Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-92-003

A Distributed Architecture to Provide Uniform Access to Pre-existing, Independent, Heterogeneous Information Systems

F Naldi, K G Jeffery, G Bordogna, J O Lay and I Vannini Parenti

January 1992

A DISTRIBUTED ARCHITECTURE TO PROVIDE UNIFORM ACCESS TO PRE-EXISTING, INDEPENDENT, HETEROGENEOUS INFORMATION SYSTEMS

F. Naldi(*), K.G. Jeffery(+), G. Bordogna(*), J.O. Tay(+), I. Vannini Parenti(*)

(*) Consiglio Nazionale delle Ricerche, via Ampere 56, 20133 Milano, Italy
(+) Rutherford Appleton Lab., Chilton Didcot Oxon, OX11 0QX, UK

The mailing address is:
Fulvio Naldi
IFCTR-SIAM, CNR,
Via Ampere 56, 20131 Milano, Italy
tel. +39(2)2365138, fax +39(2)2663030
E-mail: EXIRPSW@IMISIAM.BITNET

ABSTRACT

We present an architecture to support transparent uniform access to pre-existing independent and heterogeneous Information Retrieval Systems (IRS) and/or Data Base Management Systems (DBMS) distributed on national and international networks.
The architecture is based on a high level protocol for data exchanging that has been implemented for an application concerning access to information on government-funded research projects in several countries.

1. INTRODUCTION

The development of digital networks and the proliferation of many different Information Retrieval Systems (IRS) and Data Base Management Systems (DBMS) used locally to handle data on digital media, allow to manage information in a new way. The problem of allowing a user to access "information anywhere", via computer networks, has generally been approached by building a completely new software system to store, search, manipulate digital data [1, 2, 3, 4].
Our point of view is that efforts should be addressed to exploiting investments already made in locally operating IRSs and DBMSs. We have built a distributed architecture to support transparent uniform data exchanging between pre-existing independent and heterogeneous IRSs and DBMSs.
Data exchange among IRSs and/or DBMSs distributed on a network requires an agreed high level communication protocol to manage query procedures, updating procedures, privacy handling, the monitoring of data flow and user satisfaction, and inter-task communication. In fact, standard low level telecommunication protocols perform error correction and ensure reliability of data transmission, but do not provide higher level functions, much more dependent on data structure and on organizational aspects, for a complete management of the application.
The design of a superstructure for data interchanging among IRSs and/or DBMSs can be regarded as the design of a DIRS.
Macleod in [4], lists several reasons for requiring the development of a Distributed IRS (DIRS) rather than a Distributed DBMS and indicates the three basic facilities that implementation of a whole DIRS must provide.
First, there must be some mechanism that permits operations residing at one node in the DIRS architecture to be invoked remotely from another (query functions, updating, etc.).

Second, nodes must be able to communicate with each other.
Third, each node must know where data and functions reside in the network.
The second point is satisfied by assuming that nodes are linked by interconnected networks supporting low level communication protocols. To satisfy the requirents expressed in the first and third points, a high level communication protocol which uses information contained in control tables to manage data flow has been designed.
In this paper we describe the architecture of a DIRS derived from a previous one designed within the IDEAS project [5]; particular emphasis is placed on the description of the high level communication protocol which provides transparent uniform access to the distributed data managed by local IRSs or DBMSs. Although the high level communication protocol has been developed for a specific application, it is general enough to be applied to many other cases in which data are organized into two parts:

1. several independent data bases;

2. replicable copies of a catalog of selected attributes (fields) of the overall data bases, managed by local IRSs and DBMSs; the catalog content serves as the index for the DIRS [6, 7].

Separate copies of the indexes allow the simultaneous use of different access strategies. This protocol uses standard electronic mail (E-mail) facilities, supplied by the interconnected computer communication networks, for lower level communication functions; it is thus essentially independent of the type of services supplied by the network and does not require any changes in pre-existing DBMSs and IRSs structures and management procedures, but allows full control of information flow and data access.
An application based on the use of the developed prototype system is shown in section 5.

2. THE APPLICATION ENVIRONMENT

There are many examples of systems which require the linking of several heterogeneous DBMSs. The systems may be within the same or different organizations, in one or many countries. Such linked systems are generally described by the term multibase [8], after the project of that name [9].
A specific example is organizations' need to exchange information on current scientific research for the benefit of researchers who wish to collaborate and of science managers who wish to optimize the allocation of scarce resources. EXIRPTS (EXchange of Information on Research ProjecTS) is a collaborative effort among national research funding organizations to provide easy access to information on current research projects. The eight countries participating (Canada, France, Germany, Italy, Japan, Sweden, United Kingdom, USA) have agreed to develop a system based on the concept of a logical superstructure which allows homogeneous access to pre-existing and different national systems without changing their domestic structure, but providing for development and expansion to meet local user needs.

In this context, the DIRS architecture has been designed as the evolution of the distributed database system built within the IDEAS project [5]. However, in the EXIRPTS context, while the distributed data base organization and content definitions remain more or less intact, the high level communication protocol has been improved to meet the technical and political requirements of a wider group of partecipating countries and organizations and allow for further expansion and development.

3. GENERAL REQUIREMENTS

Users from organizations in different countries may need to access data from multiple different systems operated by other organizations in other countries. Each organization may have one or more systems, and each country may have one or more organizations. Conversely, each organization may have branches in one or more countries, and those branches may all use one centralised system.
A DIRS architecture whose primary characteristic is to make data held in the local, specific database and information retrieval systems of each particular organization available, to a wide

community of users - utilizing different DBMSs and IRSs - must satisfy certain well defined requisites:

1. **generality:** the architecture must be designed to support any application with the above characteristics;
2. **flexibility:** the architecture must allow a complex topology with any number of nodes of different types clustered together in any combination;
3. **expandability:** as additional organizations join the community and additional systems used by those organizations come online, the architecture must allow expansion of the topology, data structure, content and volume;
4. **independence:** the architecture must allow the use of any type of DBMS, supported by any hardware and operating system environment and accessed over any generally available networking system;
5. **distributivity:** the architecture must support a topology of loosely-linked, semi-independent nodes distributed geographically or logically (for parallelism and performance);
6. **transparency:** the architecture must allow the user to access remote data without knowing where these actually reside; it must also allow the user to query the catalog at his own site using the language of the local IRS or DBMS;
7. **decentralization:** the architecture must not have single nodes that supervise all the system or store all of the system resources;
8. **efficiency:** the architecture must provide cost-effective and sufficiently rapid access to the data;
9. **security:** the architecture must protect the data while they transit between nodes (which are expected to have their own local data security management) and must ensure that all messages are authorized;
10. **privacy:** the architecture must allow access only to authorized users, and must ensure that updating takes place in a controlled manner;
11. **currency:** the architecture must support all attempts to keep the data in the various systems up-to-date, and as consistent as possible.

