NAMES AND OBJECTS

IN HETEROGENEOUS COMPUTER NETWORKS

Louis POUZIN
Institut de Recherche en Informatique et Automatique
78150 - ROCQUENCOURT (France)

ABSTRACT

Heterogeneous computers do not use compatible conventions for accessing resources. The problem faced by networks of that type is to introduce a level of commonality without disrupting existing operating systems and services. To that effect, several naming schemes may be considered. The most appropriate is a mapping of local system names into a global C-names space. This provides for a user tailored visibility of network resources.

Basic objects involved in network communication are liaisons, i.e. bridging mechanisms between objects in heterogeneous systems. Virtual terminals are more sophisticated tools for handling terminal oriented data structures. Several techniques may be used to name these objects. A simple and general one uses a pair of C-names.

Other aspects of name management include the use of short names within areas, and the decoupling of names from physical resources. It is shown how reliability and resource management are directly affected by the choice of a proper naming scheme. The CYCLADES name structure is summarized as an application of the concepts presented.

HETEROGENEOUS COMPUTER NETWORK

Networks belong to distinct classes :
a - network of terminals accessing a single computer
b - communication network, connecting pieces of equipment
c - network of computers.

E.g. TYMNET would fall in category b, while ARPANET and CYCLADES fall in category c. Topological diagrams are similar. Each of them appears as a collection of computers and terminal controllers interconnected through a store-and-forward communication network. Differences lie in the kind of associations made possible for exchanging information. Only terminal-computer associations are possible in TYMNET. Hence, the complete set of computing resources connected to the network is partitioned into disjoint sub-sets. Each sub-set is attached to a particular computer, and only one sub-set at a time is visible to each user.

Networks of computers make the complete set of resources visible to each user. If computer systems are homogeneous, it is possible to consider all resources as distinct elements of a single set, because access mechanisms are identical on all computers. Therefore, the partitioning of resources among computers is a matter of physical allocation without logical implications. This is similar to the partitioning of files among several disk units within a single computer system.

Networks of heterogeneous computers raise new problems, because there is no common access mechanism to resources, nor any consistent scheme for naming them. They are not even necessarily similar.

The problem

Given a set of heterogeneous resources, without any common exchange conventions, one must define mechanisms providing for mutual access. These are usually called protocols. Ideally, they should be added to existing computer systems, without requiring modifications. Practically, some adaptations may be necessary, but a complete overhaul of operating systems is to be ruled out.[1]

The first problem encountered in designing common access protocols is the naming of resources. This is the subject of this paper.

BASIC NEEDS

In order to be acceptable from an implementation point of view, common access protocols should leave maximum flexibility. They should be independent from the nature of

resources, and leave entire freedom as to specific control procedures that might be required for accounting, security, etc.

Accessing all network resources as a single set requires a network-wide name space. Furthermore, associations are to be set-up between resources. The simplest kind of association is a pairing. More complicated kinds may be constructed out of pairs.

NAMING SCHEMES

Individual computer systems make use of various naming schemes, which may be specific for different types of resources. E.g. files, processes, peripherals. Names may have a fixed or variable length, and follow specific conventions : alphabetic, numeric, special characters, etc. It would be impractical to change naming conventions in existing systems, because they are usually so ingrained in the design that any change is a major upheaval.

From now on we shall term local names existing names in each computer system. Network-wide names shall be termed C-names. There may be several ways to construct C-names.

Hierarchy : each set of local names $\{Lj\}$ is given a C-name C_i. The network name space is $\{\langle C_i \rangle \langle L_j \rangle\}$. In other words, a network name is obtained by concatenation of a local name and a header designating uniquely a local set.

Allocation : Only a few resources are given permanent C-names, e.g. loggers. Each one is associated with a set of local resources. Accessing a local resource requires the following steps :

- access a C-named resource
- request access to a local resource by its local name
- get a C-name for the local resource
- access local resource by C-name
- after usage, release C-name.

Mapping : each accessible local resource is given a C-name. A mapping between C-names and local names is performed within individual computer systems.

One might argue that the allocation technique is actually a combination of hierarchy and mapping. Other combinations are possible, introducing more complexity. In practice, network designers tend to cook up schemes that suit their implementation problems, rather than general use.

