Extending GKS to a distributed architecture.

R. Bettarini
G. Faconti
L. Moltedo

Rapporto interno C85-1

CNUCE- CNR- PISA

Extending GKS to a distributed architecture.

R. Bettarini (*), G. Faconti (*), L. Moltedo (**)

(*) CNR - Istituto CNUCE - Pisa

(**) CNR - Istituto Applicazioni Calcolo - Roma

Pisa, 12/7/1984

Extending GKS to a distributed architecture

1.0 INTRODUCTION

The Italian National Research Council (CNR) is a public institution that promotes research activities in many disciplines from mathematics to engineering, from physics to chemistry, and several other fields of application.

The growing demand by scientific and technical users for computing facilities has made the institution of CNR computer centers indispensable in geographical areas where users exceed a certain critical threshold.

The ever growing and diversifying demands from the field of research, the limitation of finantial resources, and the cost of hardware and software equipments make it difficult for one single computer center to provide the extensive range of facilities required by the different fields of application.

The interworking of several computers through a public data network allows for the integration of facilities offered by individual computing centers, with a consequent qualitative and quantitative enlargement of the overall computing facilities which could otherwise be impossible.

Since 1979, CNR has been experimenting on network services based on a packet switching distributed computer network called RPCNET

([Cane82]). The encouraging results of the experimentation of RPCNET and the emerging of standard protocols for the data exchange between systems led to the decision of adopting a network architecture which fully complies with the Open System Interconnection Reference Model ([OSIref],[ISOOSI]). At this purpose a projet called OSIRIDE has been started in order to develop and implement the new architecture ([Can84a],[Can84b]).

As the demand for graphics facilities to be available at the OSIRIDE computers is the strongest one inside the Institutes of CNR, the management of graphical resources is a major problem. In order to offer a uniform accessible set of graphical services to unsophisticated users, and uniform tools to professional staff for the development and the management of such services, the Graphical Gernel System (GKS) ([Bono82],[Hopg83],[ISOGKS]) will be adopted as the base system with the added capability, through the workstation concept, of running in a distributed architecture; that is, the possibility for any graphical application in any computer to access the graphical resources in any of the other computers of the network.

In this paper, some major problems encountered in the definition of the model of the distributed GKS are pointed out, together with the presentation of an extension of that model.

Extending GKS to a distributed architecture

In order to allow GKS to be considered for distribution over a computer network it must be described in further details; that is, a more refined layer model may be investigated.

Five layers can be identified from the functional description of the system as represented in Fig. 1.

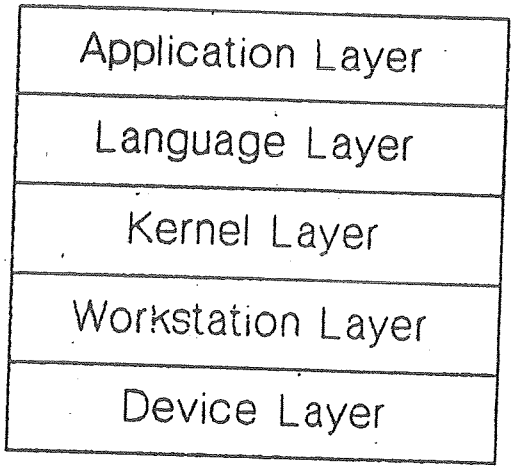

Fig. 1. Layer model of GKS

The **Application Layer** provides a means for the application processes to access the graphical environment. The interface to the next layer is defined by the language bindings.

The **Language Layer** provides a specific language syntactic support for the abstract GKS functions. It also performs conversion functions between the data types of that language and GKS data types. Its interface to the Kernel Layer is described by the functional specifications of the Graphical Kernel System.

The **Kernel Layer** is defined by the GKS functionality. All the GKS functions start to execute in this layer but might be terminated in layers of low crder. In particular, the normalization transformations belonging to output primitives and their inverse for locator and stroke input are performed in this layer. It also handles Workstation Independent Segment Storage processing, and GKS Description Table and State List management. An interface called Workstation Interface ([Beno84]), exists between Kernel and Workstation Layers separating the device independent part of the system from the workstation dependecies. As this separation is actually described only from a logical point of view, the interface should be yet clearly defined as if it coincides to a large degree with the Computer Graphics Metafile functionality ([ISOCGM]).