The requisites specified in points 8 through 11 can be met by the high level communication protocol supporting our DIRS architecture.

4. DISTRIBUTED SYSTEM ARCHITECTURE

4.1 Overview

A novel architecture has been designed and developed to meet the requisites described above: it can be represented by a superstructure (see figure 1) that links pre-existing, locally-used IRSs and DBMSs through pre-existing interconnected computer communication networks. This internet environment can include both Wide Area Networks (WAN) spanning thousands of miles and Local Area Networks (LAN) encompassing a mile or two. The architecture exploits the existing investment in those systems, but allows their use in a wider context.
The data managed by each local IRS or DBMS in the superstructure, are partitioned into data objects, i.e. unstructured textual and multimedia documents or structured data records. A document is defined as the minimum amount of data that can be retrieved in response to a query by a IRS. A structured data record is defined as the minimum set of data retrieved by a DBMS query.

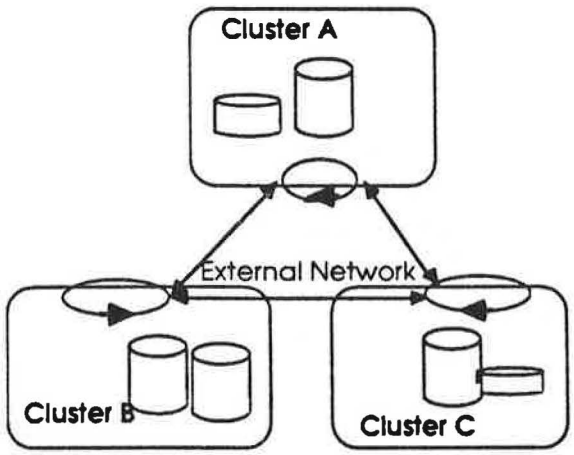

Fig. 1: High level representation of the Distributed System

The system may be thought as composed of clusters. Each cluster is populated with nodes, i.e., physical network connection points, which support actuators. An actuator consists in a set of programs which allow the processing of messages received from each node according to high level protocol specifications (see figure 2).

There are three possible types of nodes at each cluster:

a postmaster-node: which controls the routing of queries, responses and updates from its cluster to the rest of the network and from the rest of the network to nodes in its cluster;

data-nodes: hosting pre-existing IRSs and DBMSs, set up by local organizations for the acquisition, storage and retrieval of data objects. Data nodes have actuators which communicate in both directions with the postmaster-node of the cluster to which they belong.

catalog-nodes: they host identical copies of the catalog, which is replicated at all catalog-nodes and represents succinctly the data objects present at all data-nodes in the system. In DIRS architecture, the catalog hosts the indexes of all the data objects present in the system. In this sense the catalog provides the means for data object retrieval. In fact, user access to the DIRS can be obtained only through a catalog-node. Furthermore, because these indexes are replicated at each catalog-node, the data objects can be uniformly accessed at each catalog-node site. The catalog copies are maintained by the managers of each catalog by applying suitable updating procedures provided by the high level communication protocol. Catalog-nodes communicate in both directions with the postmaster-node of the cluster to which they belong.

The architecture of the DIRS is now clearer: it consists of clusters (groups of data-nodes and catalog-nodes) which share resources such as networks, computer systems, etc., and have one and only one postmaster-node. The nodes in a cluster communicate with each other by using the E-mail service supplied by the internal network, which can be either LAN or WAN. The postmaster-node of each cluster acts as a server to the internal network and is directly linked to all the other postmaster-nodes through an external network. The external networks linking together two postmaster-nodes may be different so long as they provide the same low-level file transfer facilities and gateway services.
The system is distributed at different levels:

- the data objects available for retrieval are spread all over the data-nodes;
- as the data object indexes are replicated at each catalog-node, the access points to data objects are distributed in each cluster. Each catalog-node covers a specific geographic user area, which is local to those users. An end-user may query the system only with the authorization of his local catalog-node manager.

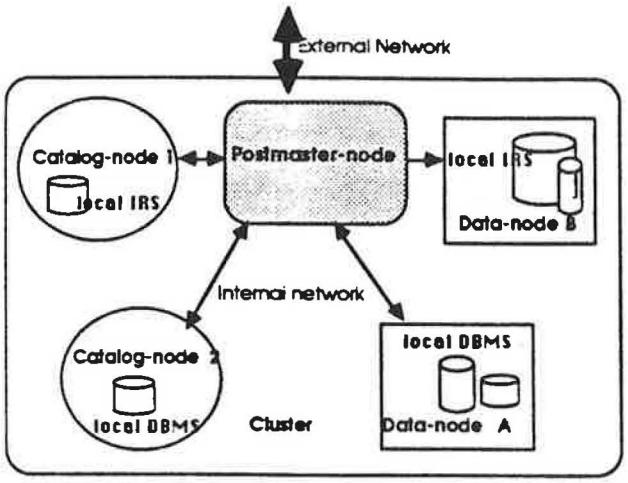

Fig. 2. Representation of the Topology of a General Cluster

4.2 Databases and Catalog organization

As mentioned in section 4.1, the data at each data-node consist of data objects; each data object must have a unique primary and identifying key (OID - Object Identifier) which identifies both the data object and the data-node where it is stored. Data objects are usually structured in fields identified by attributes. An Object type is defined by a specific set of attributes. A particular data object (instance) will have particular values of those attributes.
The data objects at different data-node databases may differ in object type. A minimum set of information (indexes) sufficient to describe the data object must be available to the catalog. The most reduced set of information is the OID, but for a system covering a single domain of interest the set of information can, at the other extreme, be the same as the OID plus all the attribute-values of the object. The optimal set of information (attributes) for catalog entries is a balance between the maximal set (to expedite retrieval) and the minimal set (to minimise data duplication).
In our model data may be collected autonomously at different data-nodes. Each data-node is responsible for providing the catalog-nodes with updated objects' indexes and for the correctness and consistency of its data objects.

4.3 Basic Functions of the high level Communication Protocol

Data and information can be exchanged between pairs of nodes by messages sent through the network and following the specifications of the high level communication protocol. The messages consist of a header, the data and a trailer. Within the header/trailer records of the message is all the information needed (by the receiving actuators) to recognize the kind of message and perform the proper action. In this way it is possible to avoid using information supplied by the low level network services (for example, for detecting the originator of the message), which would generate dependencies between the high level protocol and the kind of network facility.

