RESEAU CYCLADES

SCH 516

August 1973

INWG Note # 49

NIC # 21653

NETWORK ARCHITECTURES AND COMPONENTS

Louis POUZIN

This paper has been presented at the 1st European Workshop on Computer Networks, Arles (France), April-May 1973.

Received at NIC 22-FEB-74

O U T L I N E

I - INTRODUCTION.

There are various brands of computer networks. The present paper is only concerned with the type of general purpose heterogeneous computer network of which Arpanet is a typical prototype. In this field experience is scarce, since no network has yet progressed beyond an initial experimental stage, while most are still in a planning or construction phase. Consequently, applications and user needs can only be thought of in prospective terms. Organizational and sociological issues hiding underneath the concept of network may be of such magnitude that many years will elapse before we can record any significant breakthrough.

In the present stage major efforts bear on technical issues, with the objective of developing and mastering the network technology. We have to recognize that proposed structures and techniques have yet to be validated from experience, and that various other approaches might as well be considered. It would be nice if we had practical evidence that present views are justified by economics and user acceptance. But assessing networks on today's applications is not necessarily a sounder standpoint.

II - COMPUTER NETWORK CHARACTERISTICS

A - Heterogeneous installations

Networks of computers may be homogeneous. But this is the exception rather than the rule. Some organizations, like IBM, Control Data, University Computing Co., etc... have set up computer networks, to provide specific services, using the same brand of machines and operating systems. Sooner or later a new generation comes by and must be inserted. Replacing a whole network with new systems in a short span of time is quite unrealistic. Thus, homogeneous networks are bound to turn heterogeneous, or die out.

Most computer networks cannot afford to be homogeneous in the first place. They have to start out with existing installations, which happen to be a mixture of every computer system on the market. But this is a reasonable constraint, as it forces to design network mechanisms for the general case, which makes them more suitable to accomodate future extensions. (Fig. 1)

As a result of this heterogeneity, there is no common interface, or procedure to interchange data. Actually, this is not a direct implication of having heterogeneous computer systems. Indeed, there might exist some generally agreed methods to govern data exchange between various computers. But this is not so. Every manufacturer is still entrenched in its closed world of idiosynchrasies, and must devote a great deal of effort only to have its own machines to communicate.

Installations are not just heterogeneous in terms of computers, but also in terms of management, activities, interests, habits, and there is no magic receipe to turn them into a strong unified body. Being part of a network should not mean losing autonomy and decision freedom. They want to retain all their capabilities to run their own business as they please. Local peculiarities are to be expected as a matter of course, and operation of the whole network should be made independent of local installation behavior.

As in any association, federation, or business, some members tend to cluster in sub-groups, or clubs, based on some common ties not shared by others. In a computer network, clubs may be made of users of the same system, service bureaus, bankers, installations of a corporation. Presumably, most of the traffic generated by members of a club will be exchanged with other members. Conventions about information interchange may well be tailored to some particular dominant type of transactions peculiar to the club. (Fig. 2)

On the other hand, members of a club will want occasionally to have some exchange with members of another club. They may even belong simultaneously to different clubs. Although a club sounds initially as a closed group, it should not be that closed after all. Some excursions should be possible toward foreign installations, just in case. In fact, installations belong to what we may call semi-closed groups.

B - Data communications system

The challenge in designing a communications system for a computer network is that only qualitative aspects may be guessed. No figures are available, since computer networks do not exist. Or at least they have not yet emerged out of the initial building up stage. What we may assume is that there will be computers and terminals, and some barely predictable amount of data traffic between them. Presumably, the amount of traffic will be related to the capability of those devices to generate or accept data. In other words, computer-to-computer traffic, high speed, and low speed terminals will likely fall within distinct categories of traffic patterns.

Therefore, the communications system will have to meet simultaneously a mixture of requirements pertaining to different classes of data interchange. (Fig. 3) This comes in addition to carrying dominant flows, depending on the geographical topology of installation clubs.

Since the communications system should cater for an unlimited number of computer makes, be suitable for semi-private traffic, and work in an environment of loosely related installations, it cannot depend on any of them. In a context of mutually suspicious parties, the communications system will be deemed at fault, unless it has proven very reliable.

Furthermore, it should carry, within some limits, a somewhat unpredictable traffic, ranging from low speed conversational messages, up to bursty high speed process-to-process interplay. New installations may be added, others can be discontinued. Since the number and type of applications using the network is a priori evolutionary, there cannot be any favorite characteristics tailored to some specific traffic. Although some requirements may turn out to be contradictory, the communications system has to walk a thin line, and comply efficiently with a diversity of traffic patterns. This is called flexibility, in a subjective sense, as opposed to rigidity.