The **Workstation Layer** performs clipping and workstation transformation functions. It also handles segment operations, and Workstation Description Table and State List management.

The **Device Layer** provides several services which depend on the device capabilities in order to translate the GKS functions into sequences of functions that can be interpreted by physical devices.

The set of functions contained in the last two layers may greatly differ between implementations as it depends on the physical device capabilities. In fact, the availability of microprocessors and memory components at relatively low cost, allows for the implementability of great amounts of compunting

power into a graphics workstation; the local computing capabilities can then range from a microcoded system to one providing full operating system support.

The definition of Application and Language Layers is not strictly related with the objectives presented in this paper and will not longer be discussed. In the followings the term application will address an application program including those layers.

We will note moreover mention another way, beside the language binding, by which the application program may access the GKS Kernel functionality, that is the character coded binding ([Mayn84]). The character coded binding of GKS is defined as a sequence of 7-bits or 8-bits character codes representing messages sent to and from between an application program and another process which handles the graphic functionality of GKS.

2.2 HANDLING OF REMOTE WORKSTATIONS

The layer model discussed in the previous paragraph allows for the distribution of GKS functionality in a variety of system architectures as the workstation interface makes it possible to

exchange data between the Kernel and the Workstation Layers of different implementations ([Beno84]).

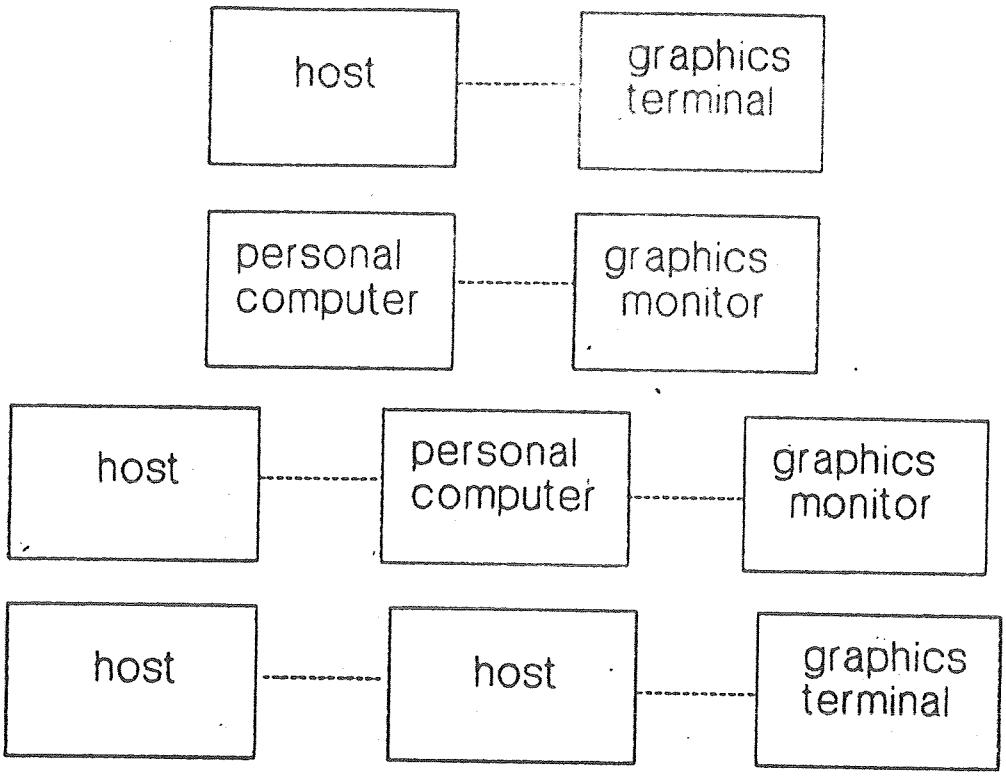

Fig. 2. Graphics system configurations

Several configurations can be exploited other than those represented in Fig. 2 ([Wagg84]); however severe limitations to the usability of the system are found when the network becomes more complex.