The messages defined in the high level communication protocol may be classified in six types, based on originating node (ORIGNODE), destination node (DESTNODE) and function (FUNCTION), as shown in table 1.

ORIGNODE	DESTNODE	FUNCTION
catalog	data	query databases at data-nodes for objects specified through interactive querying of a catalog
data	catalog	response to above query; consists of objects from one database at a data-node
data	catalog	update catalog entry (insert or delete) upon update of databases at data-node
catalog	data	acknowledge to database at data-node the successful completion of updates or deletes to catalog
(manager)	'anynode'	special message; for example control command; the manager is a person and 'anynode' is any node
'anynode'	(manager)	special message; for example, response to above message

Table 1: Basic Functions of the high level communication protocol

The postmaster-nodes do not appear in the ORIGNODE or DESTNODE columns as they are always in an intermediate position along the communication path of any message.

5. APPLICATION OF DIRS TO MULTINATIONAL DATA BASES

The protocol designed on this architecture has been applied in the EXIRPTS project (see section 2) for the exchange of information on government-funded research projects in several countries. In this case, all the data objects are from the same information domain (research projects) and the catalog can consequently be enhanced to contain together with the OID, a common subset of attributes which are used to expedite retrieval of the required data objects. The agreed common subset of information for each data object (indexes) is extracted from each DB-node (organizational DBs) and these records are merged to build the catalog (TCC - Trunc Commun Catalog). The user selects the interesting data objects by querying the TCC stored at his local TCC node. The system identifies through the PIDs (the Project IDentifiers which are EXIRPTS specific OIDs) the data-nodes (TCD - Trunc Commun Data) where the requested objects reside, sends a query through the network to those respective TCD nodes, and receives the available information for each data object requested.
Figure 3 shows the most general case for EXIRPTS project topology.

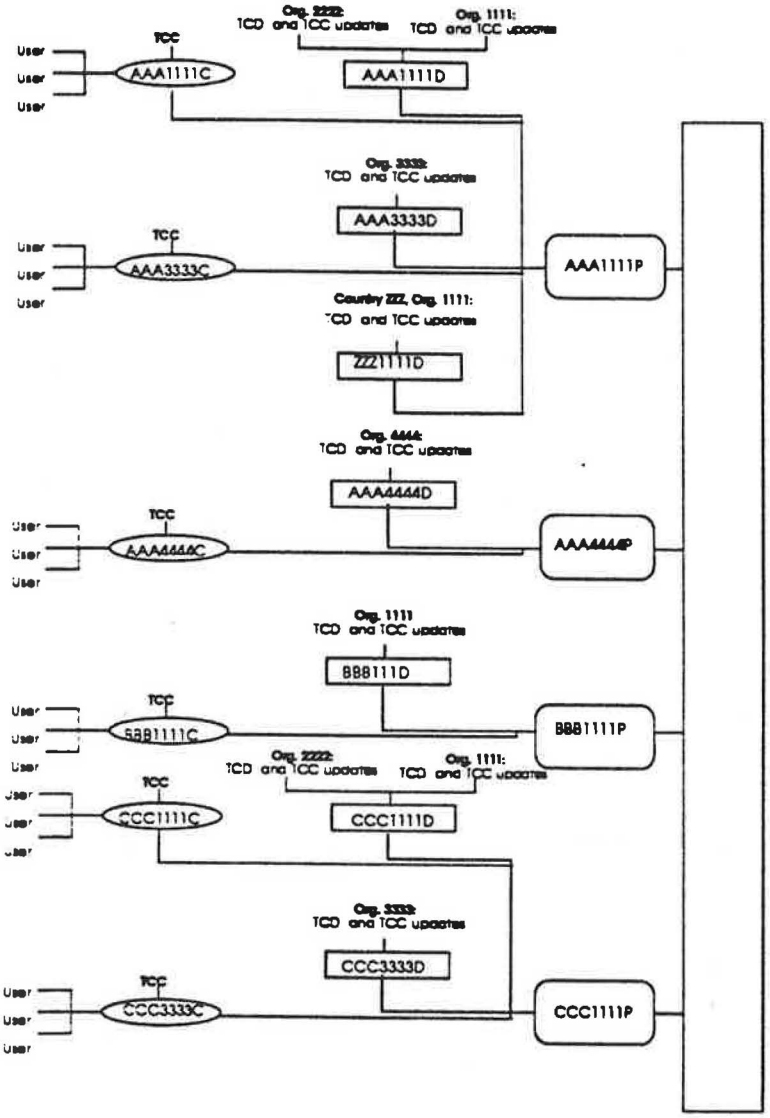

Fig. 3: Representation of a generalized EXIRPTS topology

In table 2 the format used to define the actuators is shown: this actuator identifier is called protocolname.

ccccoooP	for PSM actuators corresponding to Postmaster node actuator
ccccoooC	for TCC actuators corresponding to Catalog node actuator
ccccoooD	for TCD actuators corresponding to Data node actuator
"ccc" is the cluster code (country code), "oooo" is the site code (organization code).	

Table 2: Protocolname actuator format

In the case of EXIRPTS, the cluster corresponds to the country and the site corresponds to the organization.

Country AAAA has 2 PSM actuators, managed by organization 1111 and 4444 respectively. Organizations 1111, 3333 and 4444 have their own terminal network and manage by themselves their TCD and an autonomous copy of TCC. Organization 1111 provides postmaster services for organizations 2222, 3333 and, by a bilateral agreement, for organization 1111 of

country *ZZZ* as well. Organization 2222 has no (or does not want to use) computer facilities of its own and is fully supported (query, update, etc.) by organization 1111.

Country BBBB has just one site. In this case there is necessarily only one TCC and one TCD actuator.

Country CCCC has 3 sites. Organization 1111 is postmaster for all three sites. Organization 2222 is fully supported (terminal access, TCC query and update, TCD management) by organization 1111. Organization 3333 manages by itself its TCD and the copy of TCC.

The protocol covers only communications between actuators. In the EXIRPTS application there are modules interfacing both the systems managing the catalog, and the systems managing the organizational DBs [10]. This is shown using a simplified topology in figure 4.