Carrying data between heterogeneous computers or terminals, including all sorts of private or semi-public conventions, would end up in a Penelope web type of task, if the communications system had to get involved in data codes, formats, procedures, etc... Adding constantly new features, fixing old ones, would not make for a reliable system, and installations would resent being tied with not quite satisfactory network options. A better approach is to make the communications system data transparent, in order to accomodate any type of code or convention that installations might want to use. (Fig. 4)

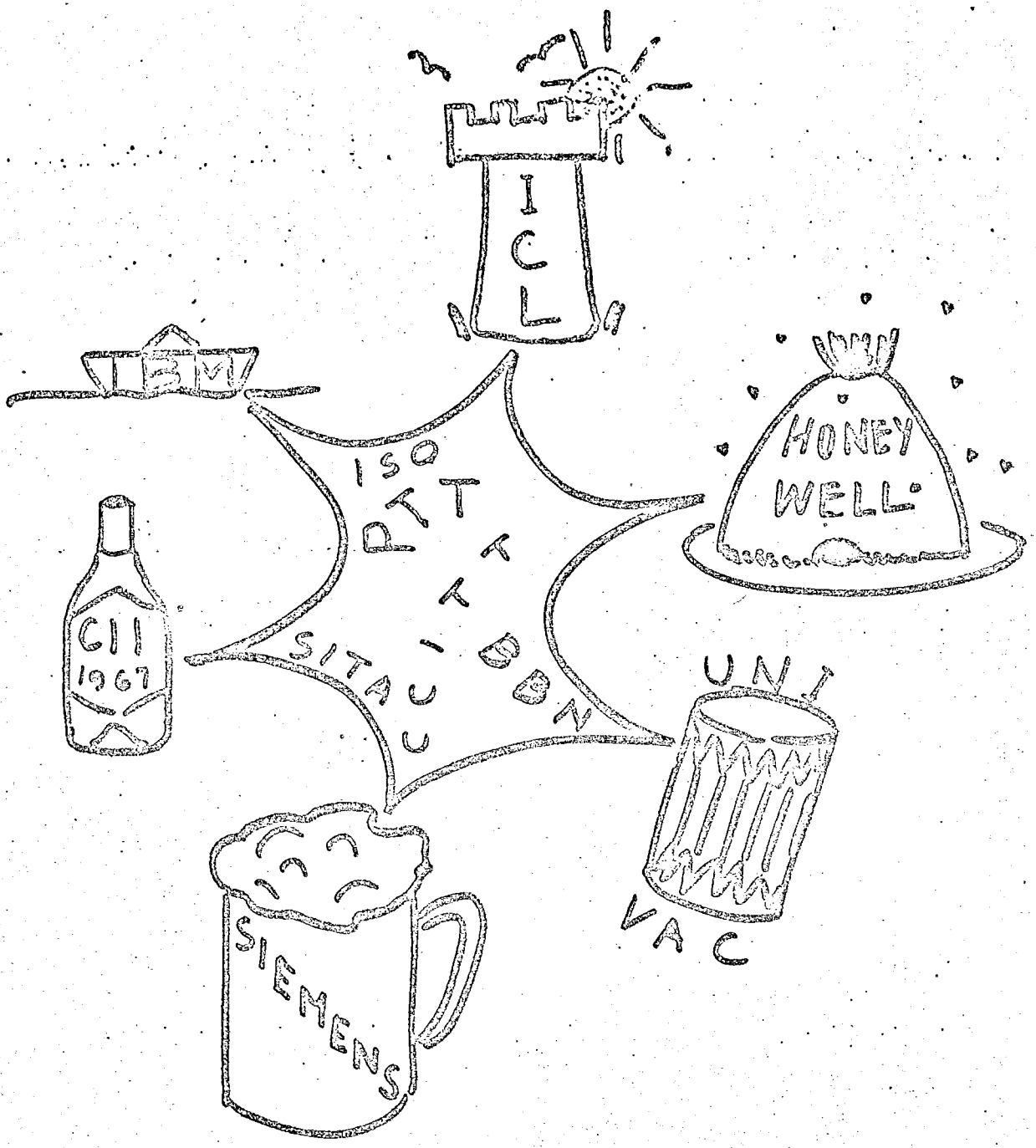
Once a network starts operating, it becomes very frustrating for users to disrupt its working. Particularly, if they have built a substantial investment in software and organizational structures. A most appreciated virtue of a communications systems is to provide for abiding service, so that long term planning be carried out, counting upon well defined network characteristics. Communications are akin to a public service. Changing components of the communications system should not have any visible impact, other than improvement, on the way installations exchange data. Requiring all users to adapt their programs overnight would verge on sheer upheaval.