Three major limitations can be pointed out from the functional specifications of GKS:

1. The Kernel must know at opening time the number and the list of available workstation types.

This constraint imposes that all the devices eligible for use by an application in the network must be known in advance and consequently the network architecture must be frozen at a certain time. This means that any change in the configuration of a node needs to be reflected in all the nodes of the network.

2. During execution of an output primitive the Kernel sends data to any active workstation in the network one at a time.

The mapping between the connection identifier of a workstation and the physical address of the corresponding physical device is done outside from GKS. The Kernel itself keeps only track of the names of the active workstations and cannot be acknowledged if multiple workstations refer to the same node in the network. As a consequence the same data are sent multiple times with different identifiers on the same connection line degrading performances of the network.

3. The deferral state of a workstation is affected by interactions occurring in other workstations if it is set to BNIG.

This constraint imposes that the Kernel inquires the deferral state of any active workstations when an interaction with a logical input device gets underway on any workstation in order to made visible the effects of the stacked functions if any.

3.0 GKS IN AN OPEN SYSTEM INTERCONNECTION ENVIRONMENT

The computer network of CNR (OSIRIDE) allows for the interworking of different computers from several firms and provides services to clustered small computers through a Front End Processor. Subnetworks (as Local Area Networks) may also be interconnected in such a way that they appear as part of one Global Network.

In the first OSIRIDE implementation, the Global Network will include ITAPAC (the Italian Public Packet Switching Network) and Ethernet subnetworks as shown in Fig. 3 ([Can84a]).