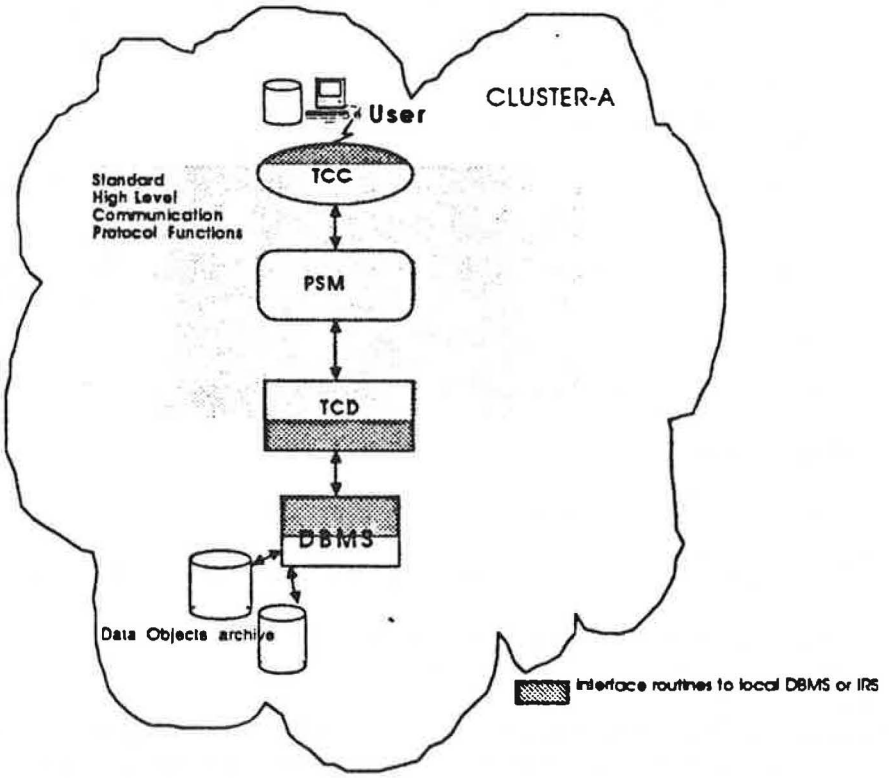

Fig. 4: Representation of interfaces to local DBMSs managing catalog and data.

6. IMPLEMENTATION

6.1 The High Level Protocol Messages

The basic functions listed in section 4.3 of this paper and needed to implement all the operations of the high level protocol are implemented through the following messages:

1. **TCD Query message (TCDQ):** containing the list of project-id (PIDs) of the projects (i.e. data objects) to be extracted from the local/remote TCDs. The message is originated by the user through his TCC and is built automatically by the TCC actuator and sent to the local PSM. The local PSM splits the list of Project IDentifiers and broadcasts the TCDQ to all the PSMs involved, including itself: the target TCDs are specified by the PIDs which define the TCD protocolnames as well as the projects. The PSM of the cluster containing the target TCD is then obtained by the control tables. The receiving PSM submits the message to its TCDs which manage the DB query session through interface routines.

2. **TCD Response message (TCDR):** containing data objects (project text) extracted from the local/remote TCD. The message is originated from each TCD and sent to the local PSM. The local PSM searches the name of the PSM from where the original query was sent and addresses the TCDR to it. The PSM keeps track of the incoming TCDQ and forwards the TCDR to the TCC. The TCC looks in the internal tables for the E-mail address of the user and forwards the TCDR to him.
   The TCD and PSM managers can always find out whether all the requested answers to a TCDQ have been satisfied. This can be done by sending a special message to see which remote TCD does not answer.

3. **TCC Update message (TCCU) and TCC Delete message (TCCD):** a TCCU contains the text of the TCC logical records to be added or modified in each copy of TCC at each TCC node. A TCCD contains the list of the PIDs of the projects to be removed from each copy of TCC at each catalog-node. They are originated by the TCD manager and broadcast to all TCCs through the PSMs. It is the responsibility of the TCC manager to apply updates and deletes.

4. **TCC modification Acknowledgment message (TCCA):** this message is needed to assure the TCD manager that the operations of update/delete have been completed successfully. It is originated by the TCC actuator interface routines managing the catalog and is sent to the TCD manager through the PSMs and the TCD actuator.

5. **TC Special Messages (TCSM)** are used by the node managers to perform certain actions on the network; to monitor for example the number of unsatisfied TCDQ at a certain catalog-node.

The metacode developed to implement these messages is shown in Appendix A.

6.2 Message Format

In the EXIRPTS application each message is sent/received as a file with fixed, 80-character length records containing only alphameric characters.
The translations of the character code (ASCI, EBCDIC, etc.) is requested of the network services.
A header/trailer pair is associated with each trasmitted file.
The header/trailer format is shown in table 3.
Upon the receipt/sending of any message, each actuator stores the header record of the incoming/outgoing message in a file (called the logfile); this file records all transactions which occur at each node over a certain period of time and constitutes a source of information for the compilation of statistics on the EXIRPTS network transactions (for example, to monitor the number of rejected messages, time of reply, etc.). This file can be erased by a special message which can be sent only by the local manager of the actuator.
The logfile format is shown in table 4.

6.3 Control Tables

Each EXIRPTS actuator keeps, together with the logfiles, other control files, called control tables and used for message routing.

There are two kinds of control tables:

1. internal to a cluster (to a domain) for messages between a PSM and a TCC, or a PSM and a TCD;
2. inter-cluster (domain) for messages between PSMs.

While internal tables can be managed locally from the EXIRPTS node/s involved, inter-cluster tables must be agreed upon by all EXIRPTS nodes.
The various control tables at each actuator are shown below.

#	cols.	leng	field name	content
1	01-08	8	TRANS-ID	protocolname of originating actuator
	09-14	6		transaction number
2	16-19	4	TRANS-TYPE	message type (eg. TCDQ, TCDR, etc.)
3	21-28	8	ORIGNODE	protocolname of originating actuator
4	30-37	8	DESTNODE	protocolname of final actuator
5	39-46	8	SENDNODE	protocolname of last sender ing actuator
6	48-51	4	NOLRECS	number of logical records (eg. PIDs)
7	53-59	7	DATE	production date: yyyyddd (GMT) from 1-Jan. set by originator
8	61-64	4	TIME	production time: hhmm (GMT) set by originator
9	66-66	1	LASTMSGFLG	in TCDR and TCCA messages: = "*" if no other messages are expected
10	68-75	8	PASSWORD	protection code of the sender ing actuator

Notes:

1. there is 1 space between each field.
2. TRANS-ID is set by the actuator which generates the message of the type TCDQ, TCCU or TCCD: the TCDR and TCCA messages inherit the same TRANS-ID of the correspondent TCDQ and TCCU TCCD respectively.
3. ORIGNODE (field 3) is set by the actuator which has generated the message by the local TCC actuator, in the case of TCDQ or TCCA, by the local TCD actuator, in the case of TCDR, TCCU or TCCD, and is left unchanged during the whole process.
4. DESTNODE (field 4) is set by the first actuator where the control tables needed to determe the destination are stored: in the case of TCDQ/TCCU/TCCD, the PSM actuator local to the TCC TCD actuator which has generated the message determines the DESTNODEs of the whole message or of part of it.
5. SENDNODE (field 5) is modified at each actuator and set with the current actuator protocolname determined by looking at the LOCAL TABLE.
6. LASTMSGFLD is set to "*" by a PSM actuator upon receipt of the last expected answer (TCDR or TCCA) for each pair TRANSID, (TCDQ or TCCU TCCD).
7. PASSWORD is a protection code in a cryptographic form peculiar to each sending actuator of the message (SENDNODE). The PASSWORDs decryption algorithms are exchanged between neighboring actuators.