The previous considerations converge towards opting for a communications systems as independent as possible from installations or terminals which it is to serve. On this way, both users and communications system can protect their investment, and be free to introduce whatever gadget they like, as long as they keep the same conventions to talk to each other.

C - Basic structure

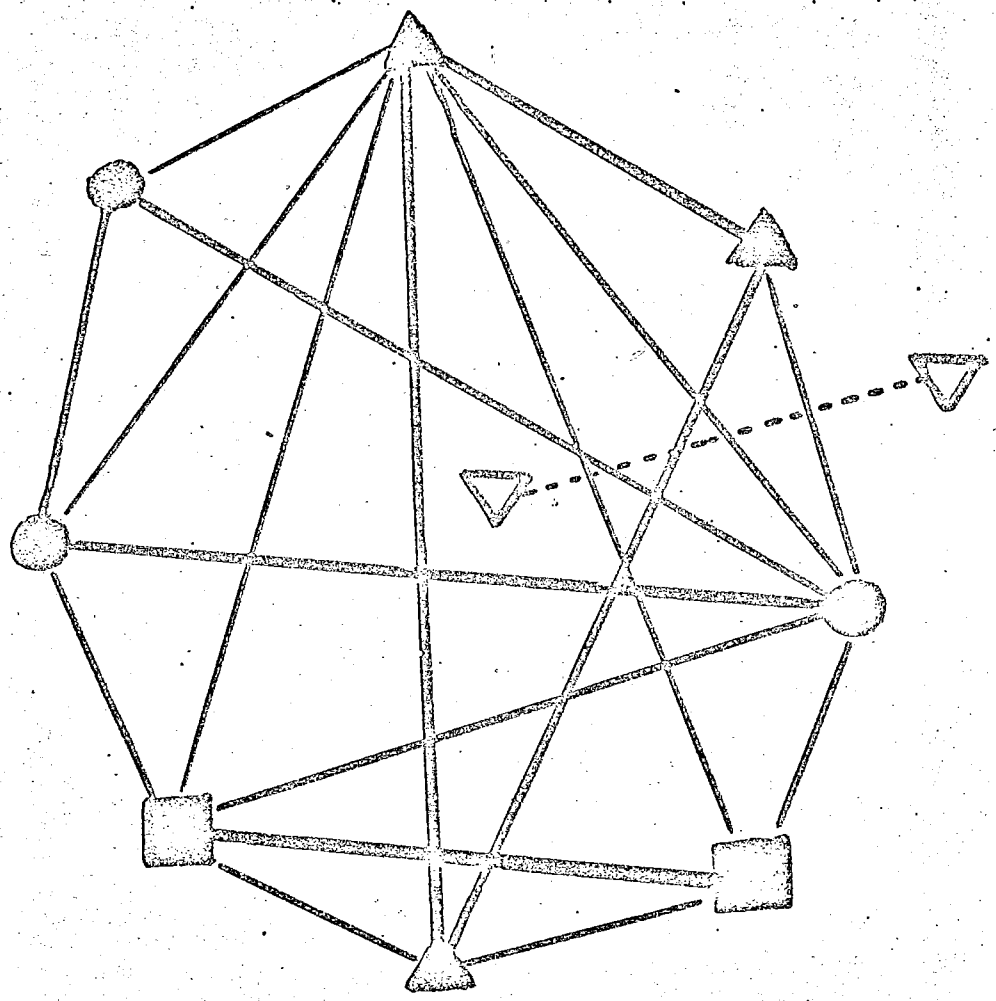
As already exemplified in Arpanet, the resulting structure of a computer network is two-level. A first level is the communications network, in charge of carrying messages between computers. A second level is the set of computers, concentrators, terminals, which use the communications system as a black box. (See e.g. Pr Dadda's paper)

Ideally, the communications system should be so transparent that installations would not even notice it is there. In other words, installations are not concerned primarily with exchanging messages with some communications medium. They would like to send and receive data as if they were in direct connection with other installations. But we have seen previously that installations are heterogeneous,



Heterogenius computer network

Figure 1.



Semi- closed groups

Figure 2.

