OPEN SYSTEMS INTERCONNECTION COMMUNICATION ARCHITECTURE

H.J. Burkhardt

Rapporteurs: G.D. Parrington
S. Pfleger

Open Systems Interconnection Communications Architecture

H. J. Burkhardt
Gesellschaft für Mathematik und Datenverarbeitung
Rheinstrasse 75, D 6100 Darmstadt, FRG

This paper presents the basic ideas of the OSI Reference Model, a communication architecture developed within international standardization bodies during the past years.

1. Objective of OSI

The objective of Open Systems Interconnection (OSI) is to enable heterogenous systems - i. e. systems differing in origin and technology - to communicate.

Communication is understood as cooperation comprising the transfer of data (from a source in one system to a sink in another system) as well as the processing of transferred data according to the application specific purpose of the cooperation.

Communication among heterogenous systems is needed in a scenario as depicted in Fig. 1.
In this szenario heterogenous data terminal equipments and networks are embedded in organization and communication structures among human beings supporting them in the distributed processing and storing of data and providing for them open access to a wide variety of DP-services offered all over the world.

In such a szenario with a large number of heterogenous systems, which have to communicate, the conventional approach of adapting heterogenous systems on a bilateral basis (s. Fig. 2) is no longer economical.

In the OSI-approach, heterogenous systems, therefore, get the capability to cooperate by observing standardized communicational behaviour (s. Fig. 3).

Heterogenous systems which follow the OSI-approach are called real open systems.

To develop real open systems, three major steps are necessary envolving various parties and requiring a common understanding among them:

- Specification of communicational behaviour by international standardization bodies
- Implementations of the standardized behaviour by the various manufacturers on their real systems
- Test of the implementations on conformity with the underlying standards by authorized test centers.

This common understanding does not a priori exist. To establish it, is an important and ongoing task of the standardization work on OSI which began in 1977 when the ISO/TC 97/SC 16 Open Systems Interconnection was founded.

2. The OSI-Reference Model (OSI-RM)

2.1 Objective of the OSI-Reference Model

To specify communicational behaviour for OSI means to describe formally cooperation in distributed systems. Descriptions must be independent of technology and implementation - due to the assumed heterogenity of real systems - but have to be nevertheless complete, error free, consistent and unambiguously understandable.
As basis for their work, the standardization bodies responsible for OSI within ISO and CCITT have developed a communication architecture.

This architecture defines functions needed, and relationships that must exit in communication, and thereby determines standards needed to develop compatible equipment. Therefore, this architecture is called 'The Reference Model for Open Systems Interconnection (OSI-RM)'.

The objective of having a Reference Model is to establish a general framework for the standardization work by identifing standards necessary for OSI and relating them to one another, by identifying any need to improve such standards in the light of experience and by developing further standards - if possible independently but in a coordinated fashion - and finally, maintaining them in an orderly manner and thus, creating a complete set of standards.
The reference model is not an implementation specification - it is far to coarse to precisely define communication among open systems. Such definition is in the scope of specific standards determining communication services and protocols in detail. For this reason the Reference Model ist not suitable to check any actual implementation for 'openness'.

The OSI-RM is the result of an abstraction process (s. Fig. 4). The reality is a projection into the future when heterogenous systems can cooperate supporting man-man communication, man-computer communication and computer-computer-communication.
The model abstracts from the heterogenity of real systems and the variety of real sources and sinks. It reflects their distribution and the different functional categories of cooperation, to which they should be able by describing cooperation among abstract instances located in abstract systems.

It is important to have this relationships between model and reality in mind since they establish the meaning or "semantics" of a model.

2.2 Structuring principles of the OSI-RM

To emphasize the abstraction process leading from reality to the OSI-RM we introduce the concepts of cuts to derive the structure of the OSI-RM.

A cut separates an inner world from an outer world and defines discrete interaction points, which are the only points at which the inner world and the outer world may influence each other (see Fig. 5). This influence on each other takes place by means of discrete elementary operations, called interactions.
The inner world and the outer world of a cut may be structured by means of other cuts (see Fig. 6). Cuts are used for:

1. identifying the units which are able to interact with each other and their interaction points,
2. relating the interaction points of interacting units, and
3. defining meaningful interactions at these interaction points.

Cuts are the means to completely hide the internal structure of units, make them "black boxes", and thus enforce to define the interactions between these units in terms of externally visible behaviour at their interaction points.

The communication architecture is based on three kinds of cuts, namely (1) system cuts, (2) service cuts, (3) protocol cuts.

System cuts are used to achieve a topological decomposition of the real world. They identify individual abstract systems as being representatives of those physical components of the real world hosting individual pairwise communication activites.

Service cuts are used to achieve a functional decomposition of individual pairwise communication activities. They identify functional layers.

Protocol cuts are based on system cuts and service cuts. They identify protocol entities which have to cooperate according to a protocol to fulfill the functionality of a layer.

2.2.1 System cuts

System cuts define open systems as abstract units able to communicate with other open systems.

Open systems may arbitrarily be defined by applying system cuts to the real world consisting of user stations, central units, switching devices, transmission lines, etc. (see Fig. 7). Thus, the communication architecture does not say anything about which components of the real world are to be put together by a system cut so as to form a system. This decision has consequences in so far as any system must show the same externally observable behaviour, independent of its particular physical components or structure.

Any system cut defines at least one interaction point. This interaction point is the point of the system cut where it is penetrated by the physical transmission device. By performing a system cut in the real world we replace this part of the real world by its interaction point and its characteristics (see Fig. 8). Note that this allows physical components to belong simultaneously to different systems.

This interaction point has an address that gives the locality of the whole system (i.e. of the part of the real world it represents). This is all that the communication architecture says about the locality of systems. In total, a system cut abstracts from the real inner world of a system but maintains from the real world the locality and characteristics of the interaction point.

System cuts generate two different kinds of systems (1) end systems, and (2) intermediate systems.
End systems are systems which host "application entities". Application entities are the logical units among which communication may take place. Communication among application entities can have various purposes and hence various contents. An application entity in the Reference Model is only defined by the type of communication(s) it is capable of achieving; or, in other words, about what subject it can 'talk'. An application entity is thus a conscious abstraction behind which hides a human user at a terminal or just as well a FORTRAN program residing in a data processor and managing a distributed data base, or a process control program.

Intermediate systems are systems without application entities serving to interconnect end systems for the transfer of data.

End systems and intermediate systems are interconnected by physical transmission media.

Systems of the Reference Model should not be equated with components of the real world such as user data stations, data processors, data transmission node etc. The term system represents a similar abstraction as the term application entity does. Behind an end system may hide a single user data station equally as well as a cluster of them, a single processor or a whole multiprocessor configuration.

Intermediate systems in the communication architecture represent all resources, common to all end systems, available for transmission of information; this includes all data transmission and data switching equipments.
In the real world, end systems and transit systems may overlap, although they are disjoint in the architecture. For example, some computer system may primarily be used for implementing a part of an intermediate system while at the same time containing application entities (for network management applications for example), and therefore implement an end system as well.

Each application entity is located in one endsystem and each endsystem contains at least one application entity, i. e. if we denote by A the set of application entities and by E the set of endsystems, then a localisation function L exist such that $E = L(A)$ holds true.

The OSI-RM is a model for two party communication between application entities residing in different endsystems. This means that transmission media are the only resources which are assumed to be in common between communicating application entities; there are no other common resources such as common memory, common processor etc. which may facilitate communication within real systems. With respect to one communication act each application is either the initator or the responder, i. e. plays either an active or a passive role.

Each application entity may be able to play in their communication with other application entities an active role or a passive role or both. If we denote by A_a the set of application entities capable to play an active role and by A_p the set of application entities capable to play a passive role, then A_a and A_p are both subsets of A, not necessarily disjoint; the union of A_a and A_p constitutes A.

In each individual communication an application entity a_a playing an active role and residing in one endsystem e_a is paired with an application entity a_p playing a passive role and residing in another endsystem e_p. Therefore each individual communication can be denoted by a tupel (a_a, a_p) with $a_a \varepsilon A_a$, $a_p \varepsilon A_p$, $e_a = L(a_a) \neq e_p \varepsilon L(a_p)$.

Having in mind the OSI-objective of systems compatibility, we consider these tupels as elements of the set $(C \subseteq A_a \times A_p)$ which is characterized by the pairing of an active and a passive role.
This consideration applied to a topological structure with individual systems and application entities (as shown in Fig. 8) leads to a structure with types (i. e.playing the same role) of systems and application entities (as shown in Fig. 9).

2.2.2 Service cuts

By means of system cuts the systems are identified which should become able to communicate.

The purpose of service and protocol cuts is to define a virtual structure for systems, thus determining their communication behaviour. This virtual structure for the inner world must not be confused with the real inner world of the systems. The explicit purpose of the system cut is to abstract from this real inner world. The only purpose of the virtual system structure is to describe the communications which they must be able to maintain. This virtual structure is the same for all systems, whereas the real inner world of system may be different from system to system (see Fig. 7). In particular, this virtual

structure for a system does not directly refer to any operating system function (of the realizing computer system). But it does require that the system's internal hardware/software components involved in communication activities - whatever they may be - externally show a communication behaviour as if the virtual structure were really implemented.

A service cut is applied on the configuration of a set of systems shown in Fig. 9 and defines a service provider extending over system boundaries as counterpart to a set of service users (see Fig. 10). Service users and service provider interact across so called "service access points". Each service access point is associated to exactly one service user. Each service access point belongs to exactly one systems. The elementary interactions between service provider and service user are described by "service primitives". The task of the service provider is to perform exactly those execution sequences of service primitives at the various service access points that are prescribed in the service specification for the service being considered. The service primitives are merely conceptual in nature, they are part of a service description technique. There is no immediate relation between them and "interface primitives" (as one would design them for a concrete implementation). A service specification given in this technique describes only global services. It describes no services the execution of which does only affect one of the two service users, because they have no "communication value".

