The design of the OSIRIDE Management functions.

E.GREGORI, L.LENZINI, C.MENCHI

Rapporto interno C84-26

Pisa, Maggio 1984
Istituto CNUCE - CNR

TABLE OF CONTENTS

LIST OF FIGURES

Abstract: This document describes what and how the Network Management designed in the first phase of the OSIRIDE project can be used to meet the initial requirements of the OSIRIDE environment.
The criteria for splitting and distributing responsabilities among the OSIRIDE End-Systems have been dictated by the need to have, on one hand, a planning center for the network and, on the other the need to cope with a large set of manufacturers minimizing the work to do in order to wait for stable OSI choices.

1.0 DESIGN PRINCIPLES

1.1 OVERVIEW OF THE OSIRIDE MANAGEMENT FUNCTIONS.

In this section the Network Management System (NMS) requirements are briefly described.

1.1.1 Alarm Handling

Any time a fault is detected the OSIRIDE software should be able to send an alarm to a specialized End-System for its analysis and solution. The history of each alarm and its solution is stored in a data-base which can be accessed to retrieve information during the alarm treatment phase.

1.1.2 Diagnostic Handling

In general, the report provided by an alarm is not specific enough to allow an immediate determination of repair action. The NMS must have facilities for further isolating the fault until there is a high degree of certainty as to the repair responsibility and scope. This usually involves the run of special modules that measure detailed aspects of OSIRIDE operation in both the frequency and time domain.

1.1.3 Data Collection

It should be able to record each End-System traffic in order to determine End-System usage and to detect potential overloads. In particular for each OSIRIDE Connection between Applications, information such as number of bytes exchanged (sent/received) during the life of the Connection, times at which the Connection was established and released should be available for accounting purposes.

1.1.4 Software Distribution

New versions of OSIRIDE software modules and lists must be distributed to the appropriate End-System. There must be a capability for loading End-Systems with the current software version after a system start-up. The OSIRIDE software configuration which is currently working should be stored in a well defined End-System which is

named Network Control Centre (NCC) from which it can be retreived if necessary. Software Distribution operation is controlled by NCC. However the real software distribution is performed by an End-System which is named Distribution Host (DH). In principle there is a DH for each make.

1.1.5 Notify

There must be the capability for an active Application to **notify** its availability to accept connection requests coming from one or all remote End-Systems.

In this way, it may be avoided:

1. that Applications start timeouts or loops to repeat connection requests if the remote counterpart is not active[1]

2. that in any End-System, functions are implemented to activate Applications when connection requests arrive. These functions, which may be applicable only to Applications already known by the NMS, would in fact limit the flexibility, which is a requirement of the research environment where OSIRIDE is running.

Any Application which requires a connection to a not yet active counterpart may ask the NMS to hold its request until the related NOTIFY arrives.

1.1.6 Transparent Transfer Command

There must be a capability to transfer a command, issued by an OSIRIDE operator, transparently to a remote End-System according to the syntax and semantics of the related local implementation. A completion code and any message generated by that command must be returned to the command originator.

Authorization is needed for using this capability.

---

1 By **not active Application** we mean an Application which did not declare itself to its NMS, or which resides on an End-System not reacheable at the moment.

1.2 OSIRIDE GENERAL ORGANIZATION

The facilities provided by OSIRIDE Network Management System (NMS) are achieved in a distributed way, i.e. the various modules implementing each of them may reside on different End-Systems using a management protocol for information exchange.

These modules are located in the Application Layer and therefore the information exchange among them is performed by means of Application protocols. In order to meet the communication requirements of management processes, a set of **Management Service Elements** have been defined, each providing an independent management capability. The Management Service Elements constitutes the boundary between the NMS and the communication system by defining an abstract interface which hides the actual messages exchanging. By this way the Management Service Element allow the NMS to be designed independently on the communication system.

It is worth noting that this approach fully complies with OSI directives.

1.2.1 OSIRIDE roles definition

Some of the NMS modules **must reside on each End-System** constituting its Local Management System (**LMS**) and some others may reside on **one or more End Systems** constituting the Control Management System (**CMS**) and/or the Field Service support center (**FS**) i.e the End-System entitled to manage technicians intervention on failed resources[2].

Besides the LMS, CMS, FS roles, the Data Accumulation Center (**DAC**) role exists and it is played by a unique system in the network; it is in charge of collecting and storing data concerning error history and resource utilization.

A detailed description of the NMS roles will be given in chapters 2.1, 2.2 and 2.3.

1.2.2 OSIRIDE roles distribution

The following figure shows how the NMS design was used to meet the OSIRIDE requirements.

---

[2] At the moment FS role has been envisaged but no specialized module was developed. Initially, in OSIRIDE, this role will be played manually by the ACC operator (see next chapter).