Table 3: Message Header and Trailer Record Format

#	cols.	leng	field name	content
1	01-08	8	TRANS-ID	protocolname of originating actuator
	09-14	6		transaction number
2	16-19	4	TRANS-TYPE	message type (eg. TCDQ, TCDR, etc.)
3	21-28	8	ORIGNODE	protocolname of originating actuator
4	30-37	8	DESTNODE	protocolname of final actuator
5	39-46	8	SENDNODE	protocolname of last sender ing actuator
6	48-51	4	NOLRECS	number of logical records (eg. PIDs)
7	53-59	7	DATE	production date: yyyyddd (GMT) from 1-Jan. set at each actuator
8	61-64	4	TIME	production time: hhmm (GMT) set at each actuator
8	66-66	1	SEND REC	contains: "R" for received messages "S" for sent messages
9	68-68	1	COMPLET	completation flag: = "*" when all expected answers have been received

Notes:

1. Records are removed by a suitable message sent by the local manager of the actuator from the current logfile to an archive logfile after some time, that is, when the date of the logfile record is equal to the current date minus "interval"; "interval" is a modifiable parameter and must be the same for all actuators.
2. COMPLET is set to "*" by each actuator on receipt of the last expected answer for each TRANSID.

Table 4: Logfile Header and Trailer Record Format

Control tables at any actuator

LOCAL table

In the first record it contains the protocol name and E-MAIL address of the actuator itself. In the second record it contains a signed three digit decimal number representing the TIMEZONE differential from GMT to be used in time conversions.

FIRST RECORD

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of the local actuator
2	10-80	71	E-MAIL	full E-mail address of the local actuator

SECOND RECORD

#	cols.	leng	field name	content
1	01-03	4	TIMEZONE	TIMEZONE differential from GMT

MSG table

It contains a record for each message defined by the protocol: This is used for recognizing the type of message received:

#	cols.	leng	field name	content
1	01-08	8	TRANS-TYPE	string identifying a type of message defined by the protocol

Control tables at TCC actuator

PSMNAME table
It contains the name and full E-mail address of the local PSM actuator. This is for sending the TCDQ or TCCA file produced to the local PSM actuator.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of the PSM actuator
2	10-80	71	E-MAIL	full E-mail address of the PSM actuator to which the TCC actuator is connected

ENDUSER table
It contains a record for each TCDQ transaction, providing a link to allow returned TCDR to be sent to the end-user.

#	cols.	leng	field name	content
1	01-06	6	TRANS-N	transaction number (set by the TCC itself)
2	08-71	63	E-MAIL	user full E-mail address
3	73-80	7	ACTIVE	set to " " only for active TCDQ, i.e. TCDQ for which not all expected TCDRs have been received

TRANSNUM table
It contains the current value of the transaction number, incremented for each new TCDQ message.

#	cols.	leng	field name	content
1	01-06	6	TRANSNUM	transaction number padded with zero

Control tables at TCD actuator

PSMNAME table

It contains the name and E-mail address of the correspondent PSM actuator. This is for sending the TCDR, TCCU, TCCD messages produced to the local PSM actuator.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of the PSM actuator
2	10-80	71	E-MAIL	E-mail address of the PSM actuator to which the TCD actuator is connected

MNGNAME table

It contains the name and E-mail address of the TCD manager. It is used by the TCD actuator to check the sender of a TCCU and TCCD file, and to send back TCCA messages, statistical data and other results of special messages.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of the TCD manager
2	10-80	71	E-MAIL	E-mail address of the manager

TRANSNUM table

It contains the current value of the transaction number, which is incremented for each new TCCU and TCCD message.

#	cols.	leng	field name	content
1	01-06	6	TRANSNUM	transaction number padded with zero

Control tables at PSM actuator

PSMTABLE table
It contains a record for each (organization-country) that provides a TCD archive and is used to split/send TCDQ/TCCA messages from a local TCC actuator to the PSM actuators.

#	cols.	leng	field name	content
1	01-08	8	PRTNAM1	protocolname of a TCD actuator
2	10-17	8	PRTNAM2	protocolname of the PSM actuator local to the TCD actuator

TCDLIST table
It contains a record for each TCD actuator in its domain and is used to send TCDQ/TCCA files received from a PSM actuator to the local TCD actuators.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of a TCD actuator
2	10-80	71	E-MAIL	E-mail address of the TCD

TCCLIST table
It contains a record for each TCC actuator in its domain and is used to broadcast, within the domain, TCDR, TCCU, TCCD messages received from a PSM actuator.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of a TCC actuator
2	10-80	71	E-MAIL	E-mail address of the TCC

PSMLIST table
It contains one record for each PSM actuator in the EXIRPTS network. This table describe the topology of the EXIRPTS domains. It is used,for example, to broadcast TCCU, TCCD messages originated by the manager of a local TCD actuator to all PSM actuators

#	cols.	leng	field name	content
1	01-04	4	PRTNAME	protocolname of a PSM actuator
2	10-80	71	E-MAIL	E-mail address of the PSM

MNGNAME table
It contains the name and E-mail address of its manager. It is used by the PSM actuator to check the sender of a special message, for example, of a statistical data collection message.

#	cols.	leng	field name	content
1	01-08	8	PRTNAME	protocolname of PSM manager
2	10-80	71	E-MAIL	E-mail address of PSM manager

7. DISCUSSION

7.1 Advantages

The architecture developed, described in section 4, meets the requirements listed in section 3. Certain particular features, however, deserve closer consideration:

Network independence

The protocol is designed so that the control information required to interpret the kind of message is contained within the body (text-part) of the message sent. No use is made of the control information supplied by the E-mail service itself, upon which this protocol is supported.

This means that the E-mail services are used only to deliver the message from one node to another; at each node the actuator 'unpacks' the message and interprets it by decoding the information found in its header record, data and trailer record.