As stated above, a cut makes an abstraction from the real inner world of the cut. In case of a service cut this means that we abstract from the fact that actually providing the service at two service access points requires appropriate interactions between the two systems to which the service access points belong. From the point of view of a service user, the service provider is a single abstract machine. Thus, all details of the topological distribution of the systems involved are removed. All that is left is a set of service access points, anyone of which (or its associated unique service user) is identifiable by means of a service access point address.

Service cuts serve to achieve a functional decomposition of communication activities between application entities in different systems. This functional decomposition establishes a separation of concerns in communication activities,

and thus drastically improves their understandability; it is a multiple step procedure.
In a first step it is differentiated between "content components" of communication which are specific to each individual communication, and the "general components" which prepare and support these content components of communication; the general components are part of each communication independent of content. The content components require a common, a priori, understanding about how the exchanged information is to be interpreted and processed; therefore, they can only be understood by application entities of a particular application. Such content components are handled by application protocols. These protocols depend on application and different applications have their specific protocols (e. g. File Transfer, Remote Job Entry).

In subsequent steps the general components, present in each communication, are further partioned by applying service cuts.
This leads to a hierarchy of communication services (see Fig. 11).
A hierarchically higher communication service includes all hierarchically lower communication services.
In this manner, function layers result (see Fig. 11). These function layers extend, similarly as communication services do, over system boundaries. What such layer must be able to do is determined by the functional difference between the next higher communication service and the next lower communication service.

2.2.3 Protocol cuts

Service cuts define communication services. The hierarchy of communication services defines functional layers as the cooperation capability which must be added to a lower communication service to provide the next higher communication service.

But Service cuts leave undetermined how two communicating systems must interact to provide the functionality of a layer. Protocol cuts fill this gap left open by service cuts and systems cuts. Protocol cuts define the protocol entities which are responsible for running protocols correctly, thus actually achieving the service to be provided (see Fig. 12).

If protocol entities are not located on the top or bottom layer of the architecture they have interaction points with a service user from the next higher layer, with the next lower service provider - interaction points of both these kinds are service access points - and with one (or several) protocol entities on the same layer. Protocol entities interact with the service user above with each other and with the service provider below, in such a way that they enhance the service used by that communication oriented semantic aspect that distinguishes it from the service provided.

The purpose of the protocols is to describe the details of these interactions between protocol entities and services in order to achieve this objective.

Thus, a protocol is a common behavioural convention between two entities of one layer in different systems. It is defined by completely and precisely determining all possible interactions of both protocol entities with their surrounding environments (through their interaction points described above). In general, a protocol description consists of two parts, each of which describes one protocol entity (including the enumeration and the use of the service primitives used, and the service primitives provided, as well as the protocol elements by means of which the information exchange between the cooperating protocol entities is performed.
The protocol elements are not only defined in their meaning - as it is the case for service primitives - but they are determined also syntactically (which is a matter of local concern, only, in a service specification).

Applied on each layer resulting from the hierarchy of communication services, protocol cuts leed to a hierarchy of protocol entities and protocols (see Fig. 13).

It is important that the hierarchy of entities be not misunderstood as a rule for real system implementations. Its purpose is only to provide a virtual structure for a system helping to describe its externally visible behaviour in communication activities. This communication behaviour becomes manifest through a set of protocols by means of which any global communication is performed and for expressing the meaning of which the entities interpreting them are required.

2.3 The layers of the OSI-RM

The OSI-RM defines seven layers which can be grouped into three application oriented layers on the top and four transport oriented layers on the bottom (see Fig. 15).
The boundary between this two groups is formed by the Transport Service (see Fig. 14).
The Transport Service enables Transport Service users in different systems to exchange data.
Transport Service Users in the OSI-RM can best be compared with persons in the human communication. What the Transport Service does is to let one TS-user hear what another TS-user has said to him.

The physical layer as the lowest transport-oriented layer serves to transmit streams of bits over transmission media between systems. Thereby, transmission errors may occur.
The data link layer has to recognize and correct transmission errors and has to provide for transmission of data, free of errors and loss.

The network layer has the task to route data through intermediate systems from one endsystem to another endsystem.

At least, the Transport Layer has to route data from a Transport Service user in one endsystem to a Transport Service user in another endsystem, possible closing any gap between the transport service asked for by the transport service user and the available network service.

Each of the transport oriented layers of the OSI-RM defines a cooperation functionality between systems related with the data transfer, the syntax and semantics of which can be completely defined.

In contrast to that, the application oriented layers define the functional prerequisites that persons, speak Transport Service Users, which can hear each other, can cooperate since they understand each other.

In this sense, each Transport Service User is substructured into an application entity, a presentation entity and a session entity, thus constituting the application layer, the presentation layer and the session layer.

The application layer is the highest layer and performs the processing of data exchanged, it operates on the meaning of data; it can more vividly be described as the layer of 'thinking in the same notions'.

Its immediately underlying layer is the presentation layer which operates on the data structure; again, it can be described as the layer 'speaking the same language'.

Next follows the session layer which takes care of ordered opening, conducting and terminating of communication; it can also be described as the layer of 'obeying formal manners'.

The presentation layer and the session layer can by characterized as offering a set of language elements which must be semantically enriched within each application by combining them with application protocol elements specific for that application.

3. Structuring general distributed applications

The present ISO Reference Model for Open System Interconnection, IS 7498, supports only two party communication between application entities residing in different endsystems and leaves open how it can be used to model distributed applications containing more than two parties.

3.1 The notion of 'distributed application'

In general, a 'distributed application' may be defined as a cooperation of application entities as shown in Fig. 16, where more than two application entities participate. The attribute 'distributed' indicates that the application entities forming a distributed application communicate via communication channels and not via a common memory. Therefore, explicit means are

necessary to coordinate their cooperation and to interrelate their local processing states to a global state of cooperation. As a rule, in a distributed application the individual application entities have different functions which mean that a set of different application protocols have to be used.

To coordinate the cooperation between any two application entities in a distributed application, the Reference Model provides the session. In a distributed application, several sessions may take place sequentially or concurrently among which there exist logical relationships. In fact, a distributed application is characterized by having sessions logically related. However, the Reference Model does not provide means to relate individual sessions, and each session takes place independently from any other session. The need to relate individual sessions to enable the cooperation among application entities must, therefore, be satisfied by the application entities themselves; they must provide the logical relationship among sessions "in session".

Interactions among application entities, according to an application protocol, must be interpreted as a mutual request to provide a service. Whereby the application protocol defines this service in its semantic contents, or in other words, at its purely terminological level.
All the necessary terms of a particular application are reflected in the different application protocol elements. However, since application protocol elements cannot actually be communicated directly, but may only be communicated via the presentation service, each application protocol element must be mapped on to a presentation service element.

3.2 The notion of 'application services' and 'application relays'.

Architecturally, an application entity capable to provide an entire requested service must be regarded as a special case. As a rule, assistance of further application entities is needed to provide the complete requested service:
An inquiry of a distributed database by a user is an example of such a case. The user may be regarded as an application entity which serves as a database enquirer and which communicates, according to a special application protocol, with a designated application entity which is a part of the distributed data-

base. Further, there are cases which require that a series of consecutive communications has to be made to satisfy an inquiry. For such cases the representative of the services of the distributed database is introduced, over which several differing application protocols may be used to provide the inquiry service.

Keeping this example in mind, the application service may be defined as a service of a distributed application which provides this service to the user via a designated application entity of the distributed application (see Fig. 17).

The respresentative of an application service (designated application entity) appears, thereafter, as application entity belonging concurrently to two distributed applications: in the first, together with the application entities which are represented externally, it forms one distributed application, and in the second, together with the application entity which acts as the user of the application service, it forms another distributed application. Thus, a representative must be able to establish logical relationships among different application protocols which in their semantic contents may be correspondingly subdivided between the application service user protocol and the application service provider protocols.

Finnally, it seems necessary to be able to differentiate, within the application layer, between application entities in their role as representatives of an application service of a distributed application, as described above, and those application entities which function as application relay entities, or together with other application entities, as distributed application relays between pairs of other application entities. As an example of this kind of application relay entity, or distributed application relay, an application may serve which acts as a mailbox between two application entities, A and B of Fig. 18.

4. Summary

The Reference Model of the open systems interconnection is a general model for communication. It forms a basis for development of service standards which in turn represent requirements for the development of protocol standards. Only the latter are implementable, i. e. they result in products which can communicate with each other. Openess of products is always dependent on adherence to a hierarchy of protocol standards.

Although designed for two party communication, the OSI-RM provides the elementary constructs to model distributed applications envolving n parties.