The hierarchy method is apparently the simplest one because a C-name may be parsed

instantly into a network name and a local name. But it is only practical when local names are rather homogeneous. Otherwise, C-names take so many different formats that protocols become unwieldly and inefficient. In particular, the introduction of a new set of local names may require modifications in a number of network access protocols, when the characteristics of the new set have not been anticipated.

The allocation method is favored by a number of operating system-minded people. It has the advantage of fitting within conventional single computer structures. Most operating systems of the past ten years were designed as geocentric objects centralizing all critical functions. Accessing resources is usually a multi-step process starting from a well-known tree top, such a login procedure. This vision of a rigidly partitioned universe is taken for granted by a number of computer professionals.

Advantages of the allocation method is that users access computers in their own familiar way, as if there were no network. It is also contended that using a small number of C-names, for active resources only, saves overhead in network access machinery.

The deficiencies of this method may be derived from its advantages. Keeping network resources rigidly partitioned into computer systems is putting a straitjacket on the user, who would prefer a homogeneous visibility of all resources, with computer boundaries fading out. In other words, the proper vision is a network of resources, rather than a network of computers.

In addition, the allocation method is somewhat cumbersome in implementation, as it requires the management of changing associations between C-names and local resources. Due to transit delays and fuzzy states associated with any distributed system, there appear transient conditions which require specific safeguards to prevent errors.

The mapping method provides for a homogeneous name space in accessing any network resource. It is similar to a telephone numbering plan. Mapping C-names onto local names is a matter of local implementation. This allows permanent or temporary associations, or both. It is therefore more general than the allocation method. Any desirable access control procedure can be triggered as part of the mapping process, not just the login procedure. Such a facility makes it possible to offer homogeneous network-wide access protocols for specific services, which may be available on different computers, e.g. compilers, editors, mail, help, distributed data base.[2]

A criticism of the mapping method is that it takes overhead in scanning large tables of C-names, when they are permanently associated with local names. This is not well substantiated. Indeed, a search is always necessary, whether the key is a C-name or a local name, and there is no reason why searching by C-names should be less efficient.

Space occupied by the C-name table is not critical, as it can be on secondary storage, like any file directory.

Another criticism is that users prefer "symbolic names" rather than C-names. This objection is a typical misunderstanding of the method. Indeed, users (assumed human beings) always access a network through a local machinery (a terminal controller, or an intelligent terminal). Therefore they only have to use their own local names. Whether they prefer distant local names, C-names, or their specific lingo is their choice. Whichever name they use is translated into a C-name to be sent to the appropriate computer system. Therefore, the mapping method is the most flexible from a user standpoint as it does not impose any specific local name set for accessing resources.

A few examples of the flexibility inherent to the mapping method are following :

. A resource may be moved to a different computer without its users knowing it.
. Users may choose local names according to their own symbolism, e.g. in their native tongue.
. Aliases are possible : short or long local names.
. Homonyms may be avoided, since homonyms would appear when distant local names are used.

O B J E C T S

It has become customary within operating systems to use specific objects for handling communications between resources. This is justified by a number of needs, e.g. :

. asynchronism between processes
. control of access rights
. dynamic binding
. Buffering, formatting

Communications between resources are constrained to use specific tools : queues, mailboxes, etc. When resources are scattered over a network, the same needs hold, with additional problems, such as :

. objects are managed by different systems
. communication conventions are not compatible
. there are new error conditions and recovery procedures.

In a single computer system resources may communicate through a single object, like a queue. In a network, there are a minimum of two objects, one at each end. The interface between a resource and its local communication object is a given characteristic which may not be altered. Therefore, the problem in a network is to establish commu-

nications between objects which interface resources. Of course, objects are also a variety of resource.

Matching object characteristics would require a specific conversion for any pair of different objects. Therefore, a desirable objective is to devise a common object-to-object set of conventions applicable at network level, and make local conversions, when necessary.

L I A I S O N S

A basic building block for communications is a mechanism capable of transferring blocks of information from one object to another, as if the two objects would share a common buffer. Carrying out this function with all the necessary safeguards and error procedures is the purpose of a transport protocol. Part of this protocol is devoted to the setting up of a logical association or liaison, which is a bridge between the two objects. Buffers and state variables make up a liaison context. Thus, a liaison is a network object interfacing with local objects at each end.

Designating a liaison at network level is not useful. What is necessary is that two objects be able to set up a liaison, and then use it for exchanging blocks of information.