Independence between Actuator names and E-mail addresses

Catalog-actuators and data-actuators within any cluster only need to know the E-mail address of their cluster postmaster-node. The E-mail addresses of data- , postmaster- and catalog-nodes external to the cluster are invisible to the catalog and data-actuators of the cluster. Only the postmaster-actuator knows the E-mail addresses of the postmaster-nodes of all the other clusters. This allows the manager of a cluster to reorganize his internal distribution of the catalog and databases independently of the other clusters, by simply modifying the control tables at its internal nodes. There is no need to inform the other cluster managers, except in the particular case in which the postmaster-node definition is changed.

Local DBMS independence

Query on a database through a DBMS at a local node is directed by an interface routine called by the data-actuator at that node. The data-actuator interface routine, which receives the DB query message from a catalog-node elsewhere on the network, transiates it into the local DBMS query language. Thus, the contents, structure and query language of any DBMS (or group of DBMSs) supported locally for organizational purposes is hidden from the end-user by the high-level protocol.

The user can therefore access any database in the network only indirectly by querying the catalog at his own site and using the query language of the local IRS or DBMS that manages the catalog.

Different query languages and different DBMS and IRS can thus co-exist in DIRS. Furthermore, updating a catalog content at a catalog-node for changes in any one of the databases in the network, is decided by the organizational manager of the catalog-node itself. It is not a procedure imposed by the high-level protocol. However, data-nodes managers are informed when their requested updates have been applied to the catalog-nodes.

Query efficiency

With respect to the conventional batch query approach, the strategy of a "two stage query" (addressing the catalog and then the data) ensures that the query on the DBMSs supporting local DBs is limited to the set of objects which satisfy the initial catalog query. This improves the efficiency of retrieval, as iterative query formulation based on a relevance feedback strategy can be applied locally when querying the catalog, avoiding the overloading of the network.

Breakdown into domains and levels

The architecture, consisting of an external network linking postmaster-nodes, and internal networks supporting the cluster of catalog- and data-nodes linked to individual postmaster-nodes, provides a clear distinction between a local domain and distant domains. This has implications for management of the distributed system. In particular, the postmaster-node can isolate its particular cluster (local domain) from the external network (and from all distant domains).

Privacy of data assured at domain level

Frequently, a domain may be a nation-state. Thus, insolating the domain from the

external network by controls at the postmaster-node is of importance for security. The postmaster-actuator has access to tables (control tables) which give the nodename and E-mail addresses of all nodes internal to its cluster and the nodename and E-mail addresses of all postmaster-nodes in the external network. The postmaster-node tables are controlled by the postmaster-node manager (a person). If an entry for a particular postmaster-nodename is not present, the correspondent cluster cannot communicate with nodes in the cluster domain controlled by that postmaster-node.
Finally the data-nodes managers can control the viewing of their local databases by end-users via catalog-nodes at two levels: (1) they decide what data objects are accessible through the catalogs; (2) they decide what attributes of the data objects from the local databases may be made available to the users of each cluster. In other words, the data-actuator on each local data-node in the domain controls what data object fields are returned to the postmaster-node (and hence to the requesting user at a catalog-node) as a result of a query formulated by that user.
In brief, the network traffic is under complete control.

Monitoring

There is no direct information about the origination of the transaction, except at the originating node where end user identification is recorded. This means that within the distributed system all messages are anonymous (and thus have some degree of privacy), although each message has an originating nodename and destination nodename which identify the extreme points of the path which the message traverses in the network when it is successfully executed. At each node traversed by the message its header is recorded in a logout file. The name of the sender node is rewritten at each node in the header to allow the receiving node to control whether the message comes from an authorized node. The matching of the 'outbound' and 'inbound' messages can only take place by means of a transaction identifier encoded in the header/trailer records of the messages. The header also contains the originating time; the standard time (GMT, Zero time) is recorded in the node logfile as the message passes through each node in its path through the network. Thus, the performance of the network can be monitored, and the path of any message traced both geographically and in time.

7.2 Disadvantages

Lack of centralised control

The system has been designed for distribution on the heterogeneous computing systems of different organizations in different countries, with varying local IRSs or DBMSs. This is advantageous for many environments, but for some the lack of centralized control could be seen as a disadvantage.

Overload In the current design, the header and trailer records of a message are modified at each node in the message path through the network, and a logfile is written. This means that in some operating environments, such as the VM-CMS operating system where no direct access to spool files is possible, the whole message must be read at every node. However, this strategy was chosen to improve reliability and security. Monitoring the system in full production load will make it possible to determine the cost/benefit ratio of this approach.

7.3 Further Developments

Data-actuator query refinement

At present the data-actuator receives a list of Object IDentifiers (OIDs) from the domain postmaster-node. The data-actuator retrieves information about objects identified by the OIDs and contained in its local database(s) and returns as a response a subset of this information as specified by the data-node manager. As the information on objects is structured in fields, identified by attributes, the database manager may decide to make only some of these fields available.

However, if the end user is not interested in some of the available fields, there is no way to state that in the present development of our system; this results in a useless overloading of the network. There could be potential advantages in expanding the TCDQ message by allowing more complex TCD query strategies, based on some knowledge of the object structures available at each local database. This knowledge could, for example, be gained by including in the high level protocol a new type of message allowing a user to query a database on its object structure. The implications of this solution have not been thoroughly investigated.

Support for the manager of a postmaster-node

The person managing a postmaster-node has a great deal of responsibility including security considerations. We propose that he be given tools which will allow him to have:

1. a view of all nodes in the network, both local and distant, with a clear indication of the domains controlled by postmaster- nodes;
2. commands to interrogate logfiles at the local postmaster-node, to search for particular transactions;
3. commands to interrogate logfiles on distant postmaster-nodes, to search for particular transactions or to monitor performance;
4. commands to cause the generation of special messages (eg stop/restart transaction, suspend transmission).

APPENDIX A: THE PROTOCOL MESSAGE DESCRIPTION

A graphic representation of the message path through the nodes of DIRS at different detail levels and by means of Petri Nets notation can be found in [11]. Here a description of the main operations each message involves, at each actuator is given by using a metacode language.