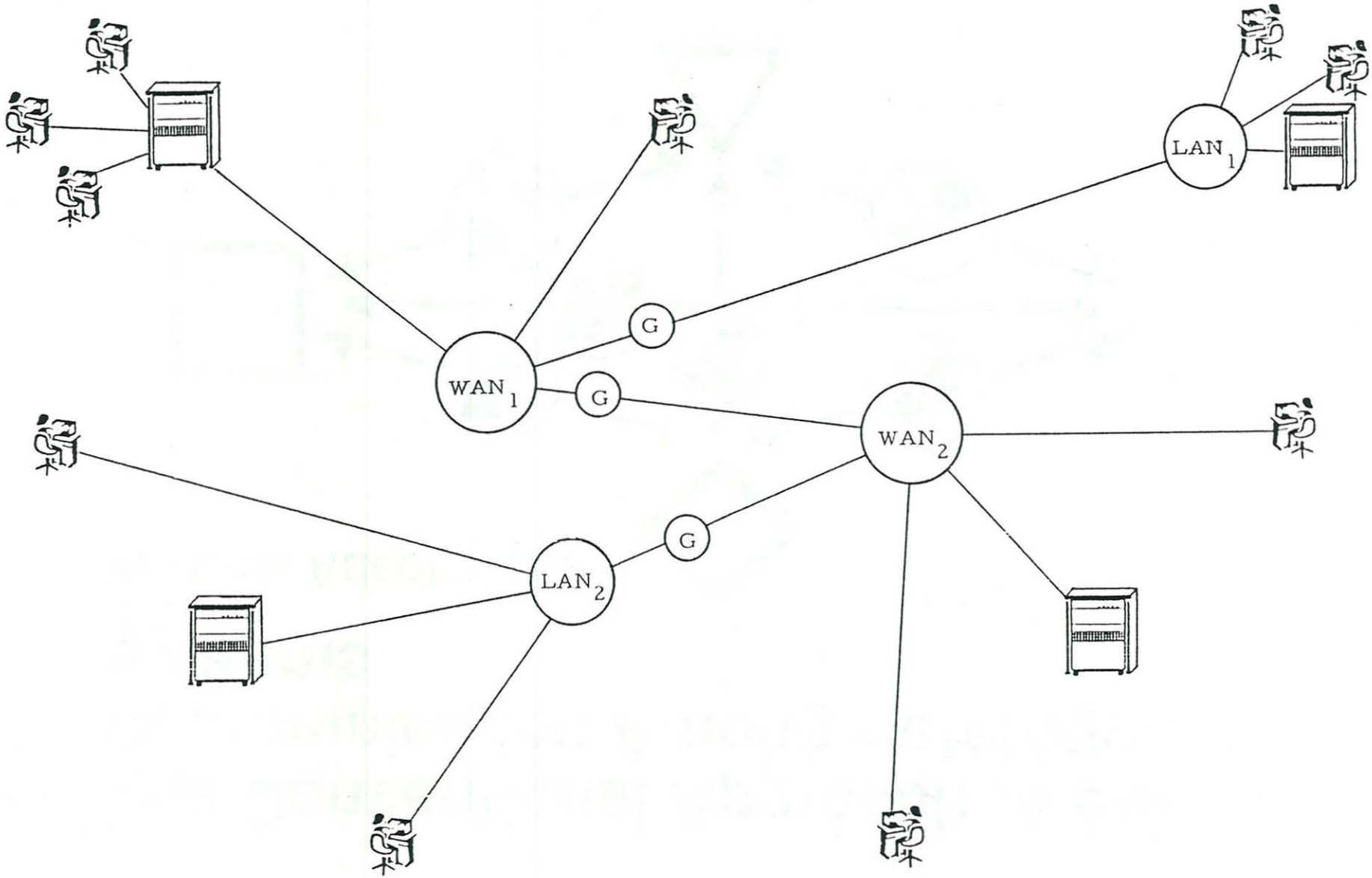


FIG. 1: REAL WORLD SCENARIO FOR OSI
(WAN : WIDE AREA NETWORK;
LAN : LOCAL AREA NETWORK;
G : GATEWAY)

FIG. 2: **The Conventional Approach to Enable Communication Among Heterogenous Systems**

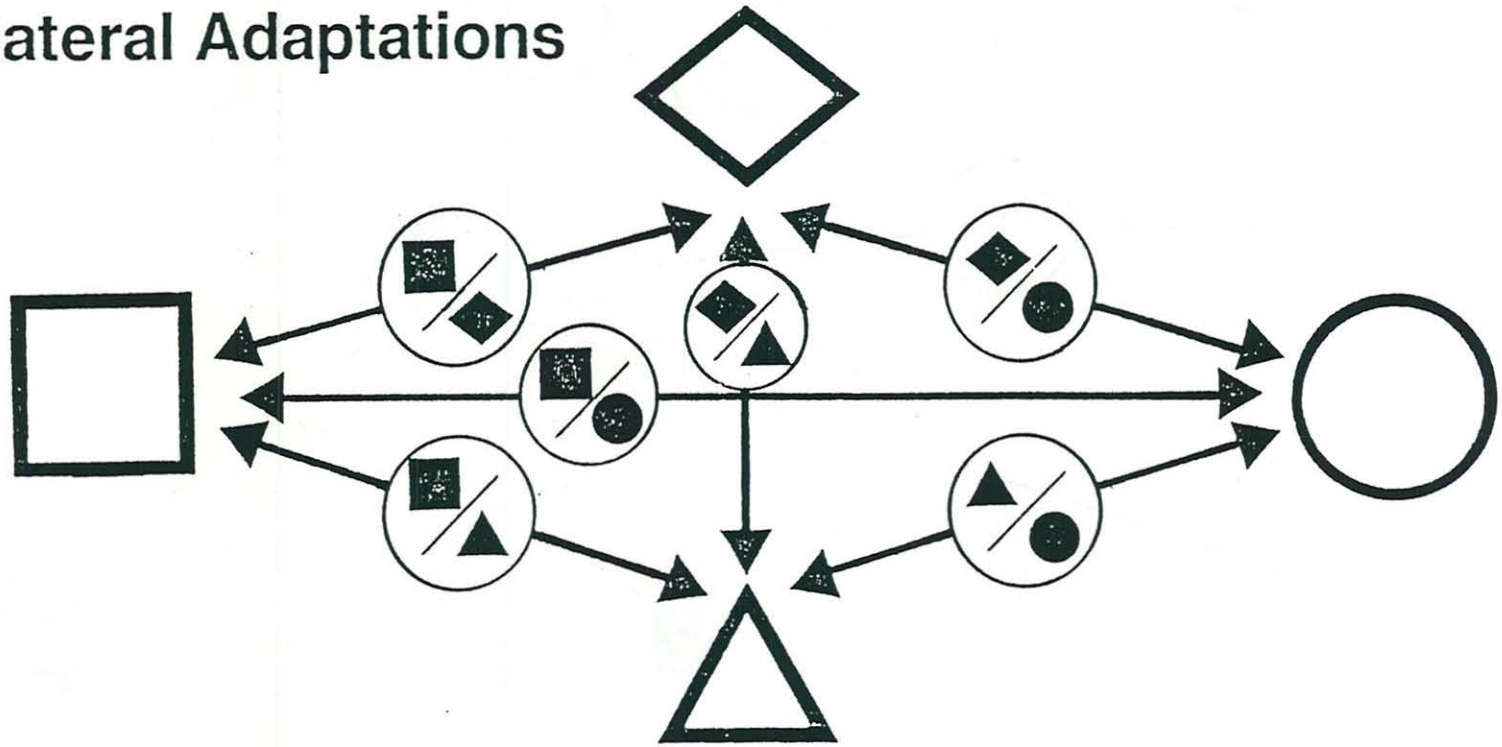
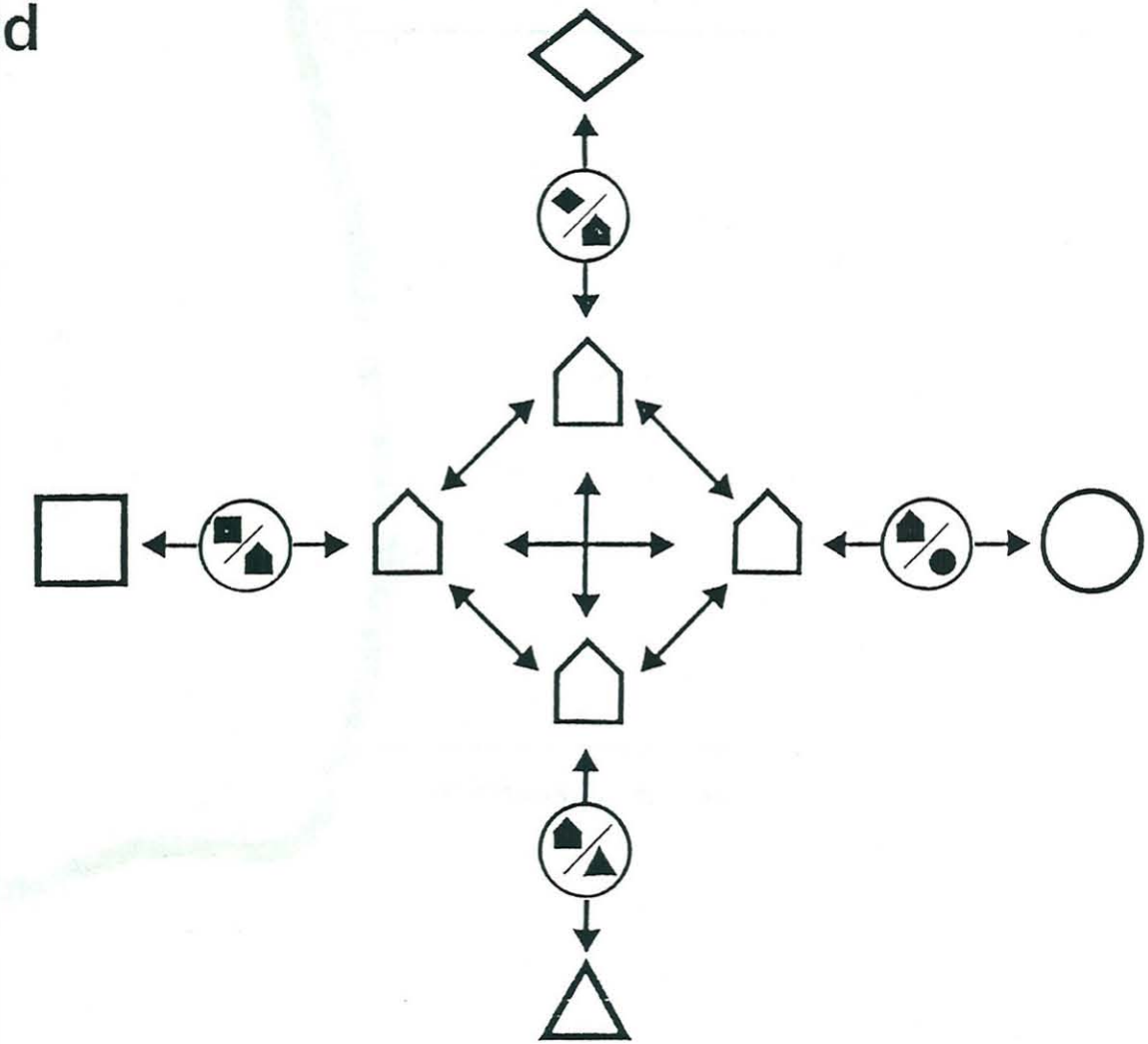