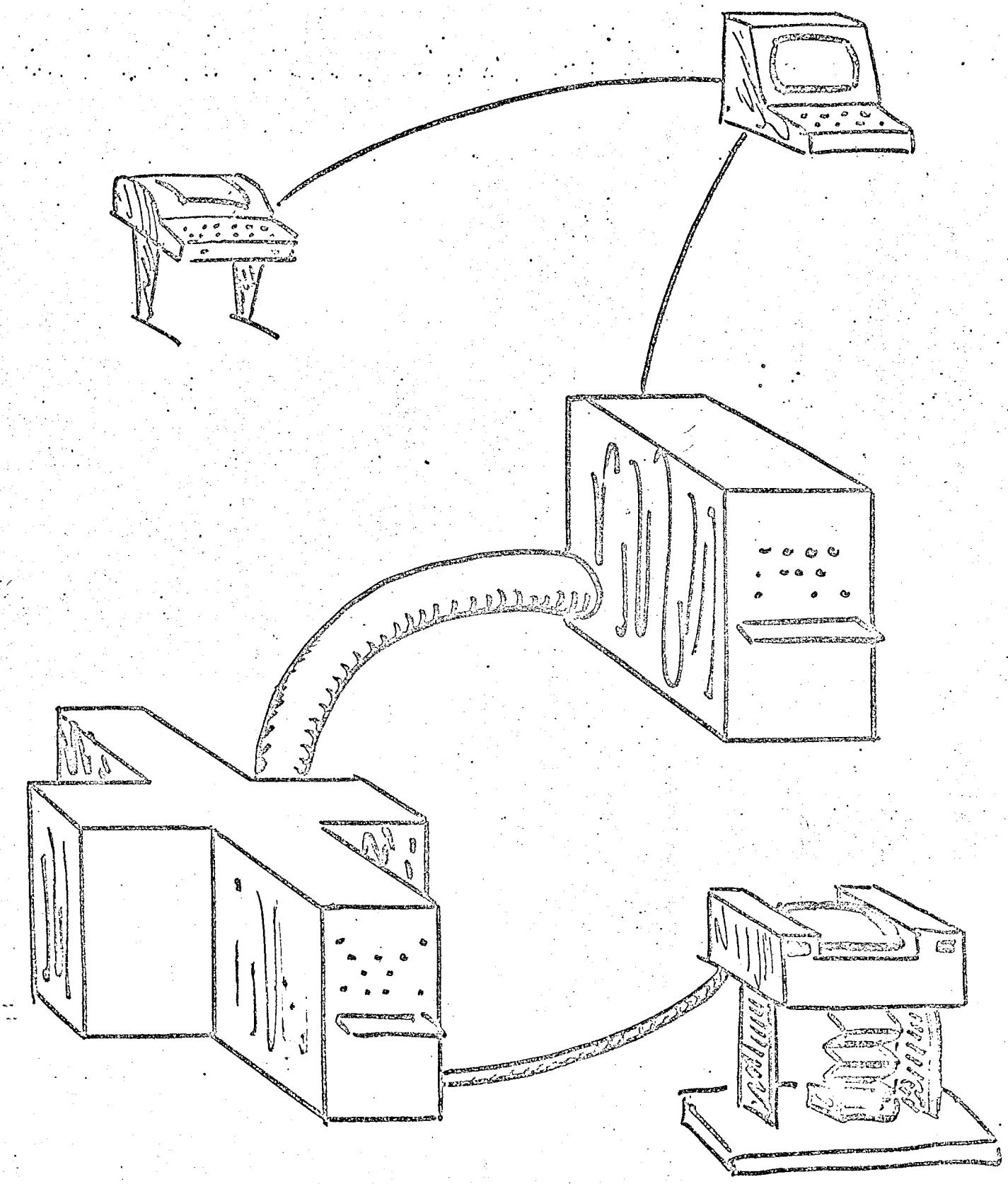


Figure 3.