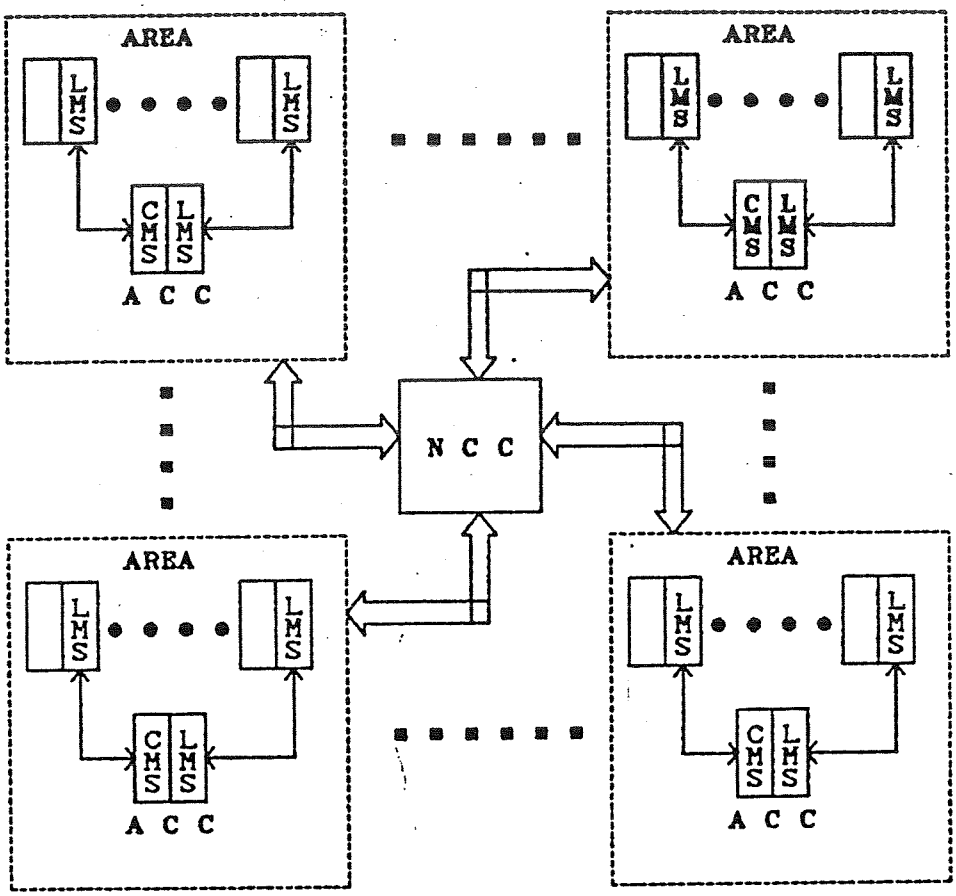

Fig. 1. OSIRIDE management functions physical organization

The Network Control Center (NCC) is unique in the OSIRIDE environment and is responsible for all the planning activities. It plays the DAC role and has the responsibility of managing the OSIRIDE software distribution.

End-Systems of the same maker are put together to form a homogeneous area. One of these End-Systems plays the CMS and FS role for the whole area and is named Area Control Center (ACC).

All the End-Systems in the network must support Local Management System functions to control local network functions and to cooperate with their ACC.

2.0 MANAGEMENT ARCHITECTURE

This section gives an example of LMS, CMS and DAC architectures. The same functionality can be obtained with different architectures but the aim of the section is to clearly point out what to do and not to put unnecessary constraints to the implementor.

2.1 LMS ARCHITECTURE

The LMS consists of a set of functional modules which allow:

- managing of the local OSIRIDE user activity and resources;
- managing of the **End-System** network resources;
- providing of the distributed management functions.

The structure of the OSIRIDE LMS is shown in Fig. 2 pag. 7. It includes the Handlers, data flows and interfaces to the local environment and the OSIRIDE software.

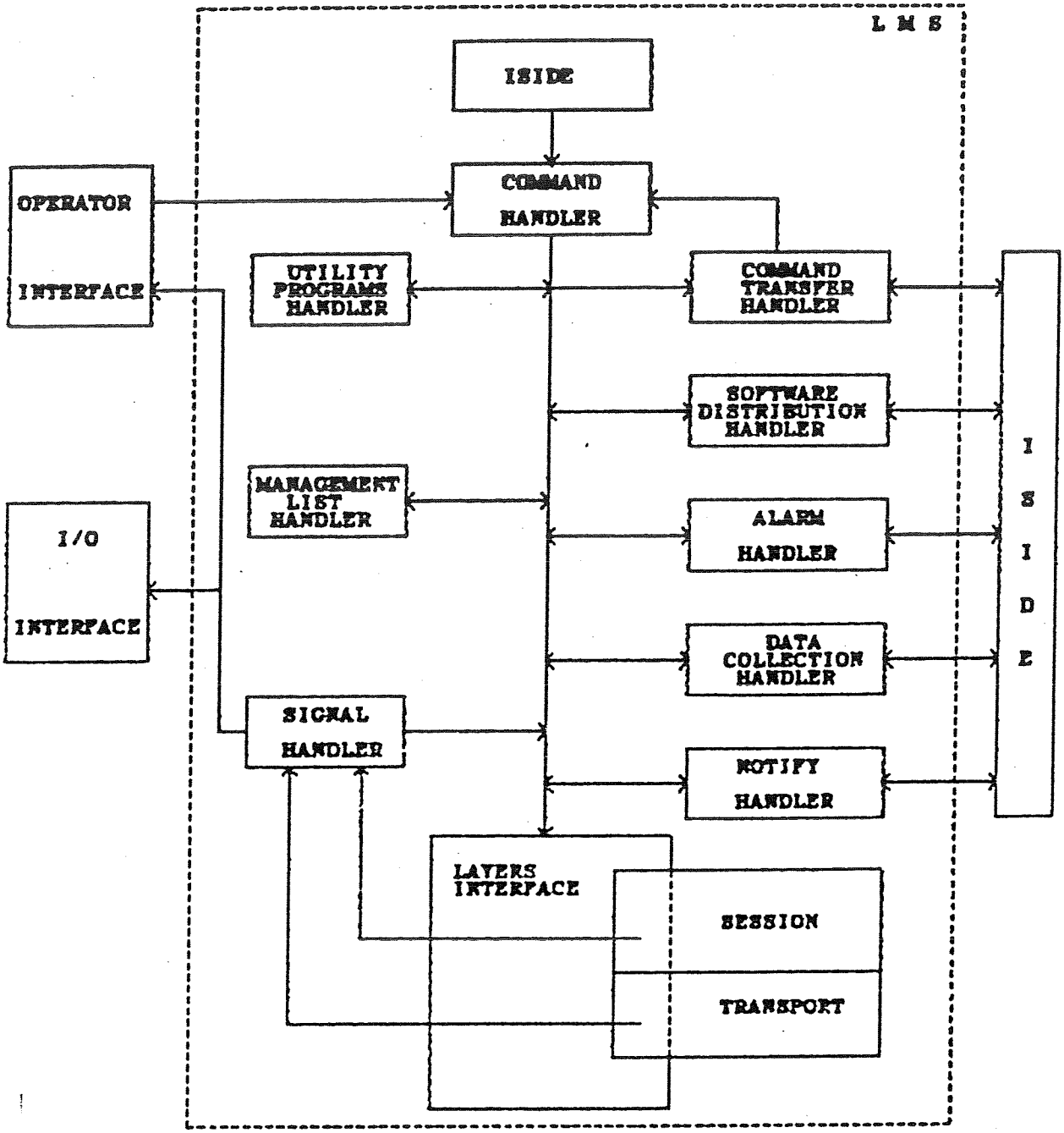

Fig. 2. LMS Architecture

The following sections contain a brief description of the LMS's functional modules. The aspect related to the Network Software activation is not considered here because it is strictly correlated with the Operating System facilities.