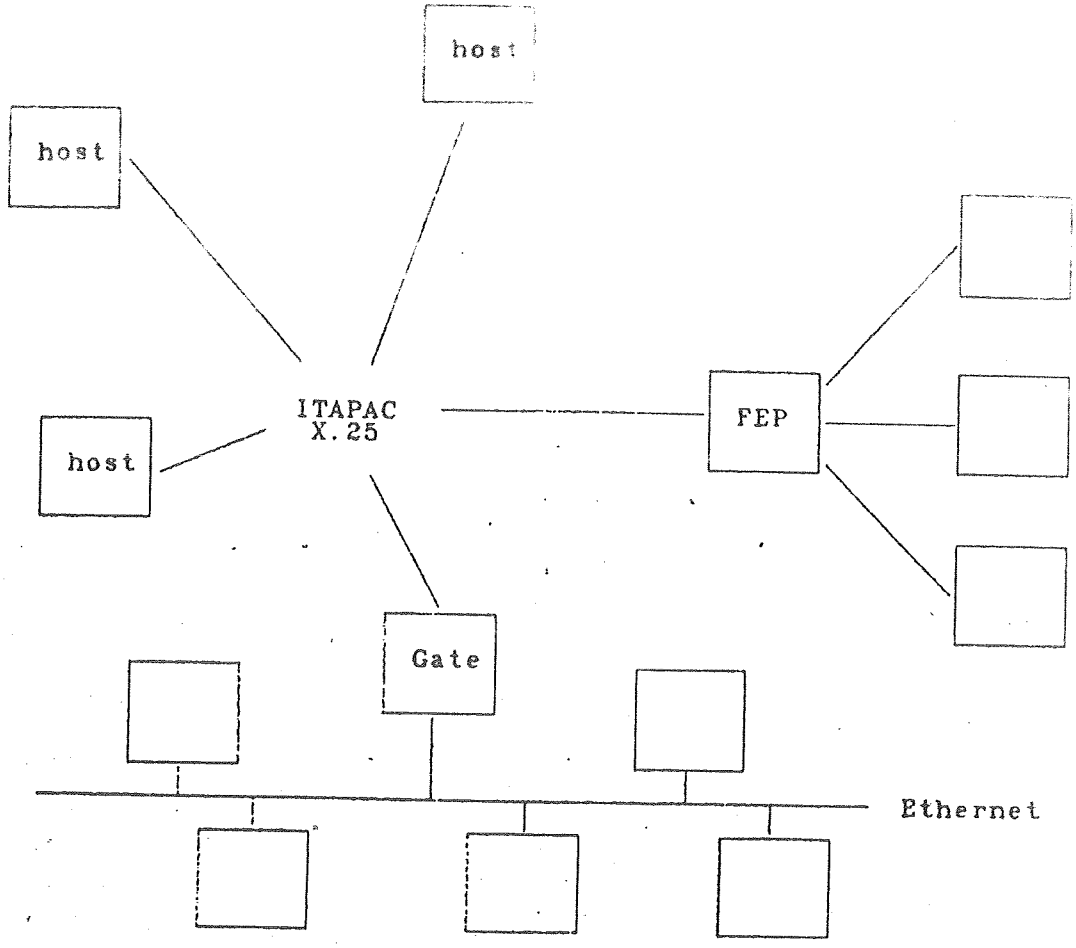

Fig. 3. OSIRIDE environment

A GKS implementation is planned to be available at each computing center connected to ITAPAC through OSIRIDE.

3.1 THE LAYER MODEL OF THE DISTRIBUTED GRAPHICAL KERNEL SYSTEM

In order to allow an application running in one of the mentioned computing centers to use a graphics device connected to another computing center, a model for a distributed GKS has been defined which is slightly different that the one presented in paragraph "2.1 The layer model of the Graphical Kernel System".

A new layer called the **Workstation Manager** has been inserted between the Kernel and the Workstation Layers as shown in Fig. 4.

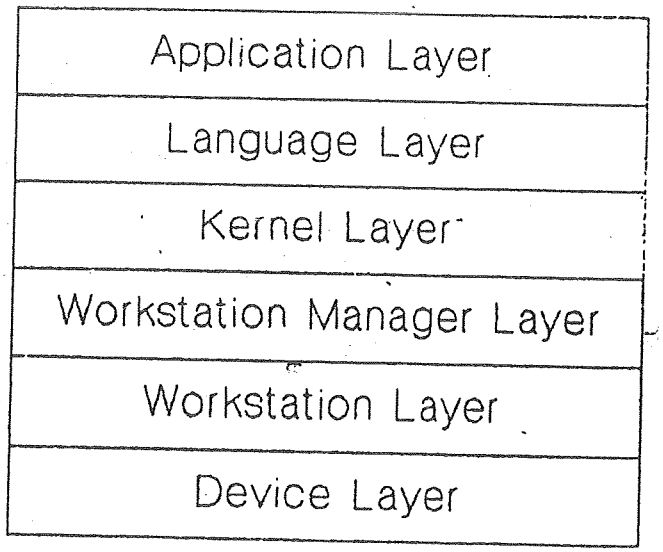

Fig. 4. Layer model of distributed GKS

As a consequence the functionality of the Workstation Interface is made available to the Workstation Layer through the Workstation Manager Layer.

Extending GKS to a distributed architecture

A possible distribution of the GKS layers is shown in Fig. 5.