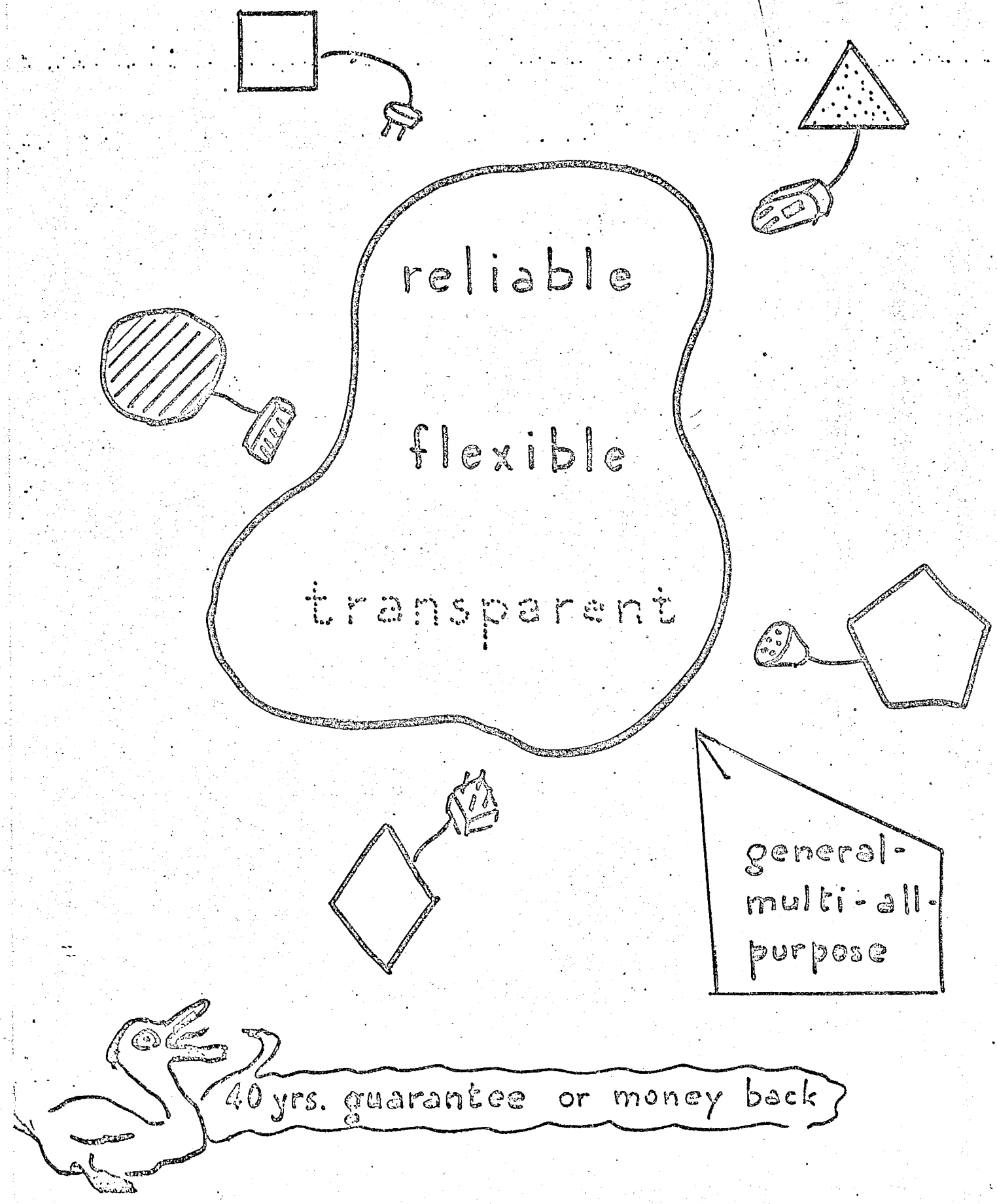

reliable
flexible
transparent
general-
multi-all-
purpose
40 yrs. guarantee or money back

Figure 4.

and do not have any common convention to exchange data. A solution would be to force every one of them to adapt to a network standard. This might be reasonable if such a standard were originating from a distinguished body, like ISO, or CCITT, or the U.S. Navy. Otherwise, one can expect most computer manufacturers being up in arms, as there is nothing they loathe so much as users putting on their machines some exotic piece of hardware or operating system.

A more attractive approach consists in putting some interface adapter in the communications system, so that no modification be required from individual installations. This is acceptable, since data speeds are standardized by CCITT, and transmission procedures can be limited to those of a half dozen major manufacturers, while others can be talked into adapting their software. (Fig. 5)

Nevertheless, procedures may change with a new system release, and new ones are put forth over the years. Thus, it is typically the kind of function that should be made modular and optional, so that the rest of the communications system be kept well insulated from procedure updating. Before long it will be located in a pluggable micro-program. (Some machines already have this feature).

We have seen above that installations may occasionally interfere with members of different clubs, and that some private conventions, as opposed to manufacturer's, might be used in the context of specific applications. Thus, a single installation may wish to use a few different options, according to its type of work. This is quite feasible if it has several links with the communications system, or if it can direct messages toward appropriate adapters according to a message type, (software link). Logically this results in the communications system seeing several virtual installations mapped onto the same physical one. (Fig. 6)

A bare communications service may be unsufficient for some installations. They may wish to have more sophisticated options, like long term storage of data, information retrieval, traffic back-up in case of installation outage, code and format conversion, and what have you. There is obviously no limit as to the type of services that may be offered, grafted on information transfer. It is an area where experience and an imaginative marketing approach can result in many an interesting business. At first sight, this seems in contradiction with the desirable characteristics of a communications system : long term stability, high reliability, while said services would smack more of a commercial product with its all too familiar lack of testing, specification changes, quick obsolescence.