The architectural modules' design also considers that the management functions handle data. From a logical point of view such data are organized in a set of lists. Each list has one or more elements containing one or more fields. In order to avoid information incoherence in the lists (e.g. two simultaneous but contrasting updatings), all operations relative to one list or one set of lists are grouped under one **executor**. Each executor is provided with an entry queue where all requests relative to that list will be enqueued to be processed by the executor following the FIFO discipline.

The executors are therefore the only ones authorized to carry out the operation directly on the management lists.

2.1.1 Operator Interface

It allows the LMS operator to interact with the LMS software through the LMS commands. It is also used by the LMS software to send signals to the operator consol.

2.1.2 Command Handler

It performs a syntactic check of the command and controls if the originator is authorized to send such command [3]. Once the job is accomplished, it informs the source of the correct execution sending also every possible message that may be produced by the execution of the command.

The commands may arrive at the LMS either locally by means of the operators or by means of the Applications through ISIDE, or remotely by means of the **Transparent Transfer Command** .

Every command known by the Command Handler is associated to an authorization class so that only sources having the requested class may ask for execution of the command. A remote source (operator or Application) is identified also by the name of the End-System from which it has asked for transmission.
It is worth while tc point out that commands exchanged between Management Application processes, advanced by means of Service Elements, are not subjected to authorization. This is to state that

---

[3] The couple management command and target resource identifies the necessary authorization class.

the authorization is an operation which is performed before enqueueing a request to a generic management function.

2.1.3 Utilities Handler

It keeps a list of utility tools (e.g. to collect statistic data with a certain time frequency) and activates them, upon request by the operator or by the concerned functions (e.g. Data Collection Handler).

It can activate these utilities even at pre-established times with the requested initialization values.

2.1.4 Management List Handler

The Management List Handler performs the main role of handling the management data structure.
It is made of all the LMS-list's executors which are entitled to update the management data structure.

2.1.5 Signals Handler

The Signals Handler (SH) performs the main role for distribution of signals. Whenever any process detects an occurrence which could be of interest to other processes, it generates a signal and sends it to the SH.

Signals may be alarms, advisory reports exceeding a pre-defined threshold, etc.

The SH first filters the signal according to selection criteria and then forwards them to the appropriate destination(s).

Information filtering is available over a large number of classes or origin. For example, a hardware maintainer could view only signals from components of a particular type, or those which appear to be hardware-related, or which exceed a particular level of severity.

2.1.6 Layers Interface Handler

The LMS needs a particular interface with the OSI-Layers. This handler contains the interfaces between LMS and the CSIRIDE Session

and Transport layers. The documents of the Transport and Session layers contain a detailed description of these two interfaces.

2.1.7 Notify Handler

The Notify Handler offers the OSIRIDE users two inter-related capabilities:

1. a function to hold a request of connection towards remote Applications that are not active at request time

2. a function to alert one or all the Applications in one or all the remote hosts that the user is ready to accept connection request which may have been previously held.

2.1.8 Transparent Transfer Command Handler

This service is called by the command requesting transparent transmission of a string of data (LMS Command) to a Command Handler residing on a remote LMS.
The command sent, if syntactically correct and authorized, will be executed as a locally given LMS command.

The "completion code" and all possible messages are sent back to the originator. The identifier of the source (OPERATOR or name of PSAP of the Application) is transmitted together with the string of characters.

Two authorization levels are thus obtained:

1. **local** control of the right of the source to ask for transmission of the command

2. **remote** control of the right of the source to ask for execution of the command on the receiving END-SYSTEM [4].

Given that no analysis of transmitted string of data is done at the sending LMS side, it is necessary to have the command in conformance with the syntactic and semantic rules of the receiving LMS.

---

[4] This control is done by the **Command Handler** of the receiving END-SYSTEM.

2.1.9 LMS Alarm Handler

LMS Alarm Handler has the task to signal locally detected alarms to the CMS Alarm Handler for their solution.

2.1.10 LMS Diagnostic Handler

LMS Diagnostic Handler manages the running of the diagnostic programs on the local End-System. It is also able to retrieve information about diagnostic programs which are present on the End-System. So it supports the CMS Diagnostic Handler in providing a suitable diagnostic operator interface.

It has a null fuctionality in OSIRIDE release one.

2.1.11 LMS Data Collection Handler

LMS Data Collection Handler manages the Data Collections activation on the local End-System. This activation can be requested by the CMS operator through the CMS Data Collection Handler. Some Data Collections are supposed to be started without the explicit CMS operator request, i.e. gathering of accounting data etc..

In OSIRIDE release one only this type of Data Collection are available.

In any case the results of these Data Collection are stored in local files, by the LMS Data Collection Handler. The names of these result files must be known by the Data Collection Manager and/or by the CMS operator.

2.1.12 LMS Software Distribution Handler

LMS Software Distribution Handler provides the capability to receive the software distributed by the CMS Software Distribution Host Handler either through mail -"off-line sending"- or through the File Transfer facility -"on-line sending"- and enables the activation of the received software required by the CMS Software Distribution Controller Handler.

It has a null fuctionality in OSIRIDE release one.