TCDQ message

Originator : A user interacting with the TCC through the local DBMS

Destination : Any TCD archive managed by a TCD actuator

Operations :

1. At the TCC actuator
   a. interception of a user request sent by the manager of the TCC, this operation is performed by the interface routine to local DBMS or IRS managing the caltalog
   b. writing a receive record in logfile
   c. writing a record in ENDUSER file: this operation is environment dependent and performed by an interface routine
   d. preparation of the header/trailer records:
      - assignment of the transaction number and its incrementation in TRANSNUM table file
      - assignment of the password: the subroutine or function performing this operation is identified by the same name as the protocol name of the current actuator (first 8 characters in LOCAL table file)
      - computation of date and time in GMT by converting the local time using the TIMEZONE value in the second record of LOCAL table file.
      - setting SENDNODE, ORIGNODE with local protocol name
      - setting of type = TCDQ
      - setting of NOLRECS with number of records between header and trailer
   e. preparation of the TCDQ file with:
      - header
      - PIDs list
      - trailer
   f. sending the TCDQ file to local PSM actuator (it takes the PSM actuator address from PSMNAME file)
   g. writing a send record into the logfile
2. At the local PSM actuator
   a. reading the TCDQ message
   b. checking the password field: there must exist a subroutine (or function) supplied by the manager with the same name as the protocol name of the sender (equal to the value of field SENDNODE in header): this subroutine decrypts the password in the header, and if the password does not present the expected value, the file is rejected
   c. writing a receive record into the logfile
   d. sorting PIDs list by country and organization code
   e. spliting PIDs list by PSM actuators into separate files (it uses PSMTABLE file to find the PSM actuator corresponding to each country-organization pair)
   f. for each file produced at step 4.:
      - preparation of header/trailer:
        - assignment of the password (as at TCC actuator)
        - setting SENDNODE using LOCAL table file
        - setting DESTNODE decoding the PIDs

- setting NRECS with number of records between header and trailer
- sending to correspondent PSM actuator (it gets address from analysing first PSMTABLE file, then PSMNAME file)
- writing a send record into the logfile

3. At each receiving PSM actuator

   a. reading the TCDQ message
   b. checking the password field (see above)
   c. writing a receive record into logfile
   d. modifying header/trailer records
      - assignment of the password (as at TCC actuator)
      - setting SENDNODE field
   e. sending file to correspondent TCD actuator (adress from DESTNODE in header)
   f. writing a send record into the logfile

4. At each receiving TCD actuator

   a. reading TCDQ message
   b. checking the password field (as at PSM)
   c. writing a receive record into the logfile
   d. querying the TCD data base by calling interface routines

TCDR message

Originator : A DBMS or IRS managing a TCD archive

Destination : End user who submitted a request involving that TCD

Operations :

1. At the TCD actuator
   a. interception of an answer to a previous query from the local DBMS or IRS: interface routines perform this operation.
   b. Preparation of the header and trailer records:
      - assignment of the password (as at TCC)
      - setting TRANSID equal to TRANSID of corresponding TCDQ
      - setting ORIGNODE and SENDNODE (using LOCAL table file)
      - setting DESTNODE equal to ORIGNODE of corresponding TCDQ
      - setting type = TCDR
      - setting completition flag field to '*' upon directions of interface routine: this means that all answers to that TCDQ has been satisfied
      - setting NOLREC equal to N. records between header and trailer
      - computation of date and time in GMT converting local time with TIMEZONE in LOCAL table file
   c. preparation of the TCDR file with:
      - header
      - TCD data
      - trailer
   d. sending TCDR file to local PSM (it takes the PSM actuator adress from PSMNAME file)
   e. writing a send record into the logfile
2. At the local PSM actuator
   a. reading the TCDR message
   b. checking the password field (as at TCC)
   c. writing a receive record into the logout file
   d. if header has completition flag field set to '*':
      - updating the logfile send records corresponding to a TCDQ whose TRANS-ID is equal to the TRANS-ID in header
      - setting the completition flag variable to '*'
   e. preparation of the header and trailer records:
      - assignment of the password (as at TCC)
      - setting SENDNODE (using LOCAL table file)
      - setting the completition flag field equal to completition flag variable value
   f. sending the TCDR file to the PSM actuator (it gets E-mail address from PSMLIST file)
   g. writing a send record into the logfile
3. At the receiving PSM actuator
   a. reading the TCDR message
   b. checking the password field (as at TCC)
   c. writing a receive record into the logfile
   d. if header has completition flag field set to '*':
      - updating the logfile send record corresponding to a TCDQ whose TRANS-ID is equal to the TRANS-ID in header
      - if logfile does not contains other send records with of the kind above with completition flag field = ' ', setting the completition flag variable to '*'
   e. preparation of the header and trailer records:
      - assignment of the password (as at TCC)
      - setting SENDNODE (using LOCAL table file)

- setting completition flag field equal to completition flag variable value

f. sending TCDR file to the TCC actuator (it takes the E-mail address from TCCLIST file)
g. writing a send record into the logfile

4. At the receiving TCC actuator

a. reading the TCDR message
b. checking the password field
c. writing a receive record into logfile (it sets the completition flag field as in the header of the file)
d. sending TCDR to end user: this is performed by interface routines which get the E-mail address of end user from ENDUSER file

TCCU message

Originator : The TCD manager

Destination : All the TCC actuators in the EXIRPTS network

Operations :

1. At the TCD actuator
   a. intercepting an update request file from TCD manager: this operation is performed by an interface routine.
      - checking the password field corresponding to the authorized manager name in MNGNAME
   b. writing a receive record into the logfile
   c. preparating header and trailer records:
      - assignment of the transaction number and its incrementation in TRANSNUM table file
      - assignment of the password
      - setting type = TCCU
      - setting TRANSID field with TRANSNUM and first 8 characters of LOCAL table file
      - setting ORIGNODE and SENDNODE equal to the first 8 characters of local table file
      - computation of date and time in GMT by converting local time with TIMEZONE value in LOCAL table file
      - setting NOLRECS with number of records between header and trailer
   d. preparation of TCCU message
      - header
      - PIDs logical records (80 characters x 40 records) list
      - trailer
   e. sending the TCCU file to local PSM actuator (it takes the PSM actuator adress from PSMNAME file)
   f. writing a send record into the logfile
2. At the local PSM actuator
   a. reading the TCCU message
   b. checking the password field
   c. writing a receive record into the logfile
   d. for each PSM actuator in the EXIRPTS network, i.e for each record in PSMLIST file:
      - preparation of header trailer records:
        - assignment of the password
        - setting SENDNODE using LOCAL table file
      - sending the file to the PSM actuator corresponding to current record in PSMLIST file
      - writing send record into the logfile
3. At each receiving PSM actuator
   a. reading the TCCU message
   b. cheking the password field
   c. writing a receive record into the logfile
   d. for each record in TCCLIST file:
      - preparation of header trailer records:
        - assignment of the password
        - setting SENDNODE using LOCAL table file
        - setting DESTNODE using current record in TCCLIST file
      - sending the file to the TCC local actuator using current record in TCCLIST file

- writing a send record into the logfile

4. At each receiving TCC actuator

   a. reading the TCCU message
   b. cheking the password field
   c. writing a receive record into the logfile
   d. sending the TCCU file to DBMS or IRS for applying the update: this operation is performed by an interface routine

TCCD message

Originator : The TCD manager

Destination : All the TCC actuators in the EXIRPTS network

Operations :