These opposite objectives may be reconciled through an appropriate system design. Anything not directly relevant to data communications purposes should be left out of the communications system. On the other hand, desirable add-on functions can be implemented by specific installations considered as somehow embedded within the communications system, as seen from external users, while they would actually be logically independent, as any other external installation. For all practical purpose; their junction with the communications system would be no different whatsoever. (Fig. 7)

On the previous chart, we have represented some "internal" installations. But there is no intrisic technical difference between internal and external, as far as the communications system is concerned. The distinction is intended to stress such considerations as guarantee of service, tariff structure, maintenance, and the like. But there is a continuum ranging from a public service, to a non-profit agency, to a service bureau, to an organization offering services as by-products. The decision as to which services should be offered from within or from outside is mainly commercial or political. One can even switch status, without interference in communications techniques.

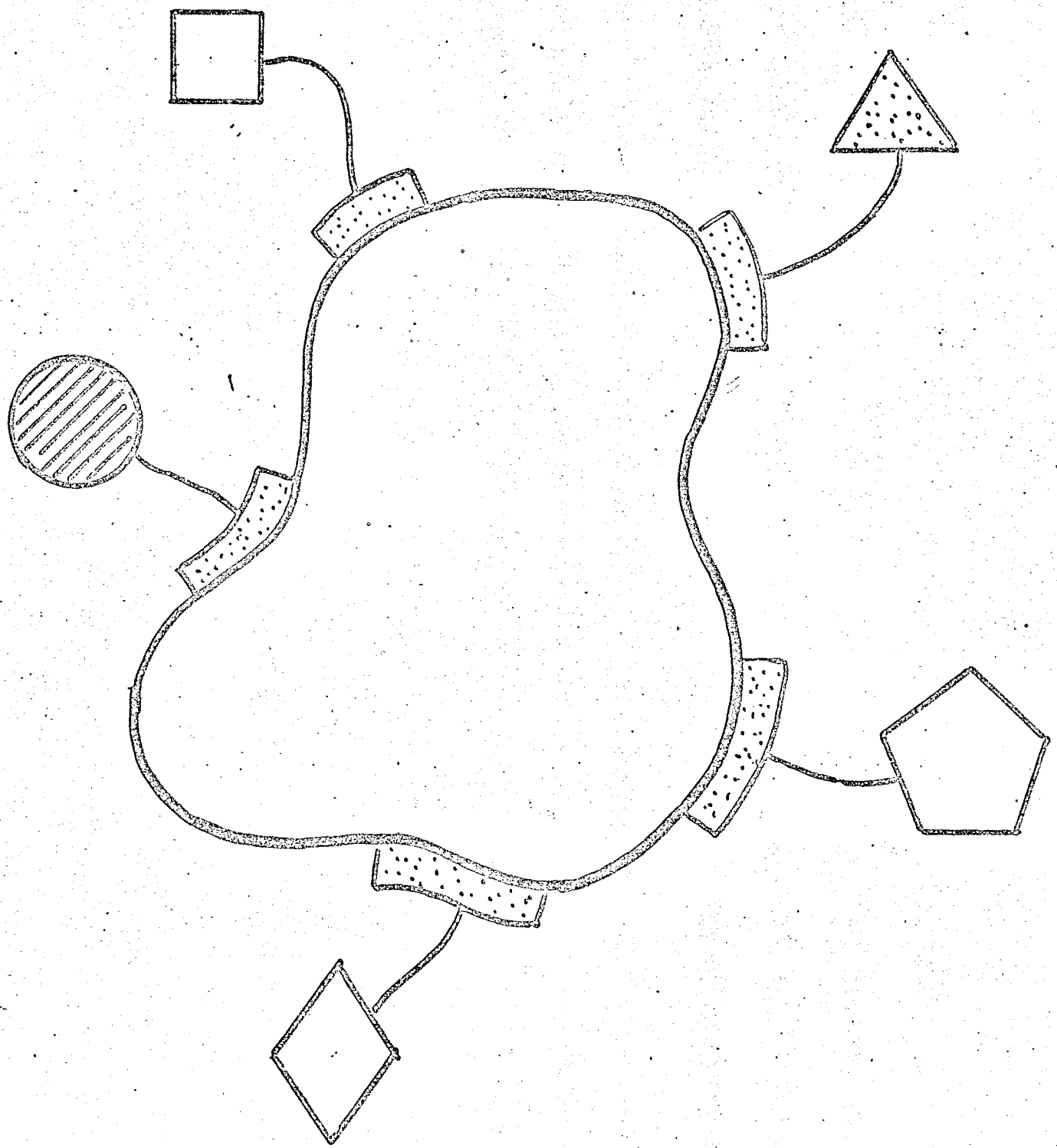


Figure 5.