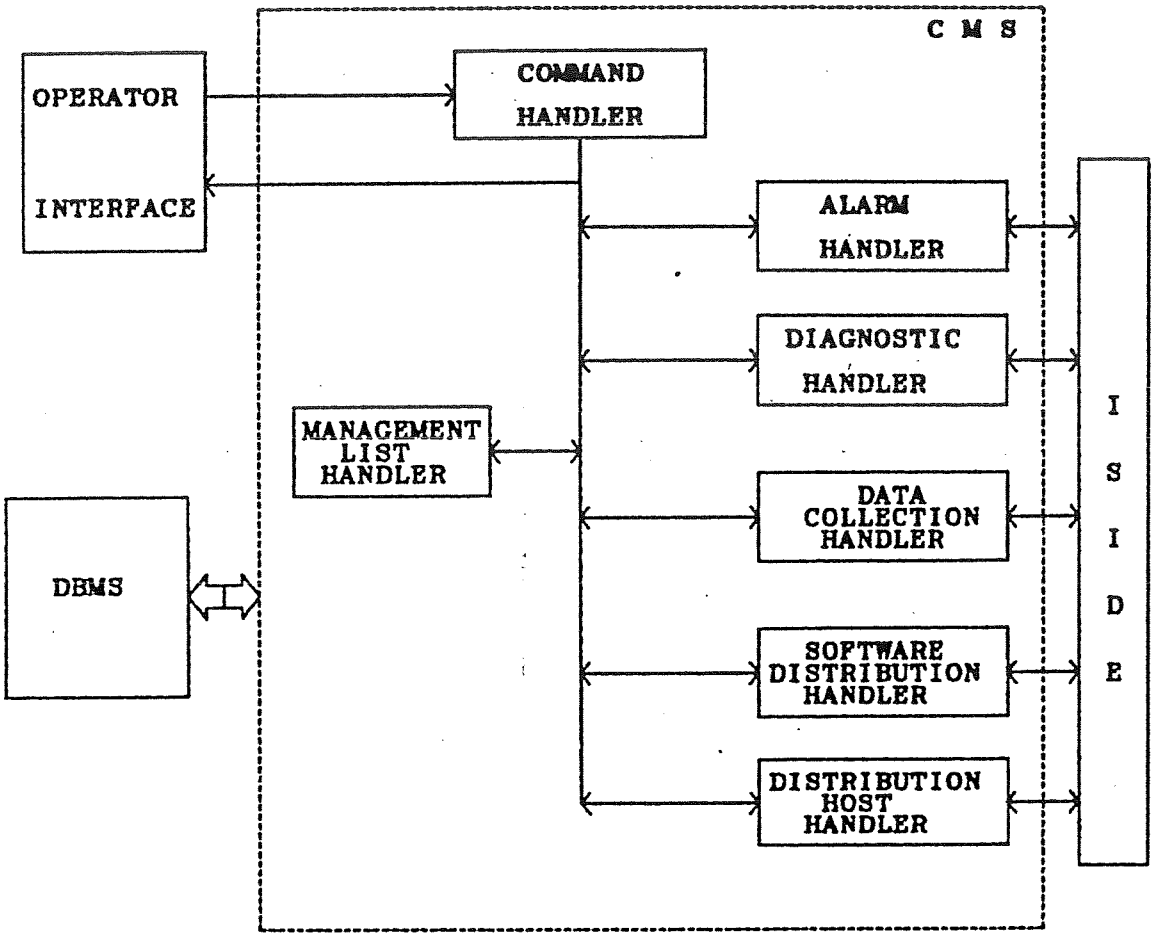

Fig. 3. CMS Architecture

In the following a brief description of each module is given. The Management List Handler, the Command Handler and the Operator Interface blocks have the **same role as in LMS**, therefore the description is not repeated here.

2.2.1 CMS Alarm Handler

CMS Alarm Handler receives the alarms communicated by the LMS Alarm Handler. It also provides the means by which a skilled team of technician may resolve the fault condition including the capability to retrieve information from the History Provider.
When a fault condition has been solved, the CMS Alarm Handler communicates information about the alarm and its solution to the History Provider.

2.2.2 CMS Diagnostic Handler

CMS Diagnostic Handler provides the operator with the capabilities to request for running of a diagnostic program on a specific End-System. The operator is assisted with a suitable interface to read the list of diagnostic programs present on an End-System and to follow the execution of one of them.

It has a null fuctionality in OSIRIDE release one.

2.2.3 CMS Data Collection Handler

CMS Data Collection Handler provides the operator with the capabilities to request a LMS Data Collection Handler for activating of a Data Collection on the related End-System.
When a CMS operator has activated a Data Collection on an End-System he must inform the Data Collection Manager about that, indicating also the result file names.

It has a null fuctionality in OSIRIDE release one.

2.2.4 CMS Software Distribution Controller Handler

CMS Software Distribution Controller Handler manages the whole software distribution operation. It asks CMS Software Distribution Host Handler to initiate the sending phase and controls the activation of the distributed software by dialoguing with the LMS Software Distribution Handlers.

It is important to point out that there must be only one End-System in the network in charge to support the CMS Software Distribution Controller Handler.

It has a null fuctionality in OSIRIDE release one.

2.2.5 CMS Software Distribution Host Handler

CMS Software Distribution Host Handler provides the sending in 'on-line' or 'off-line' mode of software to the destination End-System specified by the CMS Software Distribution Controller Handler. It is the module which accepts the Software Distribution request and performs the real sending of the software. It dialogs with the CMS Software Distribution Controller Handler to receive the request and to confirm the execution of it. To perform the "on-line" sending, it uses the File Transfer facility. The "off-line" sending is performed by the operator through the PTT mail.

It has a null fuctionality in OSIRIDE release one.

2.3 DAC ARCHITECTURE

DAC cooperates with CMS in two management functions:

- the Alarm Handling, in the role of Alarm History Provider (History Collector and Enquiry Facilities)
- the Data Collection, in the role of Data Collection Manager (the importer of collected data)

so it consists of two functional modules.

The dialog with CMS takes place via Management Protocols.

