, March 1986.

(15) K. Okumura, "A formal protocol conversion method", *Proc. of the ACM SIGCOM '86 Symposium*, pp 30-37, August 1986.

(16) Y. Ohara, S. Yoshitake and T. Kawaoka,"Protocol conversion method for heterogeneous systems interconnection in multi-profile environment", *Protocol Specification, Testing, and Verification, VII*, H. Rudin and C.H. West Ed., Elsevier Sciences Publishers B.V., North Holland, IFIP, 1987, pp. 405-418.

(17) "VAX Record Management Services Reference Manual, version 4.0", September 1984

(18) L.J.Kenah, S.F.Bate,"VAX/VMS Internals and Data Structures", Digital Press, 1984

(19) J.Blake et al, "The File Access Protocol" Network Project Note 04, CERN 1978