FIG. 3: **The OSI - Approach to Make Real Heterogenous Systems Open**

Observing standardized communicational behaviour



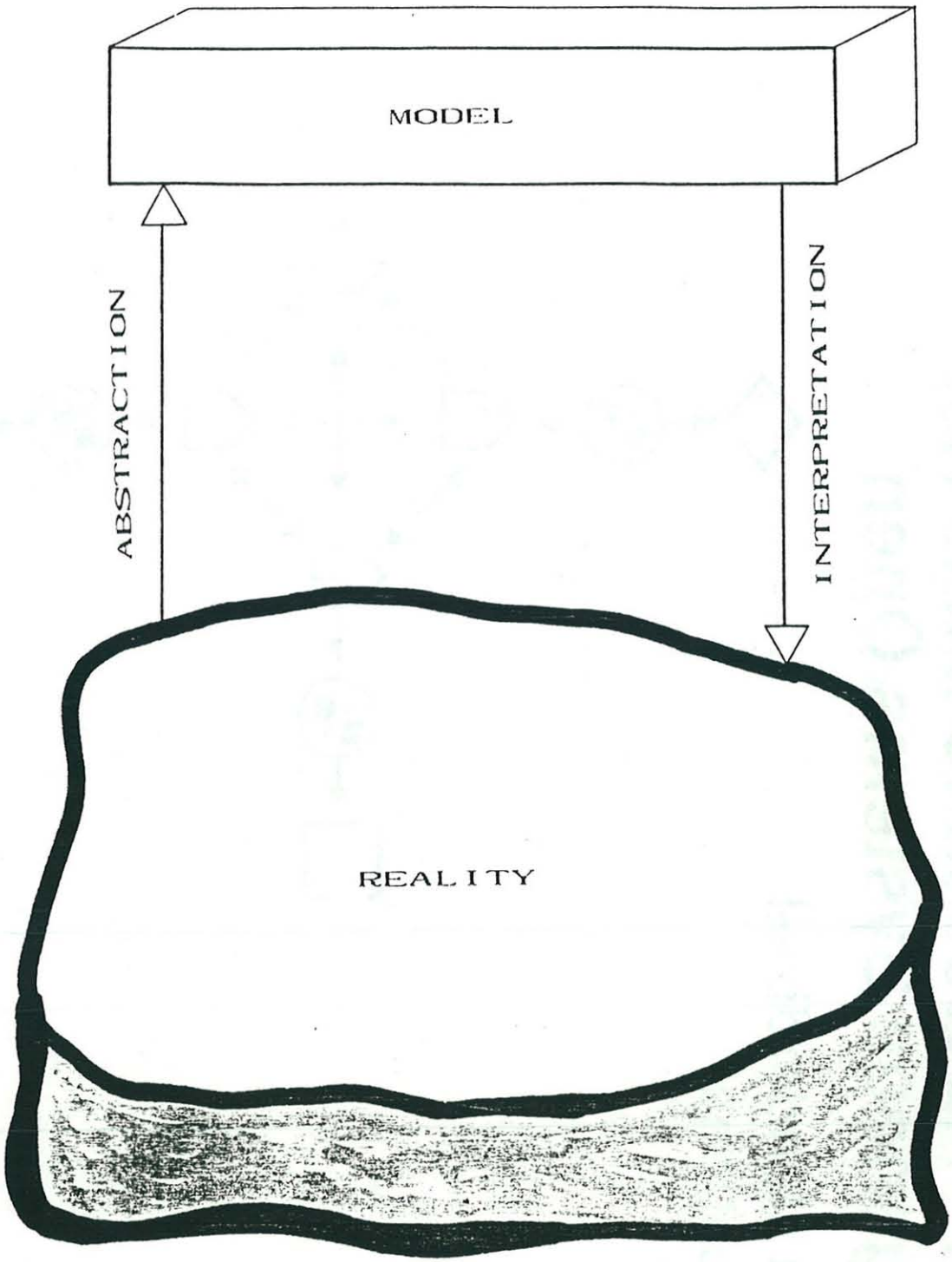


FIG. 4: RELATIONSHIP BETWEEN REALITY AND MODEL

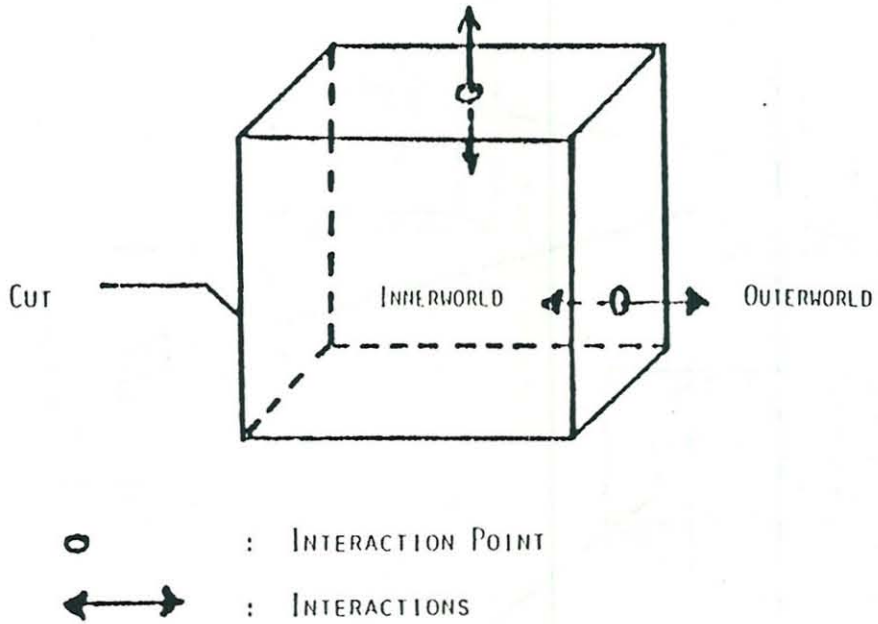


FIG. 5: THE CONCEPT OF A CUT

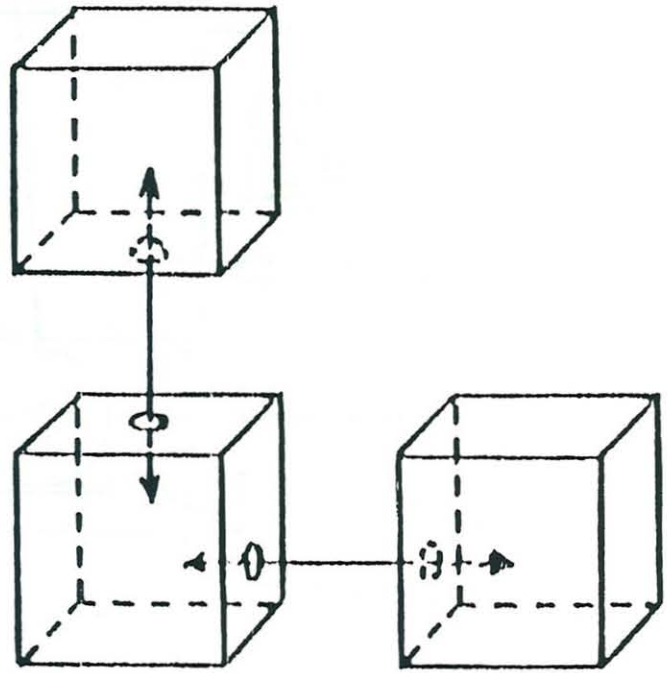


FIG. 6: RELATING INTERACTION POINTS OF DIFFEREN CUTS