The DAC system uses the File Transfer Facility to "import" data collection files and store them in his file system. The error report data are imported through the dedicated protocol and they are stored in the DBMS. It also provides a user interface to construct statistical and charging Application.

The following figure outlines the DAC organisation. A brief description is also given.

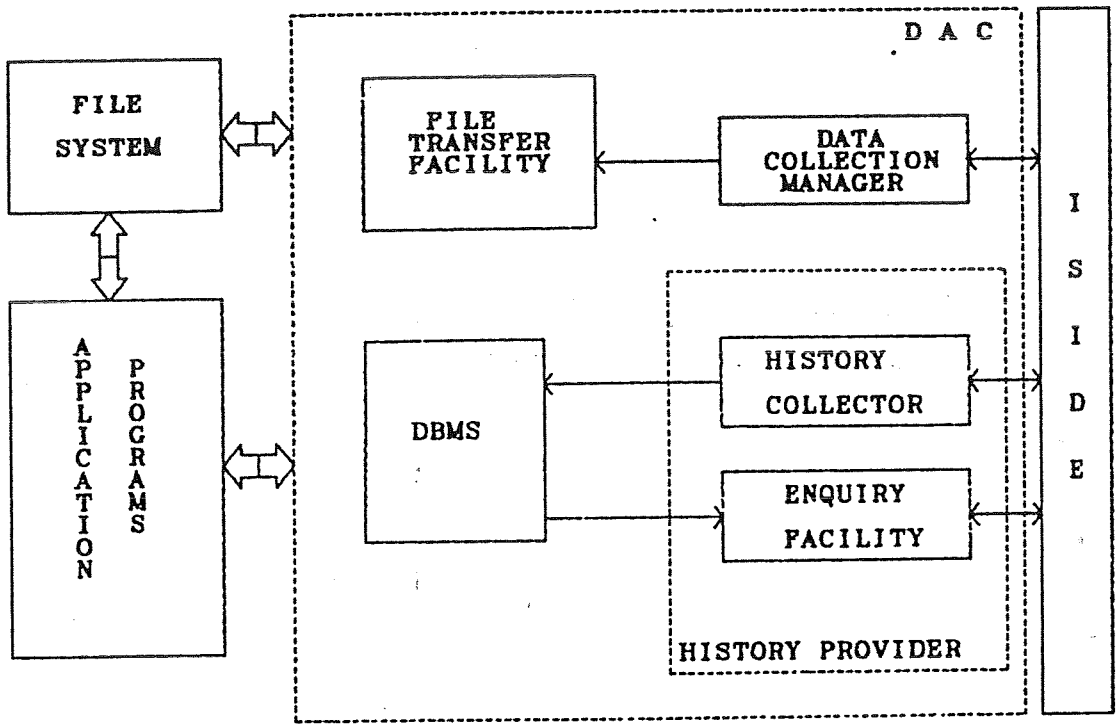

Fig. 4. DAC Architecture

2.3.1 History Provider

It is the module which receives (passively) error reports from the CMS Alarm Handler when the fault is recovered (History Collector submodule). The error reports are used both to provide for the history of the network and to satisfy the "enquiry" requests from the CMS Alarm Handlers (Enquiry Facility submodule).

2.3.2 Data Collection Manager

Data Collection Manager imports and stores the result files of Data Collections from the End-Systems.
The design of programs for data reduction and analysis is out of the scope of this document.

3.0 MANAGEMENT FACILITIES DETAILED DESCRIPTIONS

3.1 ALARM HANDLING STRUCTURE, FLOW AND USER VISIBILITY

Alarms are generated when faults are detected in hardware/software network components. They are collected locally by the LMS which selects, through a switching mechanism inside the Signal Handler, those to be notified by the LMS Alarm Handler to the CMS Alarm Handler. Alarms may also be stored in a local error logger and/or displayed at the operator console.

Two main types of alarms are required to be communicated to the CMS

- those issued because of a fault on a resource which needs an intervention to restore its normal working conditions
- those issued because attention is required on a resource where threshold levels have been overcome. This type of alarms are intended to be warnings to avoid further compromising of the normal working conditions.

LMS communicates alarms to CMS asking for cooperation. The CMS operator will generally send there a technician to solve the signalled fault. The intervention of technician can be invoked by using a suitable network service.
When a fault is solved the CMS Alarm Handler sends a report to the DAC (History Provider) to update the Alarm Data Base which keeps the history of the network components.

The Service Elements invoked for these purposes are:

M-ALARM-SEND

M-CALL-FIELD-SERVICE [5]

M-RESET-FIELD-SERVICE-CALL [5]

M-REPAIR-DONE

M-REPORT

M-ENQUIRY-DAC

and the following constraints apply:

each LMS Alarm Handler communicates to a unique CMS Alarm Handler

---

[5] not used in OSIRIDE release one

When a network component detects a fault it sends an alarm signal to the Signal Handler. This is to state that each component abled to alarm signalling is provided with the means to notify faults, in a specified format, to the Signal Handler. Transport and Session layers are provided with a Layer Interface for this; such interface is described in the related documents. An alarm signal must contain sufficient information to uniquely identify the originator network component and the type of alarm. The Signal Handler uses this information together with a predefined data structure containing threshold value associated to each couple alarm type/alarm originator to filter and dispatch the alarm.
Signal Handler may dispach alarm signals to

- local operator, through the operator interface,
- local files, through a dedicated interface,
- LMS Alarm Handler,
- some of them.

Alarms may also be originated by the local operator. For doing this he shall use an Operator Interface command. As for each LMS command, the Operator command is processed by the Command Handler for an authorization control. The Command Handler, when receiving this command, dispatches an alarm signal to the LMS Alarm Handler.

3.1.2 Alarm solution

LMS Alarm Handler, after receiving locally an alarm signal, asks the CMS Alarm Handler for ccoperation to solve the fault sending the M-SEND-ALARM Service Element. An alarm identifier is returned through the confirmation of this Service Element. This identifier univocally identifies the notified alarm both on LMS and on CMS sides.