Setting up a liaison requires that both objects know each other by a C-name of some sort. Thereafter, several cases may occur :

. Objects are limited to a single liaison

They only need to dicriminate between valid liaison messages and others. Received messages may carry a unique identifier, e.g. a password, which has to be predefined. It can be exchanged between objects when the liaison is set up.

. Objects may set up multiple liaisons

Messages must carry some identifier per liaison, which is a pointer to one of the contexts associated with each object. An index value is exchanged at set up time. Each end may use a different index. It does not matter whether an object uses its own or the other index when it sends messages, as long as this convention is known.

Index values may be assigned in monotonically increasing sequences. If the numbering cycle is big enough as compared to the lifetime of a liaison, there should not be any ambiguity. On the other hand, index values may be assigned, released, and reused dynamically. Since they are shared by two objects, there appear risks of transient conditions in which errors may occur. This is similar to the dynamic allocation of C-names.

A way to eliminate all problems associated with liaison index management is to use a pair of C-names as an index. As long as C-names are assigned in a stable or safe way, liaisons may be identified without ambiguity. The only restriction is that only one liaison at a time may be set up between any pair of objects ; but a single object may set up multiple liaisons with other distinct objects.[3]

Liaisons do not make any data or format conversion ; they are just a basic layer which is always necessary for transferring data in a transparent mode.

VIRTUAL TERMINAL

The concept of virtual terminal has been brought about by the need to use a variety of physical input-output devices in conjunction with a variety of computer systems. Problems associated with the physical characteristics of the devices are non-trivial, and will not be covered here. Rather, the virtual terminal concept will only be introduced as a network object.

Decoupling I-O devices from user programs is a typical feature of any operating system. User languages address logical I-O objects, which are mapped at execution time onto physical devices. The mapping device is what is often called an access method.

In a network context, user programs and I-O devices are normally attached to heterogeneous systems. Thus there is a need for a bridge between logical I-O devices as they are seen from both ends. No general solution has yet been invented. However, typical cases are usually tractable.

Data produced or processed by a program are somehow handled as pieces of a data structure. The mapping of this structure onto a physical device is sometimes elusive because what is really aimed at is a mapping onto a mental image of the structure as seen by a human user from the physical presentation on his terminal. When there is no human user, e.g. for program-to-program communication, only mechanized data structures are involved. Assuming that these structures can be mapped into one another, this mapping would be the task of the virtual terminal protocol. Thus, a virtual terminal may be considered as a network object making a data structure accessible from two heterogeneous domains.

Naming problems are the same as for liaisons, because a virtual terminal is only a sophisticated logical association between two local objects. Since it uses normally one liaison, the same designation can apply to both.

REGIONAL NAMES

When the total space of network names becomes very large, C-names become very long, and may generate overhead or inconvenience. A counter-measure is to partition the name-space into a hierarchy of subsets designated by area names. If the partitioning is clever enough, most liaisons fall within a single area boundary. This is typical of telephone numbers. Thus, there are two or more C-name formats, depending on the number of partition levels crossed by a liaison.

There are cases where an area boundary must cross a large number of liaisons. E.g. it must follow an administrative boundary, or some areas are too dense. A solution is a partitioning in which each area owns a subset of names disjoint from its neighbors. This means less names per area, thus more areas. But the advantage is that short names may be used to communicate not only within an area, but also with every neighbor area, (fig. 1) .

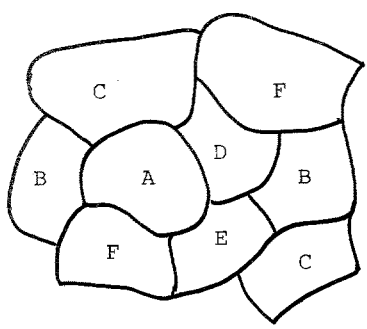


Fig. 1 - NEIGHBOR AREAS

E.g. short names may be used for liaisons between objects located in area A on one end and objects located in any adjacent area B,C,D,E,F on the other end. Also area D may use short names towards adjacent areas A,B,C,E,F.

This method is equivalent to a partitioning of the name space into large areas overlapping each other. The topological representation given here is just a logical model, without geographical connotation. The distance between objects on a diagram may be interpreted as a measure in inverse proportion to their density of intercommunication. If this can be mapped onto a bidimensional representation, areas may be delineated according to their topological properties. E.g. areas with 3 adjacent neighbors (triangles) require 4 disjoint sets of short names. Areas with 6 adjacent neighbors (hexagons) require 8 disjoint sets of short names.