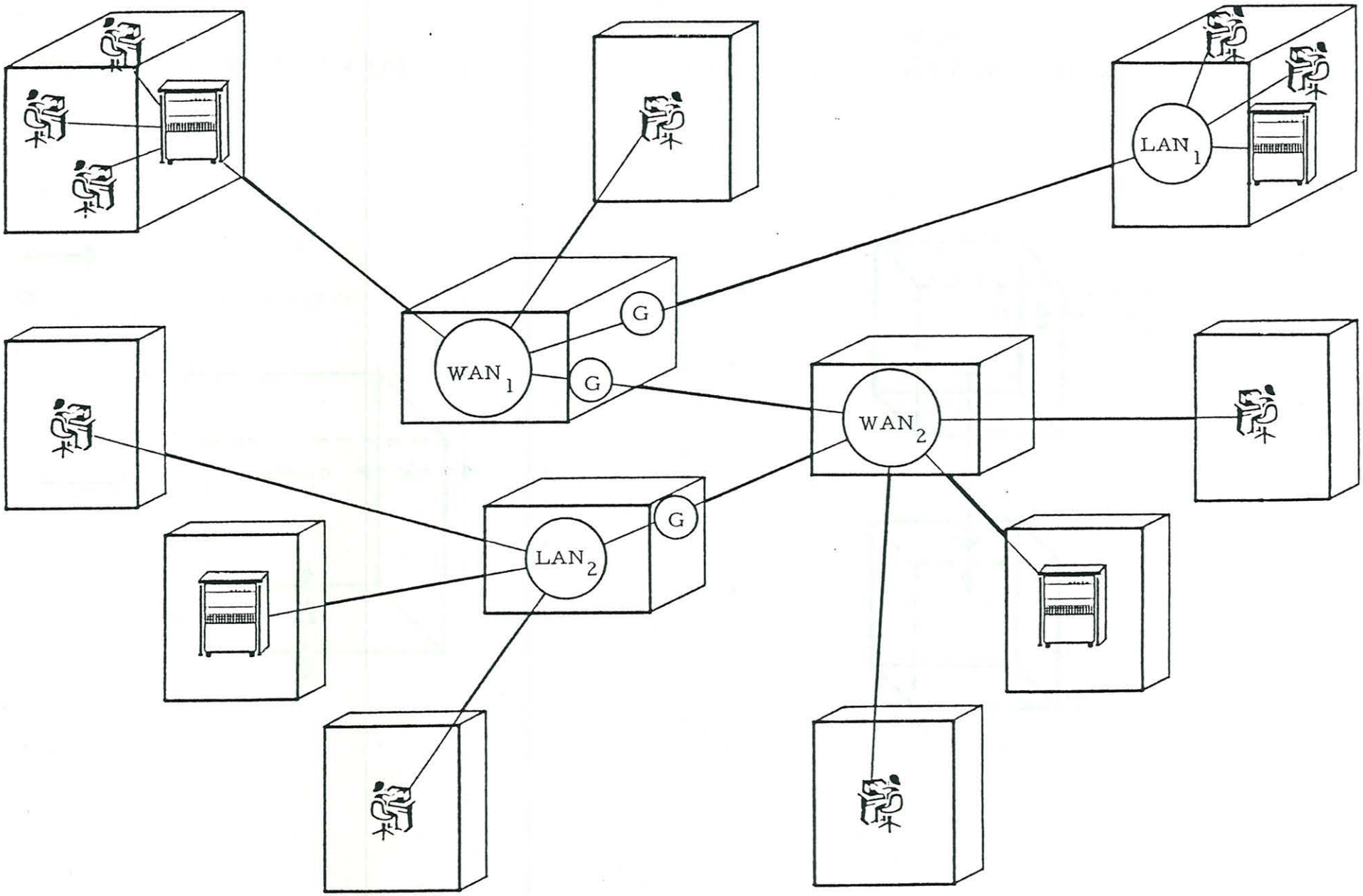


FIG. 7: APPLYING SYSTEM CUTS ONTO A REAL WORLD SCENARIO

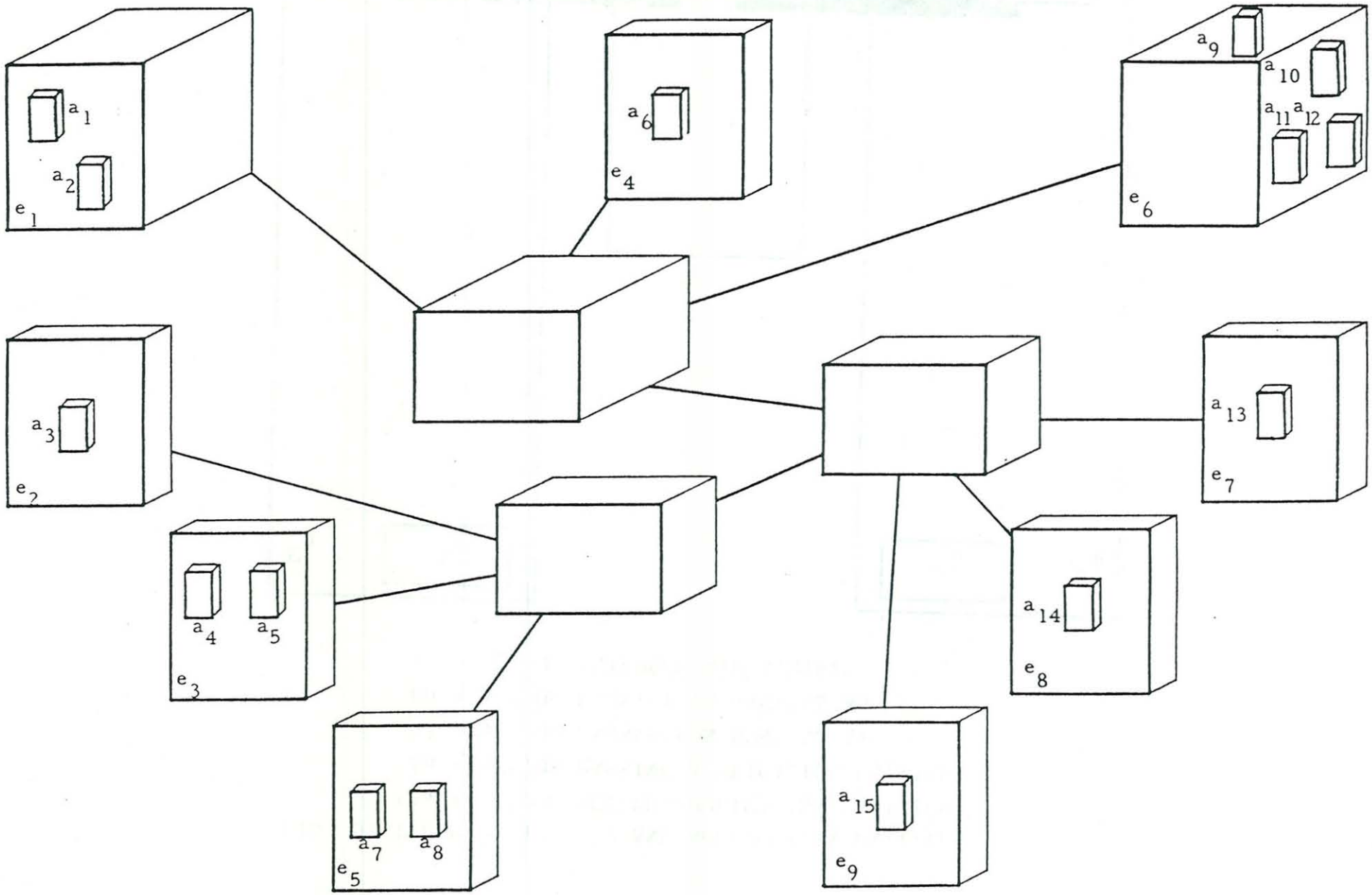


FIG. 8: RESULT OF SYSTEM CUTS

FIG. 9: TYPES OF SYSTEMS AND APPLICATION ENTITIES
(Aa = SET OF ACTIVE APPLICATION ENTITIES
Ap = SET OF PASSIVE APPLICATION ENTITIES
Ea = SET OF ENDSYSTEMS HOSTING Aa
Ep = SET OF ENDSYSTEMS HASTING Ap
I = SET OF INTERMEDIATE SYSTEMS)