The following figure describes the CMS modules involved in the repairing procedure, the interaction between them and with the LMS Alarm Handler. In the figure the main submodules constituting the CMS Alarm Handler are outlined.

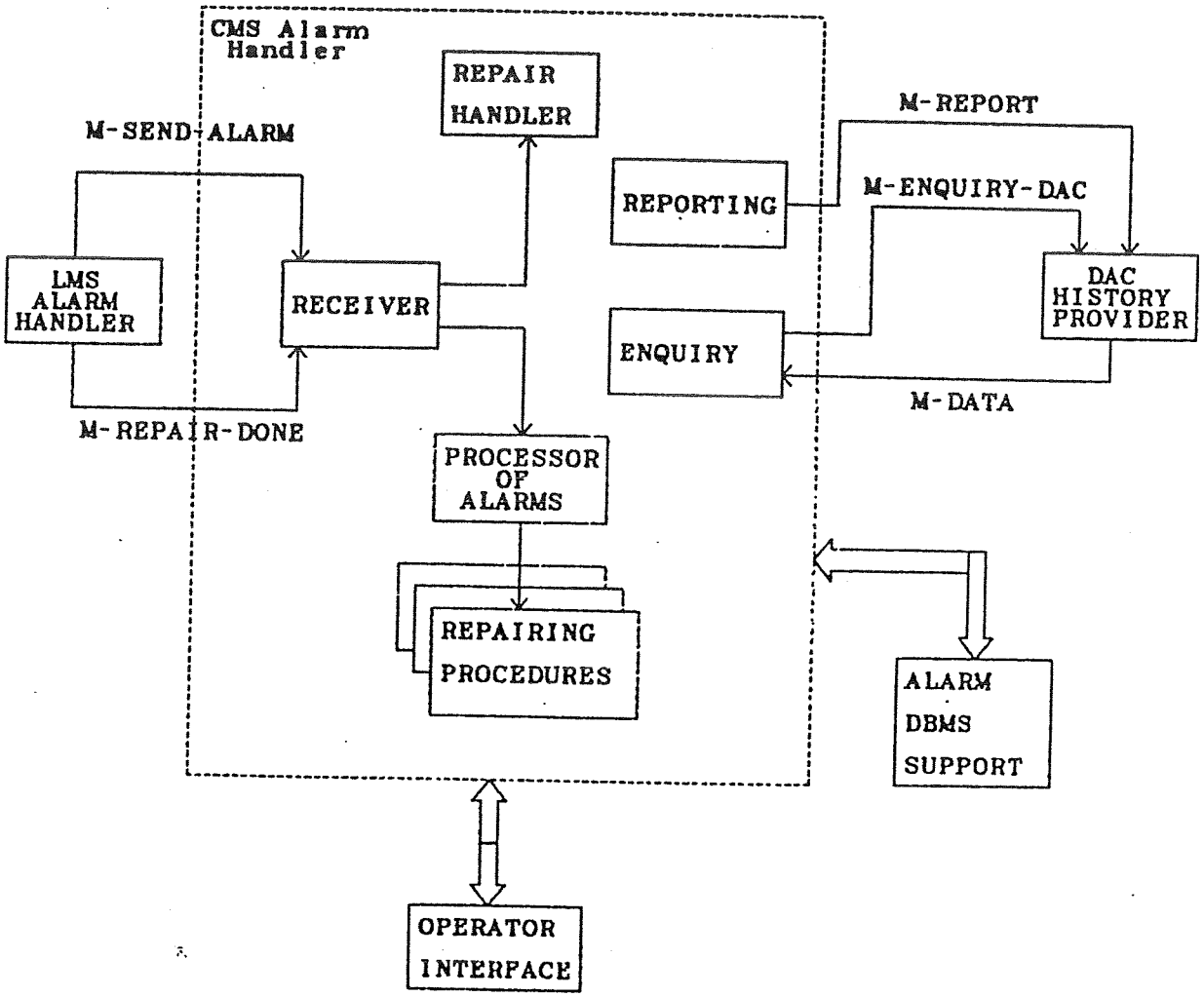

Fig. 6. Alarm solution and reporting schema

An incoming M-SEND-ALARM is processed by the Receiver submodule of the CMS Alarm Handler. It first assigns an identifier to the signalled alarm and return it back to the LMS Alarm Handler in the confirmation of the service.
After this the Receiver passes the alarm to the Processor of Alarms submodule. This submodule

- updates the CMS OPEN-ALARM-LIST, which is the list of pending alarms,
- signals the alarm to the CMS operator,
- activates a pre-defined appropriate repairing procedure if any (this is called "automatic" error handler).

The alarm can now be solved by the CMS operator or by a technician reaching the critic End-System for this purpose. This technician can be invoked by issuing the M-CALL-FIELD-SERVICE Service Element (the call can be reset through the M-RESET-FIELD-SERVICE-CALL Service Element)[6]. In the first case after having solved the fault condition, the operator updates the CMS OPEN-ALARM-LIST finally classifying the alarm as "CLOSED".

In the second case after having solved the fault condition the LMS operator signals this to the CMS Alarm Handler by means of an LMS Operator Interface command which invokes the M-REPAIR-DONE Service Element. As for each LMS command, the Operator Interface command is processed by the Command Handler for an authorization control. The Receiver submodule, when receiving such indication, asks the Repair Handler submodule to update the CMS OPEN-ALARM-LIST classifying the alarm as "AWAITING FOR OPERATOR CLOSURE".

3.1.3 Alarm reporting and enquiring

The CMS Alarm Handler contains a Report Sender submodule which can be started either automatically once over a predefined period of time or by an explicit CMS operator request. It scans the OPEN-ALARM-LIST and for each "CLOSED" alarms it:

- sends a report to the History Provider (DAC) containig information about the alarm and its solution through the M-REPORT Service Element and
- deletes the alarm from OPEN-ALARM-LIST.

The CMS Alarm Handler supplies the operator with the capability to enquire the DAC about alarms both related to a specified resource and occurred in a specified period through the Enquiry submodule.