1. At the TCD actuator
   a. intercepting a delete request file from TCD manager: this operation is performed by an interface routine.
      - checking the password field corresponding to the authorized manager name in MNGNAME file
   b. writing a receive record into logfile
   c. preparating header and trailer records:
      - assignment of the transaction number and its incrementation in TRANSNUM table file
      - assignment of the password
      - setting type = TCCD
      - setting TRANSID field with TRANSNUM and first 8 characters of LOCAL table file
      - setting ORIGNODE and SENDNODE equal to the first 8 characters of local table file
      - computation of date and time in GMT by converting local time with TIMEZONE value in LOCAL table file
      - setting NOLRECS with number of records between header and trailer
   d. preparation of TCCD message
      - header
      - PIDs logical records (80 characters x 40 records) list
      - trailer
   e. sending the TCCD file to local PSM actuator (it takes the PSM actuator adress from PSMNAME file)
   f. writing a send record into the logfile
2. At the local PSM actuator
   a. reading the TCCD message
   b. checking the password field
   c. writing a receive record into logfile
   d. for each PSM actuator in the EXIRPTS network, i.e for each record in PSMLIST file:
      - preparation of header trailer records:
        - assignment of the password
        - setting SENDNODE using LOCAL table file
      - sending the file to the PSM actuator corresponding to current record in PSMLIST file
      - writing send record into logfile
3. At each receiving PSM actuator
   a. reading the TCCD message
   b. cheking the password field
   c. writing a receive record into the logfile
   d. for each record in TCCLIST file:
      - preparation of header trailer records:
        - assignment of the password
        - setting SENDNODE using LOCAL table file
        - setting DESTNODE using current record in TCCLIST file
      - sending the file to the TCC local actuator using current record in TCCLIST file

- writing a send record into the logout file

4. At each receiving TCC actuator

   a. reading the TCCD message
   b. cheking the password field
   c. writing a receive record into the logfile
   d. sending the TCCD file to DBMS or IRS for applying the deletes: this operation is performed by an interface routine

TCCA message

Originator : The DBMS or IRS managing the TCC

Destination : Manager of TCD

Operations :

1. At the TCC actuator
   a. intercepting a message, stating the successful application of an update or delete command, sent by an interface routine of the DBMS or IRS
   b. preparation of the header trailer records
      - assignment of the password
      - setting type = TCCA
      - setting TRANSID equal to TRANSID of correspondent TCCU or TCCD message
      - setting ORIGNODE, SENDNODE by using LOCAL table file
      - setting DESTNODE equal to ORIGNODE of corresponding TCCU or TCCD
      - setting date and time in GMT by converting local time with TIMEZONE in LOCAL table file
   c. preparation of the TCCA message with:
      - header
      - message from interface routine stating the status of DB after updates or deletes
      - trailer
   d. sending TCCA message to local PSM actuator using PSMNAME file
   e. writing a send record in logfile
2. At the local PSM actuator
   a. reading the TCCA message
   b. checking the password
   c. writing a receive record into the logfile
   d. setting completition flag:
      - setting completition flag with '*' in the logfile send record corresponding to a TCCU or TCCD whose TRANS-ID is equal to the TRANS-ID in header
   e. preparation of the header and trailer records:
      - assignment of the password (as at TCC)
      - setting SENDNODE (using LOCAL table file)
   f. sending the TCDR file to the PSM actuator (it gets E-mail address from PSMLIST file)
   g. writing a send record into the logfile
3. At the receiving PSM actuator
   a. reading a TCCA message
   b. checking the password field
   c. writing a receive record into the logfile
   d. setting completition flag:
      - setting completition flag with '*' in logfile send record corresponding to a TCCU or TCCD whose TRANS-ID is equal to the TRANS-ID in header
      - if no other record in logfile is of the kind above and has completition flag field = ' ', then set completition flag variable to '*'
   e. preparation of the header and trailer records:
      - assignment of the password (as at TCC)
      - setting SENDNODE (using LOCAL table file)
      - setting completition flag field equal to completition flag variable value
   f. preparation of TCCA file

- header
- message
- trailer

g. sending the TCDR file to local TCD actuator getting E-mail address from TCDLIST file
h. writing a send record into the logfile

4. At the receiving TCD actuator

a. reading the TCCA message
b. checking the password field
c. writing a receive record into logfile
d. in the send record of logfile with TRANS-ID equal to TRANID in header and type TCCU or TCCD setting completition flag as in the header
e. sending TCCA message to TCD manager (it takes the E-mail address from MNGNAME table file)

REFERENCES

1. Caplinger, M.; An Information System Based on Distributed Objects; Proceedings of OOPSLA, 1987.
2. Jablonski, S.; Ruf, T.; Wedekind, H.; Implementation of a distributed data management system for technical applications - A feasibility study: Information Systems, Vol. 15, No. 2; pp.247-256, 1990.
3. Thomas, R.H.; Forsdick, H.C.; Crowley, T.R.; Schaaf, R.W.; Tomlinson, R.S.; Travers V.M.; Robertson, G.G.; IEEE Computer; December 1985.
4. Macleod, I.A.; Martin T.P.; Nordin, B.; Phillips J.R.; Strategies for building distributed Information Retrieval Systems; Information Processing & Management; Vol. 23; No. 6; pp. 511-528; 1987.
5. Jeffery, K.G.; Lay, J.O.; Samir Zardan, J.F.M.; Naldi, F.; Vannini Parenti, I.; IDEAS: A System for International Data Exchange aand Access for Science; Information Processing and Management, Vol.25, No. 6, pp. 703-711, 1989.
6. Salton, G.; McGill, M.J.; Introduction to modern Information Retrieval, McGraw-Hill, 1884.
7. VanRijsbergen, C.J.; Information Retrieval; Butterworths, 1979., McGraw-Hill, 1884.
8. Litwin; From Database Systems to Multidatabase System, Proc. IV British National Conference on Data Bases, Cambridge Univ. Press, pp. 161-188, 1988.
9. Swith, J.M.; Bernstein, P.A.; Dayal, J.; Goodman, N.; Landers, T.; Lin, K.W.T.; Wong, E.; MULTIBASE Integrating Heterogeneous Distributed Database Systems, Proc. 1981 Nat. Computer Conference, June 1981.
10. Naldi, F.; Bordogna; G.; The ENID2 protocol technical specification and implementation in VM-CMS environment, SIAM-IFCTR-CNR Internal Report, 9-1990.
11. Bordogna, G.; Modelling of the TCDQ and TCDR processes of the ENID protocol by Petri Nets, SIAM-IFCTR CNR Internal Report, 8-1990.