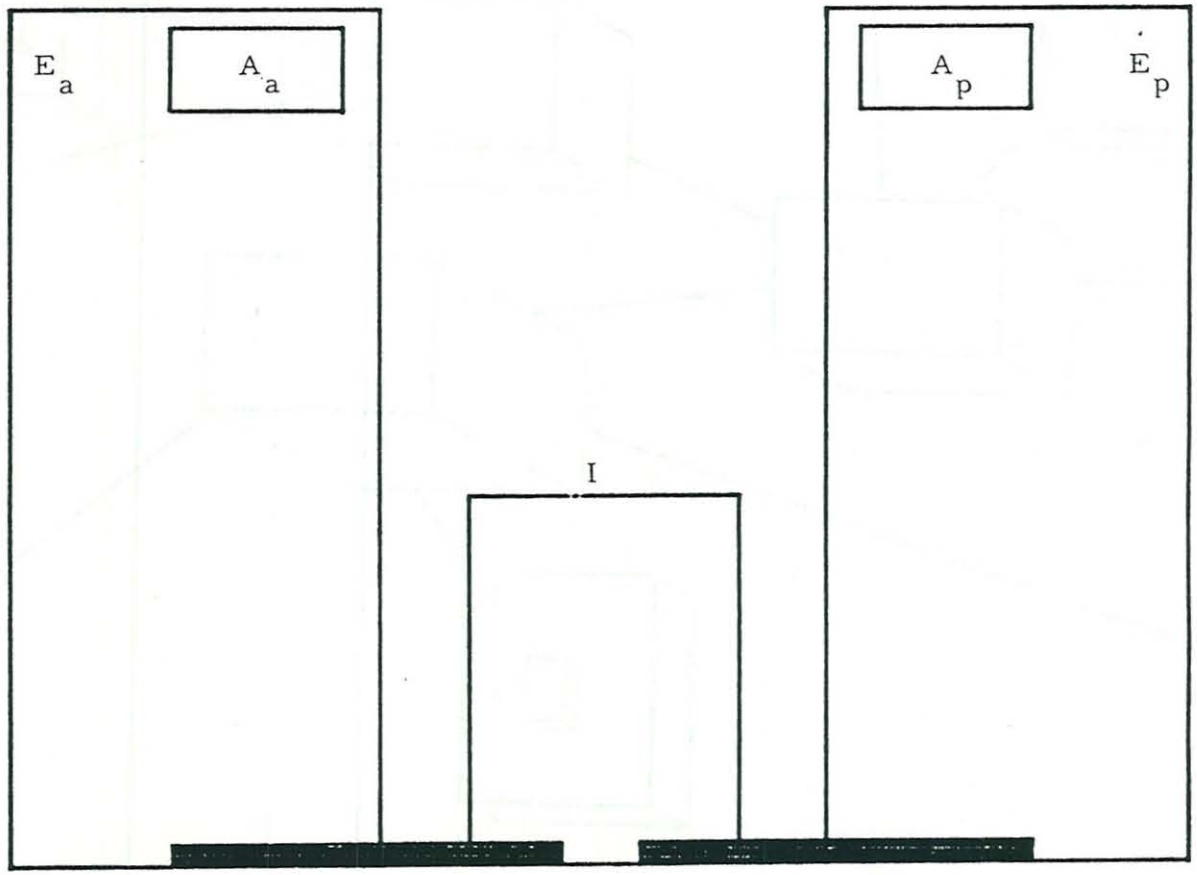


FIG. 10: APPLYING A SERVICE CUT ONTO SYSTEM CUTS

FIG. 11: THE HIERARCHY OF COMMUNICATION SERVICES
(CS = COMMUNICATION SERVICE)

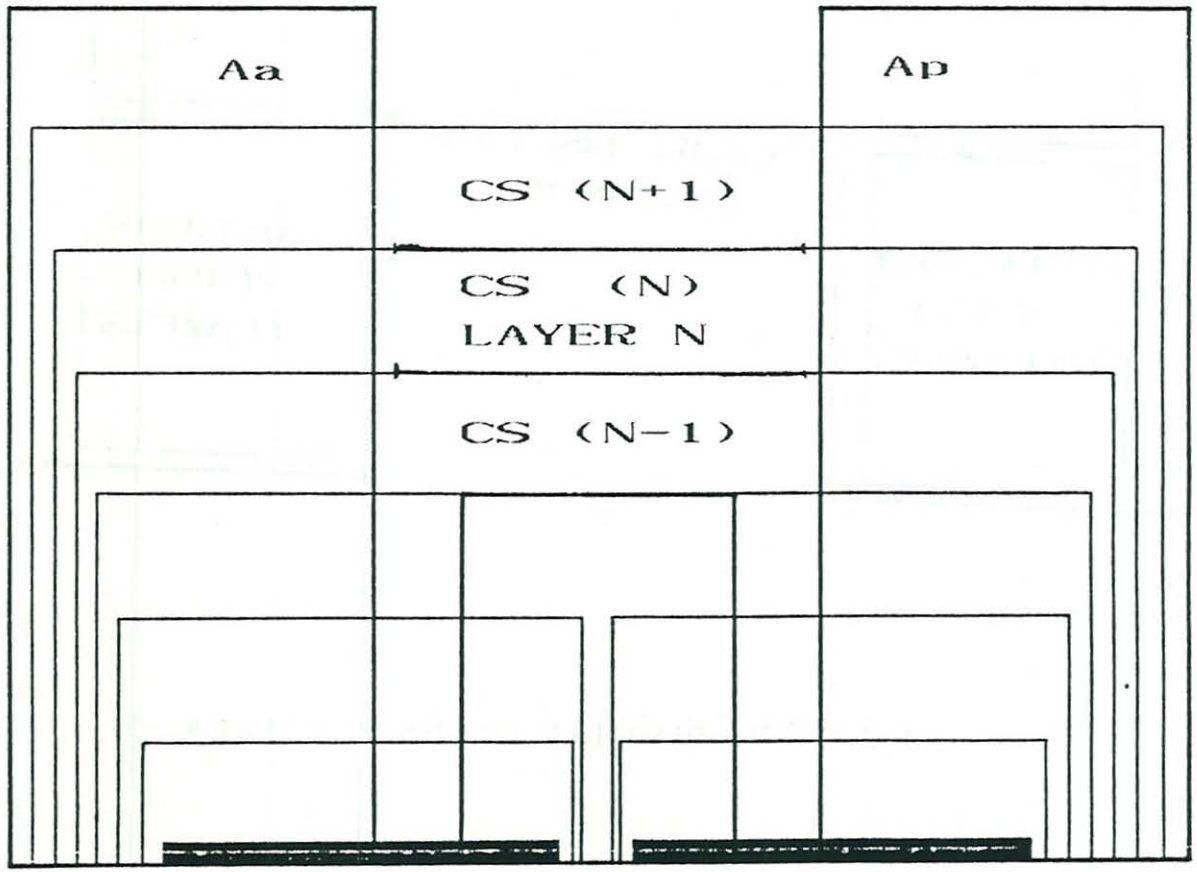


FIG. 12: APPLYING PROTOCOL CUTS ONTO SYSTEM CUTS
AND SERVICE CUTS
(Pa : SET OF ACTIVE PROTOCOL ENTITIES
IN A LAYER
Pp : SET OF PASSIVE PROTOCOL ENTITIES
IN A LAYER)

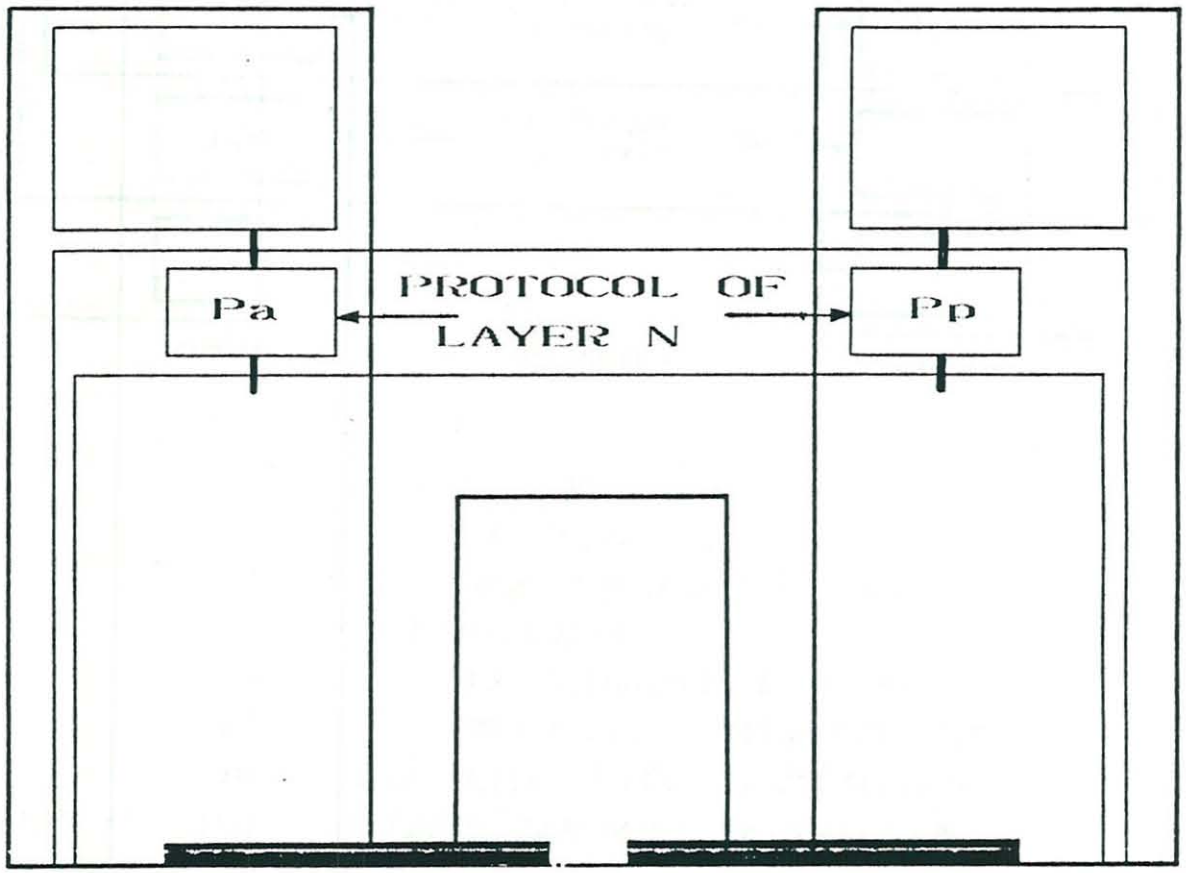


FIG. 13: THE HIERARCHY OF PROTOCOLS AND ENTITIES
(Aa : SET OF ACTIVE APPLICATION ENTITIES
Ap : SET OF PASSIVE APPLICATION ENTITIES
Pa : SET OF ACTIVE PROTOCOL ENTITIES
IN A LOWER LAYER
Pp : SET OF PASSIVE PROTOCOL ENTITIES
IN A LOWER LAYER
R : SET OF RELAY ENTITIES)