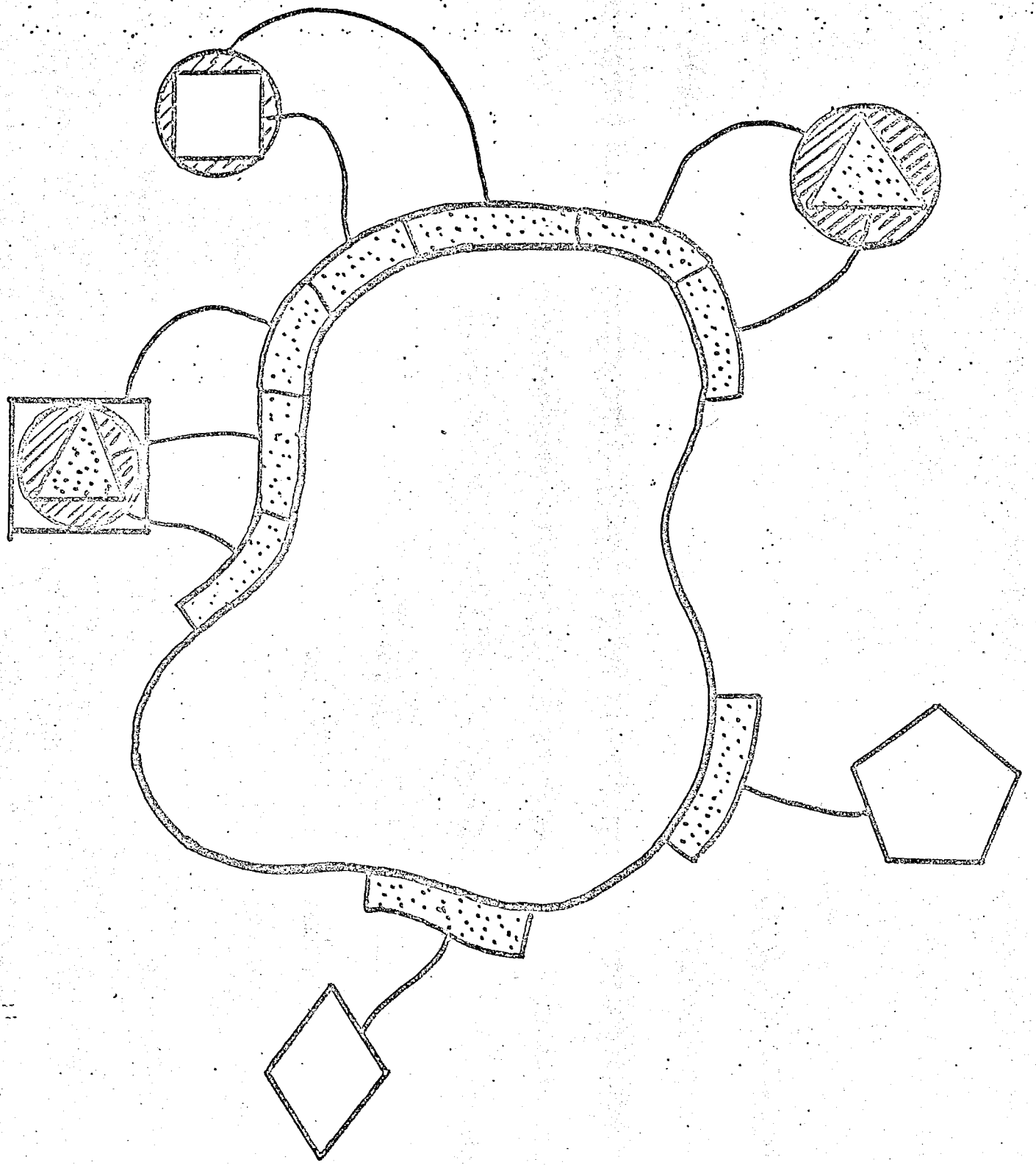


Figure 6.