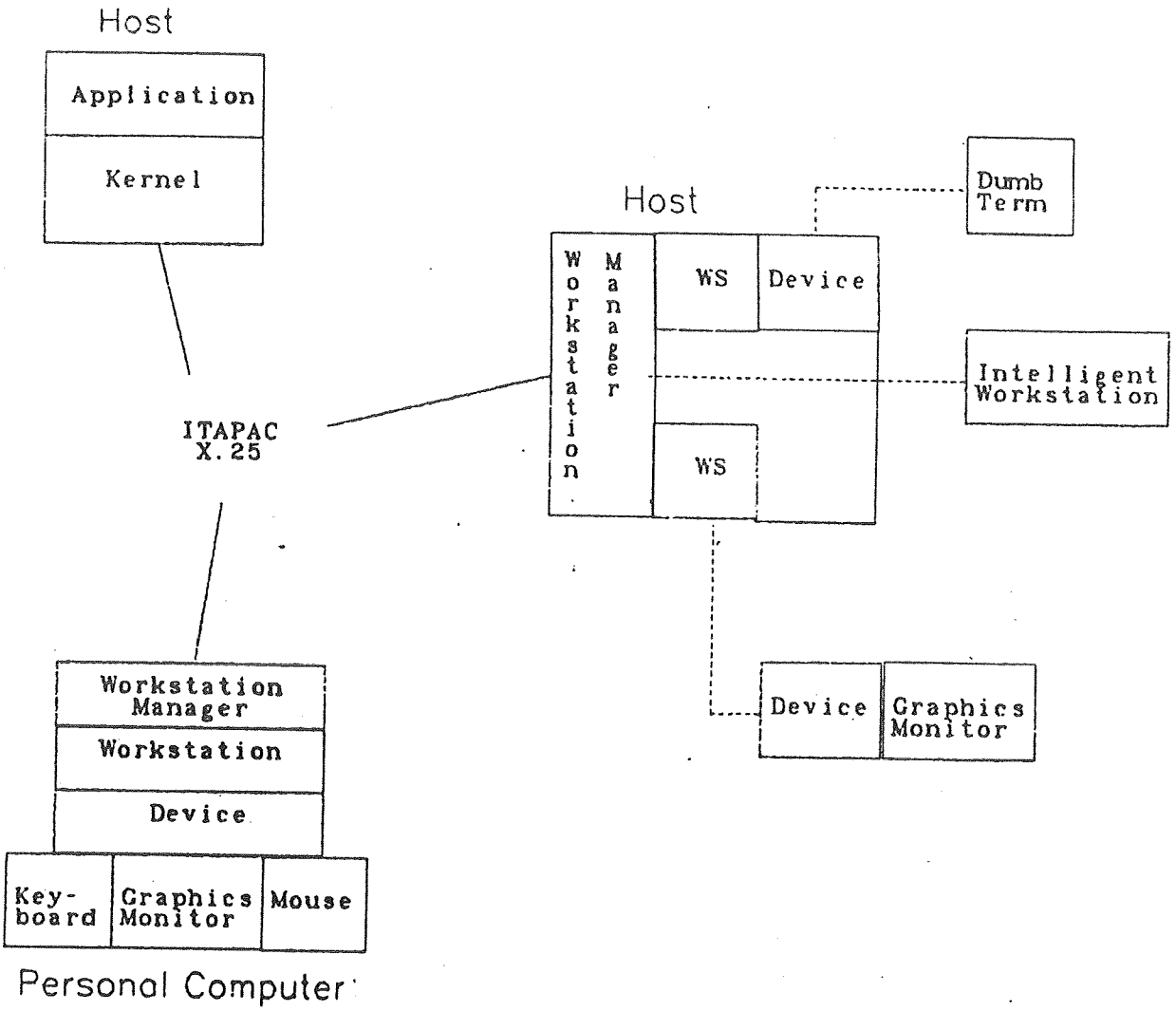

Fig. 5. Distribution of GKS layers in the network

3.2 COMMUNICATION BETWEEN KERNEL AND WORKSTATION MANAGER

Inside the Kernel and the Workstation Manager a set of logical functions, relevant to the network, can be identified which deals with communication aspects. This set of functions is seen by the network as the **application entity.**

The functionality of the application entities is to allow Kernel/Workstation Manager communications and to minimize the accesses to the network services.

The Kernel application entity may recognize that multiple workstations belong to the same Workstation Manager and performs only one data transfer of output primitives together with the workstation identifiers for those workstations.

The Workstation Manager application entity allows for the transfer of one stream of input events from multiple workstations on Kernel request as it will be explained in paragraph "3.3 Modifications to GKS data structure".

3.3 MODIFICATIONS TO GKS DATA STRUCTURE

The internal functioning of the model of the distributed GKS requires a few modifications to the GKS data structure.

3.3.1 GKS Description Table

As the Kernel Layer can't handle information such as the number and the list of available workstation types, the corresponding entries are removed from the GKS Description Table; a Node Description Table is defined instead, carrying on those information in the Workstation Manager at a given node of the computer network.

This modification implies that any check on the workstation type is deferred to the Workstation Manager during execution of the OPEN_WORKSTATION function.
Similarly the INQUIRE_LIST_OF_AVAILABLE_WORKSTATION_TYPES function requires the indroduction of a new parameter identifying the node which is the object of the query. To maintain the compatibility with applications written for the standard GKS, it is defined as an optional parameter: when it is missing an implicit reference is made to the local computer configuration of graphics devices.

3.3.2 GKS State List

A second modification is introduced in the GKS State List. In order to provide the Kernel with the capability of handling interaction from a logical input device when workstations exist with deferral state set to BNIG, a new entry is defined in the table containing the names of the workstation which are in this state of deferral.