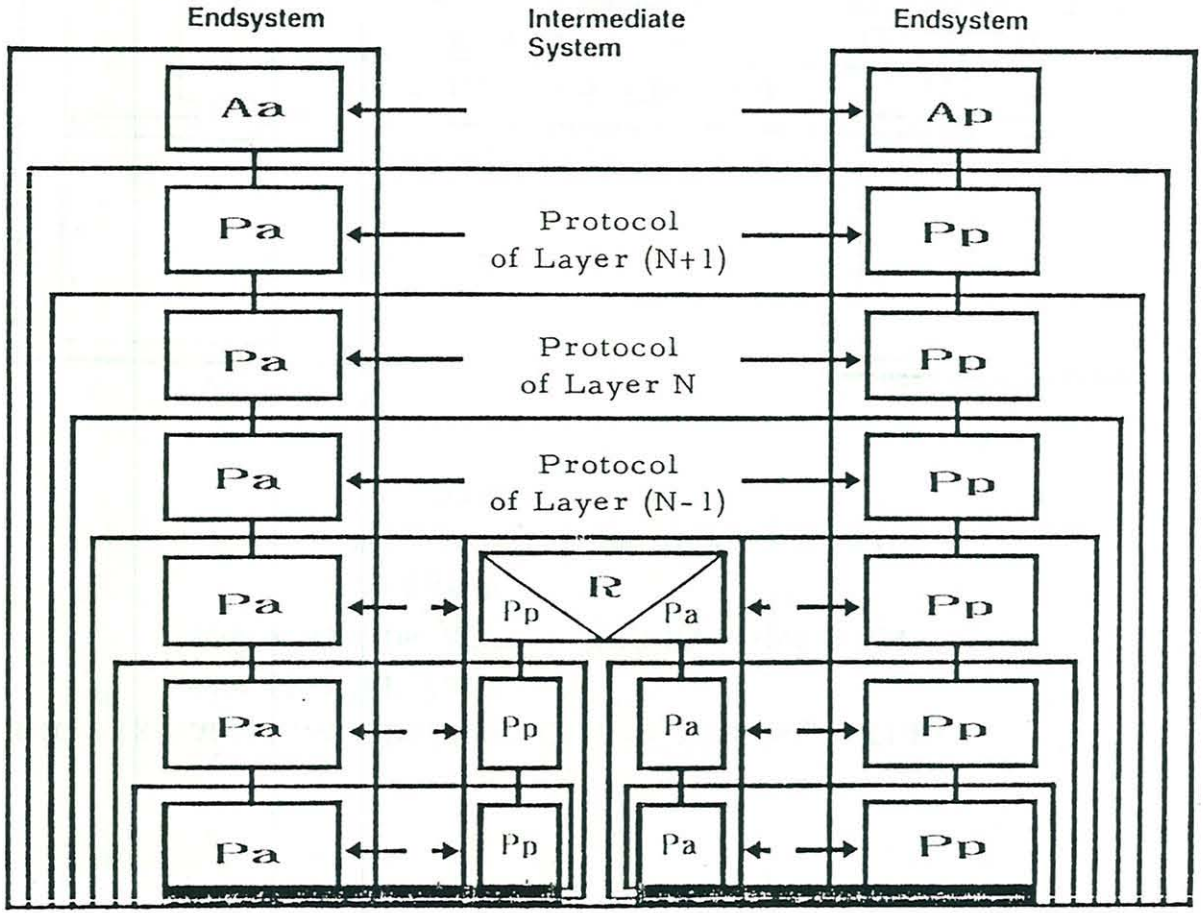


FIG. 14: THE TRANSPORT SERVICE

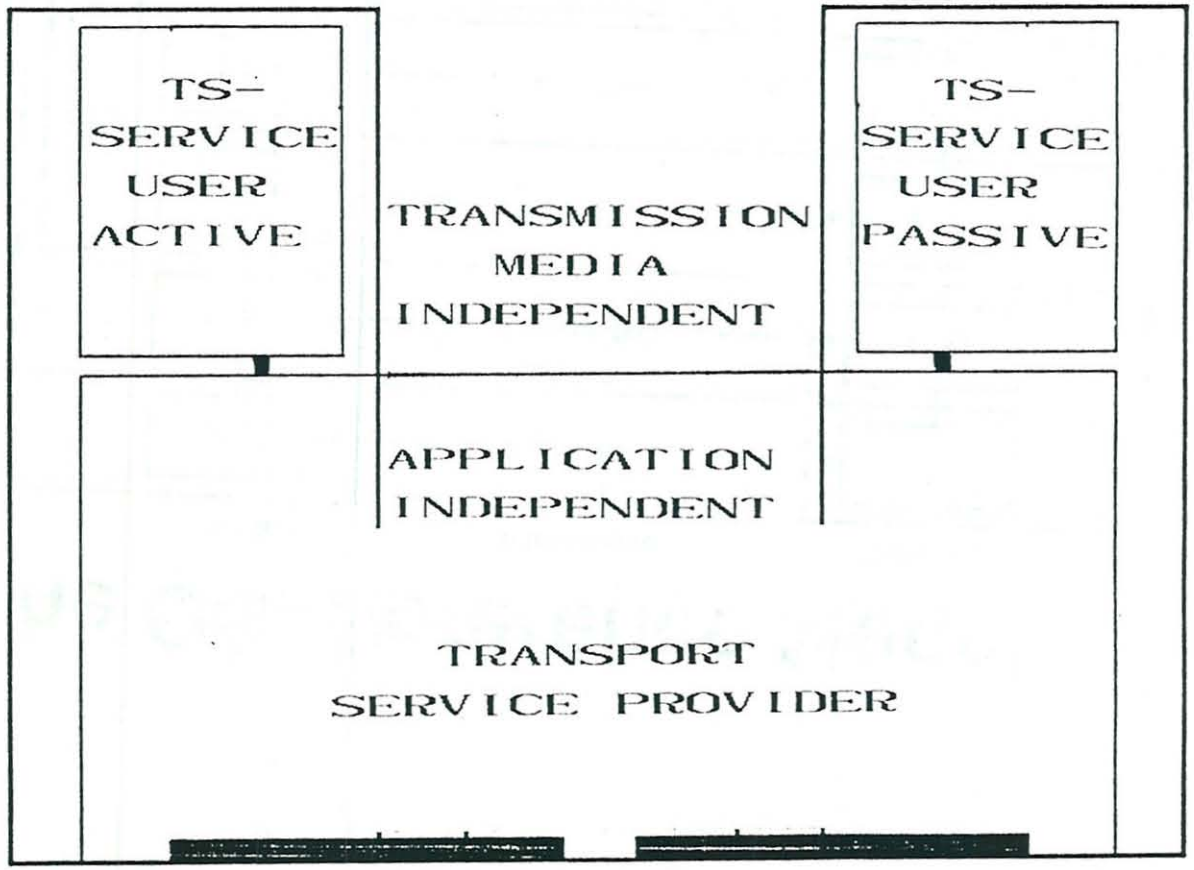
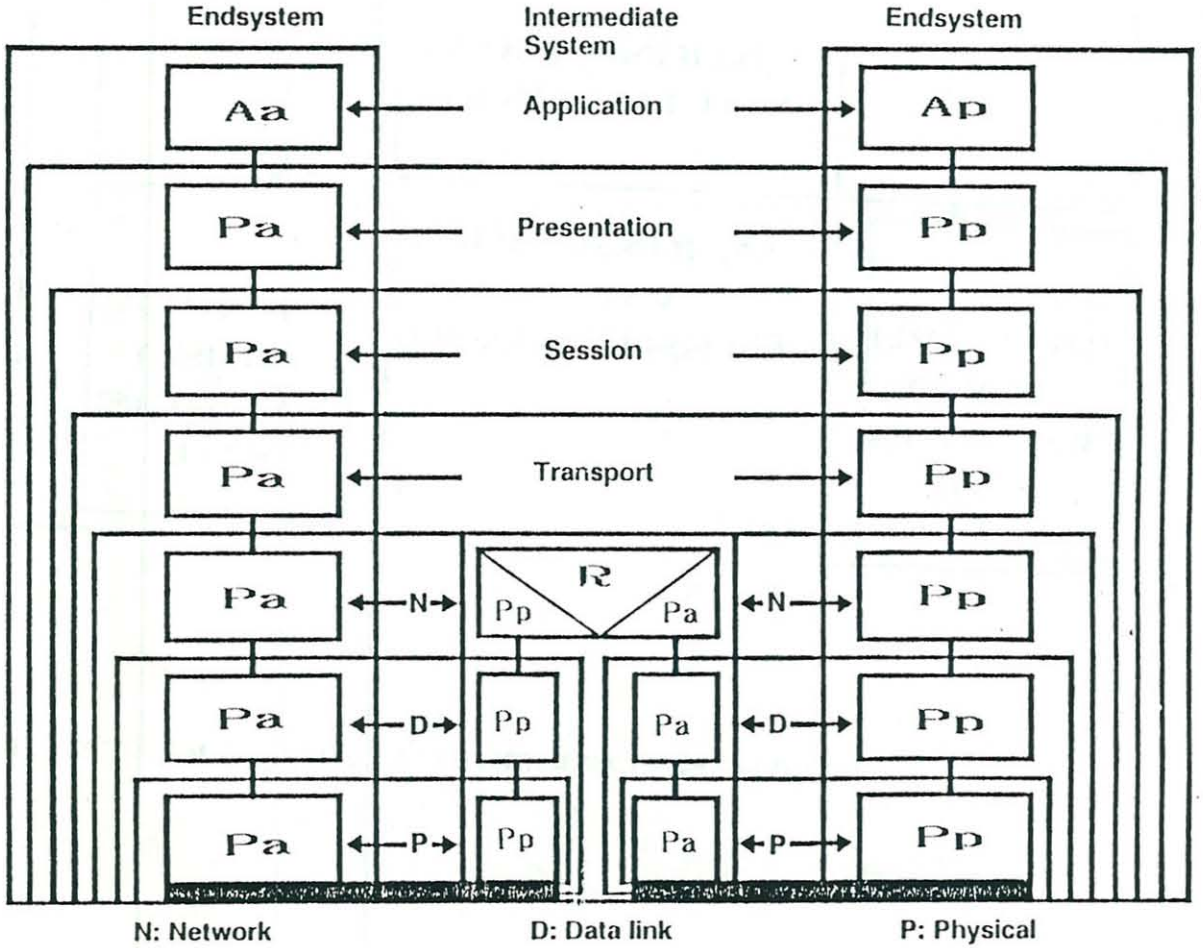
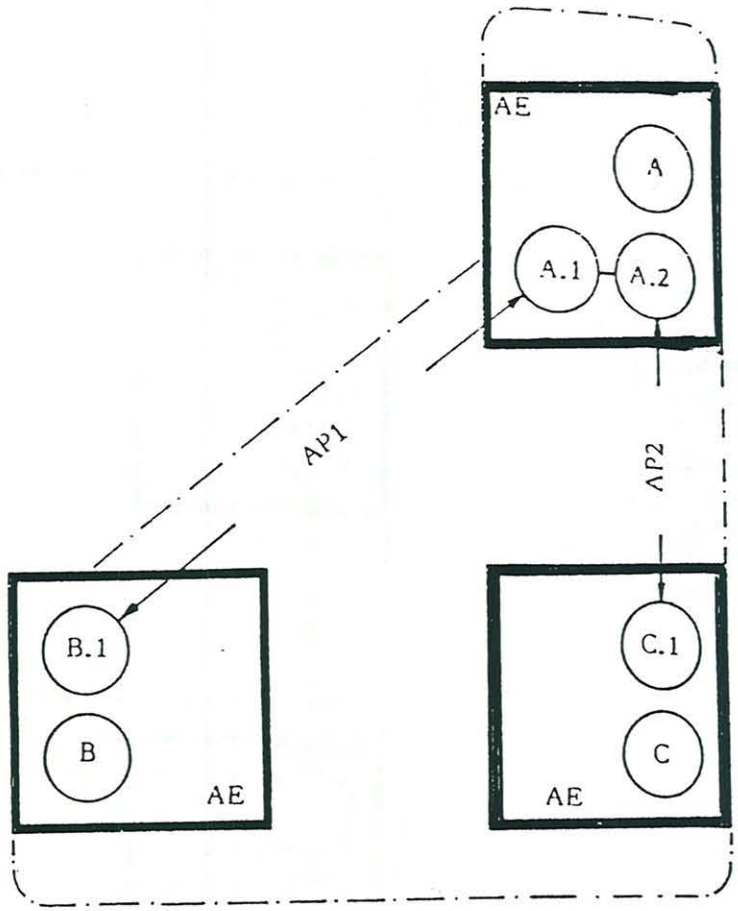