---

[6] In OSIRIDE release one the Area Control Center plays both CMS and FS roles, therefore no Service Elements are needed to call tecnicians.

This is obtained by sending an M-ENQUIRY-DAC Service Element and receiving the related M-DATA which contains the data requested.

3.1.4 CMS operator visibility

For the alarm recovery, CMS operator is supplied, through the Operator Interface, with a set of tools.
These tools enable the operator to:

- display information about current alarms;
- retrieve information about the originator of the alarm,
- retrieve information about standard recovery procedure, if any for each type of alarm;
- follow the recovery progress of an alarm;
- close in orderly fashion an alarm after its solution.

3.2 DATA COLLECTION STRUCTURE, FLOW AND USER VISIBILITY

It has a reduced functionality in OSIRIDE release one.

The OSIRIDE NMS must be able to collect information about the operation on the network resources and to make this information available to various Applications for management purposes, e.g. statistics, accounting etc..
This information is gathered by accessing data available by resource managers and is imported by the Data Collection Manager.

In OSIRIDE Release one only the communication software has to be considered as a network resource in the data collection activity.

The Data Collection management function provides the capabilities

- to supervise the activation of data collection according to the CMS operator requirements and
- to import data collected.

Data Collection function also includes automatic activation (at IPL time) of measurement functions availabale on End-Systems to collect information on the use of resources. They don't need explicit activation from CMS. Their results are imported daily by DAC.
Only automatically activated Data Collections are supported in OSIRIDE release one .

To activate a Data Collection on a specific End-System, information must be exchanged between the CMS Data Collection Handler and the related LMS Data Collection Handler. After the activation of a Data Collection, the CMS Data Collection Handler informs the Data Collection Manager which imports directly data collected from that End-System through the File Transfer Facility.

The Services invoked for these purposes are

M-RUN-PROGRAM-POSTPONED

M-ABORT-PROGRAM

M-DC-MANAGER

and the following constraints apply

each LMS Data Collection Handler interacts with a unique CMS Data Collection Handler

a unique Data Collection Manager is present in the network.

The following figure shows the functional modules and their interaction in providing the Data Collection management function.

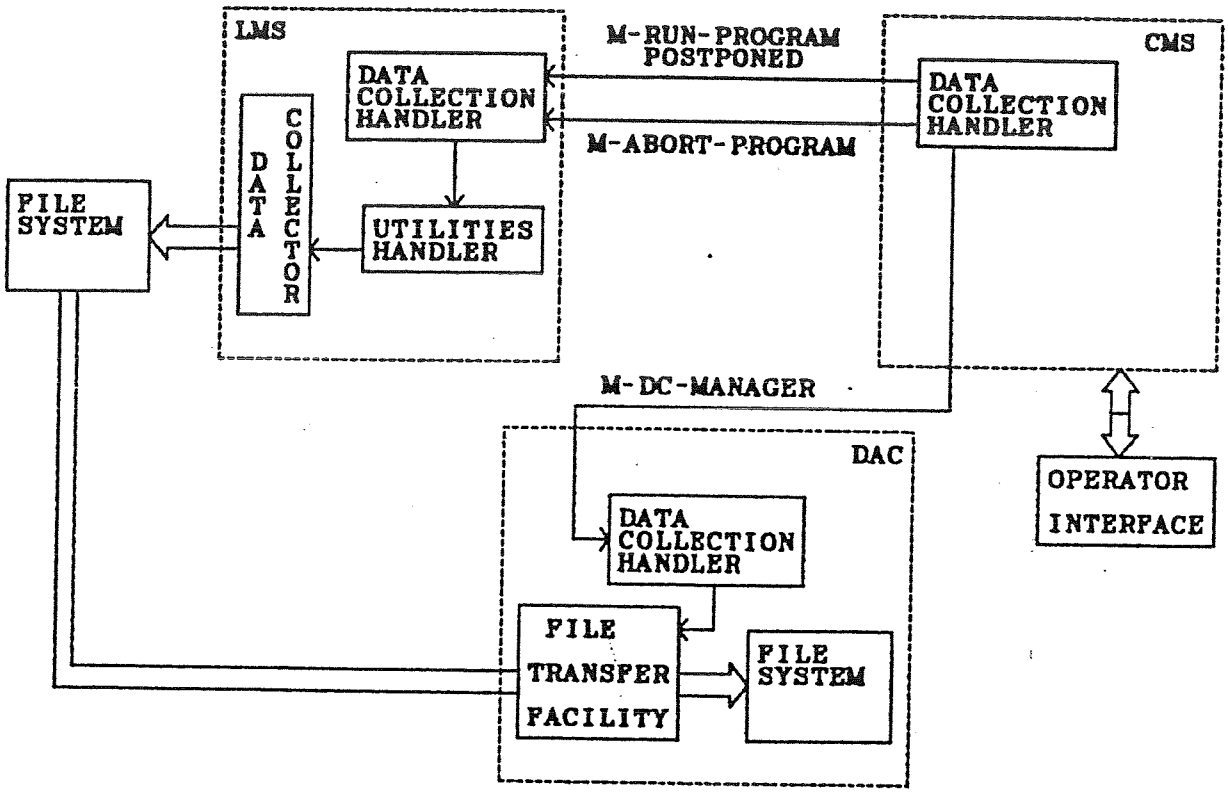

Fig. 7. Data Collection schema

3.2.1 Data Collection selection

The CMS Data Collection Handler provides the CMS operator with the means to get information about the Data Collections which are supposed to be available on a specific End-System.
The CMS Data Collection Handler retrieves this information directly from a personal data base; in fact there is no need to ask the LMS Data Collection Handler for this information, being the list of Data Collections always the same in each End-System.

3.2.2 Data Collection activation

The CMS operator can require the activation of a Data Collection on a specific End-System.
Accordingly the CMS Data Collection Handler signals such a request to the specific LMS Data Collection Handler through the M-RUN-PROGRAM-POSTPONED Service Element.