3.3.3 Event Input Queue

The last modification introduced in the GKS data structure affects the input level c of the system and it becomes active when a logical input device is set to mode EVENT.

The EVENT input enables a specific operating environment in which the operator interacts asynchronously with the system by generating entries in an EVENT queue handled by the Kernel through an independent process. Entries are removed from the queue by the AWAIT_EVENT function and then interpreted by the GET<input class> functions.

As to minimize the flow of data through the computer network, the EVENT input is implemented by enabling the workstations to handle their own event queue and to notify the Workstation Manager

of their state. Any time the Kernel empties its event input queue, the Workstation Managers are requested to send all the input generated at nodes if any; the events are registered in the Kernel event queue according to their creation date ([Beno84]). Input queue overflow error is handled at Kernel level in the standard way, and at Workstation level by the workstation program itself which inhibits any further input until more room is made available.

4.0 CONCLUSIONS

The growing demand for graphics facilities from the community of scientific and technical users imposed the Italian National Research Council to face the problem of standardizing the services of computer graphics offered by its computing centers.

The obvious answear has been the adoption of the GKS standard as the base system for the development of graphics applications.

The limitation of finantial sesources and the cost of hardware and software equipments, as well as communications efficiency led to the decision of developing a distributed model of GKS to be implemented on a computer network.

Although the functionality of GKS is specified in details, a large degree of freedom is left in defining the internals of the system so that it has been possible to define a distributed model which maintains the standard interface toward the application programs.
As a consequence of the special attention dedicated to the application interface, any graphics program developed for the standard GKS will run on the distributed environment witn no modification.

REFERENCES AND BIBLIOGRAPHY

[Gued76] Guedj RA, Tucker H, 'Methodology in Computer Graphics', Proceedings IFIP WG5.2 Workshop SEILLAC I, North Holland, 1976

[OSIref] Data Processing, Open System Interconnection, Basic Reference Model, ISO/TC 97/SC 16 N-537.

[Bono82] Bono P, Encarnacao J, Hopgood FRA, 'GKS - The first graphics standard', IEEE Computer Graphics and Applications, vol 2 no 5.

[Fole82] Fóley JD, Van Dam A, Fundamentals of interactive computer graphics, Addison-Wesley.

[Cane82] Caneschi F, Lenzini L, Menchi C, 'Status and evolution of the RPCNET network in an operational environment', Proceedings of ICCC82, North Holland.

[Hopg83] Hopgood FRA, Duce DA, Gallop JR, Sutcliff DC, Introduction to the Graphical Kernel System, Academic Press.

[Stra83] Straayer D, 'Computer Graphics Standards: Where they are', Tekniques, vol 7 no 3.

[ISOOSI] Draft International Standard, Information Processing, Open System Interconnection, Functional Description, ISO/TC 97/SC 16, ISO/DIS 8348.

[ISOGKS] Draft International Standard, Information Processing, Graphical Kernel System (GKS), Functional Description, ISO/TC 97/SC 5, ISO/DIS 7942

[Beno84] Structure of a Workstation Interface, Position Paper for WG 2 meeting in Benodet, DIN-NI-5.9.4, ISO/TC 97/SC 5/WG 2 - N 238.

[Wagg84] Waggoner NC, Tucker C, Nelson JC, 'NOVA*GKS, A Distributed Implementation of the Graphical Kernel System', SIGGRAPH'84 Conference Proceedings, vol 18 no 3.

[ISOCGM] Draft International Standard, Information Processing, Computer Graphics, Metafile for the Storage and Transfer of Picture Description Information, Functional Specification, ISO/TC 97/SC 21, ISO/DIS 8632.

[Enca84] Encarnacao J, 'Interface and Data Transfer Formats in Computer Graphics Systems', EUROGRAPHICS'84.

[Can84a] Caneschi F, Gregori E, Lenzini L, Menchi C, Zucchelli E, Implementing the OSI standards: the OSIRIDE Project, CNUCE.

[Can84b] Caneschi F, ISIDE, or the OSIRIDE Access Method, OSIRIDE/FDT/02.

[Mayn84] Maynard JH, A (revised) character coded binding of GKS, Committee correspondence, ANSI/X3H3/84-115 R 1