FIG. 15: # The OSI-Reference Model



Application oriented communication services and protocols

Transport oriented communication services and protocols



-.-.-.-.-.: set of application entities forming a distributed application.

FIG. 16: Distributed Application constituted by cooperation of three Application Entities (AE) A, B, C governed by two Application Protocols (AP) AP1 and AP2.

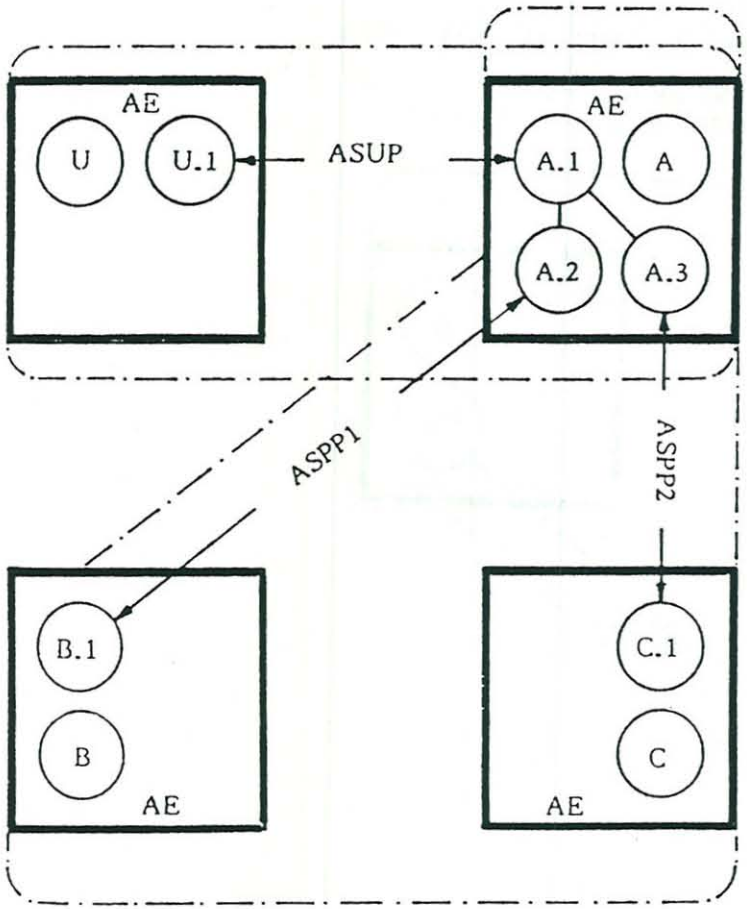


FIG. 17: Distributed Application constituted by cooperation of three Application Entities (AE) governed by two Application (Service Provider) Protocols (ASPP) and represented in regard to its service by the AE A towards an Application Entity U acting as user of the Application Service.
The user U of the Application Service and the Representative A of the Application Service form another Distributed Application: their cooperation is governed by an Application (Service User) Protocol (ASUP).

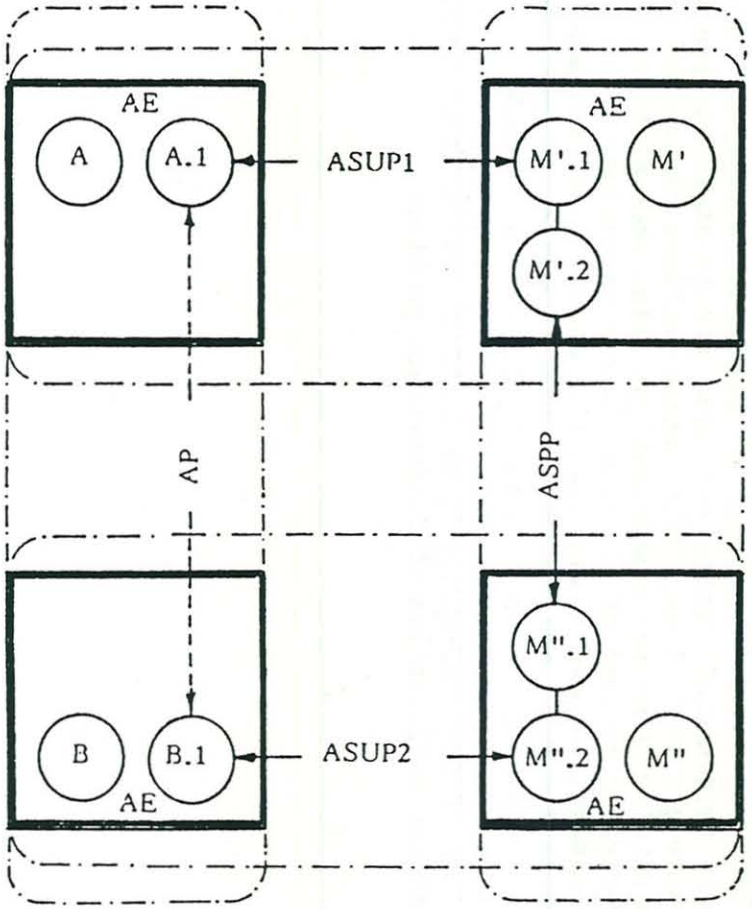


FIG. 18: Distributed Application constituted by co-operation of two Application Entities (AE) A and B which communicate via a distributed Relay Application constituted by two Application Entities M' and M".

DISCUSSION

Professor Pyle asked whether Dr. Burkhardt viewed the model of the OSI architecture he had presented as a piece of research or standardisation. Dr. Burkhardt replied that it was one of research, but that it had been inspired by the standardisation bodies. He doubted if the idea would have occurred without the efforts of the standardisation bodies. Professor Pyle disagreed with this, reporting that the recently terminated SERC project on distributed systems had not been influenced in any way by the OSI standardisation effort.

Dr. Cerf drew attention to one of the slides Dr. Burkhardt had used and asked whether systems really were active or passive. He suggested that in a general distributed system all systems started in the same state (active) and he wondered if this case would fit the model. Did it for example mean that if two systems simultaneously decided to connect to each other that two connections would result. Dr. Burkhardt replied that it depended on the number of free end points in each system. It was impossible for the service provider to know if two connections were really required in this case, and if there existed enough free end points then two connections would result.

Professor Randell wondered whether it was possible to apply the model to connectionless (i.e. datagram) protocols. Dr. Burkhardt replied that this was essentially the same as the first phase of the initiation of a connection and could be modelled in that way.

Professor Tiernari asked whether it would be possible to replace the model with the state transition model. Dr. Burkhardt replied that it would be possible using extended finite state machines but that there were problems. Using finite state machines restricted the model to describing distinct communications acts, whereas the model he had used described the set of all possible communications acts. The major disadvantage, however, was that the state transition model took an implementation view and so it was not possible to model the whole picture since certain things could not be formally defined. Professor Milner proposed that it would be possible to use the Calculus of Communicating Systems (CCS) as an alternative as it remedied the deficiencies of finite state machines.