Such methods are typical of telephone or telex numbering plans. They are intended to

reduce the burden of using long names tailored to a world-wide space.

PHYSICAL VS. LOGICAL NAMES

It became long ago standard practice in operating systems to designate objects by names (e.g. files or I-O devices). In some cases names are pointers or indexes in a table. But it is extrelemy unusual that names be tightly associated with pieces of hardware.

On the contrary, names in telephone systems designate exchanges and physical subscriber ports. As a result, telephone numbers change whenever a subscriber moves, or when the telephone company redistribute its subscribers among new exchanges. This is particularly awkward, as telephone numbers are becoming the most frequent names used to access businesses or individuals. This constraint was understandable when telephone plant was entirely electromechanical. Since the introduction of computers for circuit switching, there is no technical justification for maintaining such an obsolete limitation.

An essential by-product of logical naming is reliability. Indeed, designating an object by means of a physical equipment leading to it creates a dependency on the availability of that equipment. Failure, maintenance, reconfiguration are bound to disrupt accessibility and continuity of service. E.g. a telephone subcriber is cut off when his local loop is disabled, because there cannot be any alternate access, due to the physical connotation of telephone numbers.

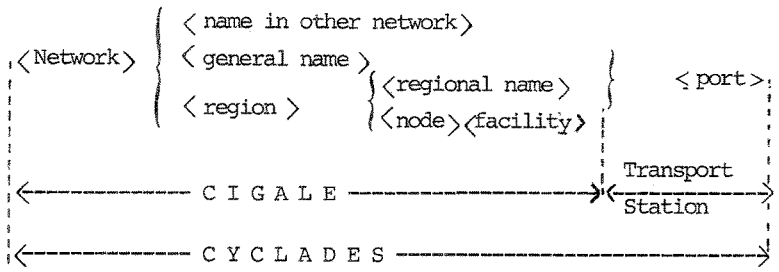
When logical naming is used, it is possible to organize the access to objects through alternate paths, depending on the availibility of physical resources. This allows for higher reliability. E.g. a data processing center offering services to a large population of users, through a public transmission network, may be composed of a duplex computer system connected by two separate circuits to two different exchanges. As long as the circuits do not follow the same physical path, there is complete redundancy in access gear, with a single name.

Another benefit of logical naming is a higher degree of flexibility in physical resource management, since there is no predefined allocation. Reconfigurations may be performed without service disruption. It is also possible to introduce classes of service, for which specific resource management strategies can be devised, e.g. bulk, interactive, real time.

THE CYCLADES NAME STRUCTURE

An application of the concepts presented in this paper is the CYCLADES[4] computer network, of which CIGALE[5] is the packet switching sub-net.

The C-name structure follows :



The distinctive characteristics are :

- The name space includes all name spaces of other networks.
- The name space includes specific addressable facilities within CIGALE nodes.
- All other names are hardware independent.
- At the CYCLADES level, C-names designate ports.
- At the CIGALE level, C-names designate transport stations.
- Transport stations names are usually regional, but they can also be known as general names at the whole CIGALE level.
- C-names are independent of local names and are mapped locally.

C O N C L U S I O N S

An observation of existing computer and communication networks shows that the naming problem is usually not well understood. In most designs, names are just ad hoc conventions tailored for limited implementation purposes, rather than being viewed as a critical tool for resource management at network level. This is all the more a concern that planned public data networks do not appear more advanced in that matter than antiquated telephone systems.

R E F E R E N C E S

1 - ZIMMERMANN H. - Insertion d'une station de transport dans un système d'exploitation (Jan. 75), Réseau CYCLADES SCH 546. 8p.

2 - POUZIN L. - Access protocols in a network environment. (Mar. 1975) Réseau CYCLADES SCH 549. 2p.

3 - ZIMMERMANN H. - The CYCLADES end-to-end protocol, Fourth Data Communications Symposium, Québec city, (Oct. 75) 6p.

4 - POUZIN L. - Presentation and major design aspects of the CYCLADES computer network, 3rd Data Comm. Symp. IEEE Tampa, Florida (Nov. 73) 80-87.

5 - POUZIN L. - CIGALE, the packet switching machine of the CYCLADES computer network, IFIP Congress, Stockholm, (Aug. 74) 155-159.