As a consequence of this the LMS Data Collection Handler first controls the correctness of the request (i.e. if the correspondent function exists and is really available).
If the control fails an error code is returned, otherwise it asks the Utility Program Handler to start that Data Collection and sends back the informations necessary to construct the name of the result file in the confirmation of the Service Element.

At this point the CMS Data Collection Handler, if the request has been accepted by the LMS Data Collection Handler, signals the event with the related information (e.g. the mask to construct the result file name, etc.) to the Data Collection Manager by means of the M-DC-MANAGER Service Element. So the Data Collection Manager is able to construct the result file name without knowing the file system syntax of the related End-System and can import the collected data.

3.2.3 Data Collection information retrieval

The Data Collection Manager has the task to import the files containing the results of the active data collection.

When receiving an M-DC-MANAGER Service Element, the Data Collection Manager constructs the result file name and decides when that file has to be retrieved from the related End-System, according to the

Start-date

End-date

Start-time

End-time

Deletion-date

parameter values specified in the Service Element.
Files are imported through the File Transfer Facility and are stored in its DEMS.

Some data collection are supposed to be activated without the explicit request of the CMS operator (e.g. accounting data). They are activated automatically on each End-System at IPL time.
The Data Collection Manager is able to daily retrieve the result files generated by this class of data collection programs because each End-System must generate the result file names according to a predefined rule.

The data collection programs definition is out of the scope of the document. Nevertheless it is important to point out that the result files produced must be constructed in a standard format in order to minimize the number of the DAC's reduction programs.

3.2.4 CMS operator visibility

The CMS Data Collection Handler provides the CMS Operator with the following capabilities in order to supervise the data collection activity.

- capability to provide the CMS operator with the means to activate a data collection on a specific End-System.
- capability to forcedly stop a previously activated Data Collection specifying if the already collected data must be deleted or not.
- capability to signal to the DAC the activation of a Data Collection with the information necessary to retrieve the result files.

3.3 TRANSPARENT TRANSFER COMMAND STRUCTURE, FLOW AND USER VISIBILITY

This function enables an authorized OSIRIDE user to request transparent transmission of a string of data (LMS Command) to a Command Handler residing on a remote End-System. The management Service Elements used for these purposes are:

M-COMMAND

M-DATA.

The command is sent by means of the M-COMMAND Service Element to the appropriate End-System.
If such command is syntactically correct and authorized[7] it will be executed as a locally given command.
The M-Command Service Element contains also the identifier of the source (OPERATOR or User-name); this information is used, together with the sending End-System name, by the destination Command Handler to perform the authorization control.

It has to be noted that no analysis of transmitted string of data is done at the sending LMS side, so it is necessary to have the command in conformance with the syntactic and semantic rules of the receiving LMS.

The "completion code" and all possible messages are sent back to the originator by means of the M-DATA Service Element.

3.4 NOTIFY STRUCTURE, FLOW AND USER VISIBILITY

The NOTIFY function enables an active Application to signal its availability to accept requests of connection to one or all the remote LMS. It also enables an Application executing a connection request on a non-active Application to ask the LMS to store this request until the arrival of the relative NOTIFY event. The re-execution of the connection request is the job of the LMS.

An Application may use the NOTIFY function to ask for a new attempt of all the pending incoming connection requests, or only those of a certain Application through the READY primitive (see ISIDE) using the options BROADCAST, SYSTEM AND USER.
If no connection request with this Application is pending, the NOTIFY Service Element is ignored[8].
An Application asking for a connection may specify, through the WAIT option (see ISIDE - CONNECT primitive) whether it wishes to wait until the receiver becomes active.

In this way no loop of the connection request of the Applications is needed, nor is it necessary to provide every End-System with special functions to activate the Applications for the connection requests. When applied only to the Applications already known to the LMS, such functions will reduce the flexibility required by the research environment of CSIRIDE.

---

[7] This check is performed by the Command Handler of the destination End-System.
[8] This is the reason why this Service Element is not confirmed.

The Service Element invoked for this purpose is:

M-NOTIFY

4.0 OSIRIDE RELEASE ONE MINIMUM REQUIREMENTS

This section describes the CSIRIDE release one management minimun requirement. It points out for each management function the supported subset, the related Service Elements and the rationale of the choices.
Alternative solutions are suggested for the services not selected.

4.1 ALARM HANDLING

This management function is of primary importance; OSIRIDE release one shall support the function described in "3.1 Alarm Handling structure, flow and user visibility" pag. 18.
Only two Service Elements are not supported in OSIRIDE release one:

- M-CALL-FIELD-SERVICE
- M-RESET-CALL-FIELD-SERVICE.

This is due to the fact that the CMS and FS roles are played by the same End-System, namely the Area Control Center.

4.2 DIAGNOSTIC HANDLING

A simple Diagnostic activity can be initially performed in a homogeneous environment. In fact the Area Control Center may activate the diagnostic tools locally available on each End-System by means of Transparent Command Transfer facility.

Results, if stored in files, can be imported by ACC using the File Transfer Facility.
Anyway, initially all the OSIRIDE End-Systems are supposed to have an operator which can supply a diagnostic activity. For this reason **no Diagnostic Service Elements** are supported in OSIRIDE release one.

4.3 DATA COLLECTION

Initially a subset of the function descrited in "3.2 Data Collection structure, flow and user visibility" pag. 24 is supported. This simplified data collection does **not need Service Elements.** In OSIRIDE release one Data Collections are supposed to be always active (i.e. started at the IPL of the network software).
The DAC has the task to daily import the result files whose name root is predefined i.e there is a table in the DAC containing the root of the data collection files of each End-System.

4.4 SOFTWARE DISTRIBUTION

Initially software distribution will be performed in a simple way by means of file transfers under the control of the End-Systems operators.
This simplified Software Distribution does not need any Service Elements.

4.5 NOTIFY

The Notify management function shall be provided.

4.6 TRANSPARENT TRANSFER COMMAND

The Transparent Transfer Command management function shall be provided.