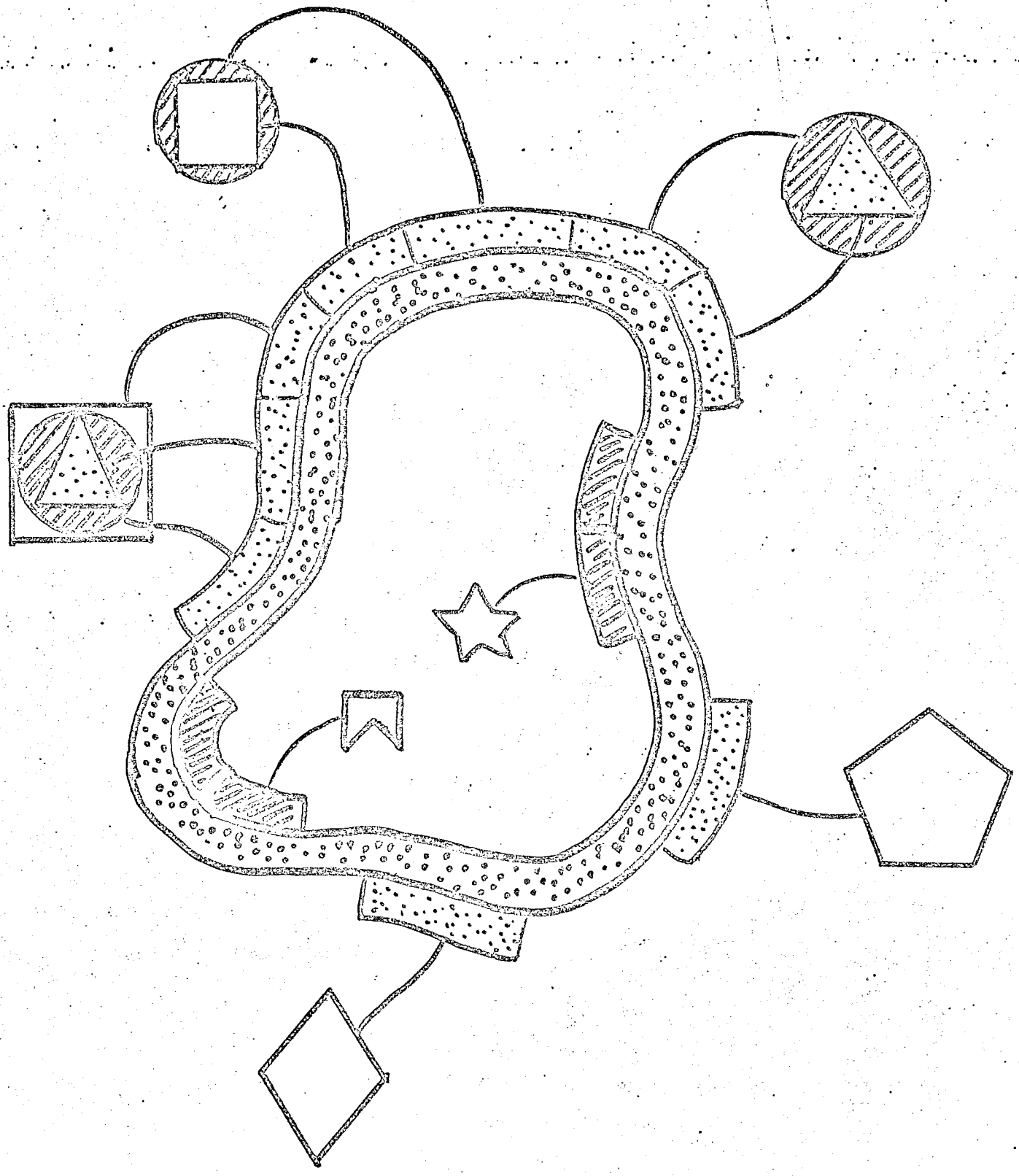


Figure 7.

D - Network Authority

It certainly could be read as a miracle if every component of a communications system would belong to a different organization, and if the service were satisfactory with no one responsible for it. Myriads of operational problems require a well defined responsibility vested in an organization we may call the Network Authority. It is not necessarily the owner of every piece of equipment, but it is assigned the mission of seeing to it that the whole system works properly. The extent of this mission may vary : typically it would cover the following areas :

1. Maintenance :

Namely error tracing and repairing. This is a tricky task, as it involves equipment from various suppliers, including lines leased from common carriers. And it is positively undesirable to bring the communications system to a complete halt, just to find out which component needs fixing up. Interventions must be done on a live system, with appropriate care.

2. Accounting :

Exchanging data costs money. Again, it would not be very satisfactory to rely upon individual installations to declare the amount of traffic they have been generating. On the other hand, the Network Authority is in a good position to record traffic figures suitable for later accounting and billing.

3. Traffic analysis :

Communications networks are far from being completely understood. From mathematical theories to practical cases, numerous studies are presently being pursued. In order to validate or feed in models, traffic patterns must be recorded and sorted out. Furthermore, due to the individual freedom of installations, traffic patterns may change rapidly. A continuous analysis is necessary to anticipate in due time configuration changes and tune the system to its actual requirements.

4. Development :

It is an obvious extension of the responsibility for maintenance and traffic analysis. Developpement is probably best assured by the organization which has the best knowledge on inner workings of the system.

5. Clearinghouse :

Independent from the cost of communications, installations provide services to one another, which is liable to some cross-billing. Rather than having each one to send invoices to every other, it is more convenient to consolidate all accounting in a single location, and apportion each one's balance sheet to its total activity. All the more that some tariffs may be tied with the amount of traffic exchanged between installations. Although the Network Authority is not the only place in a position to perform this task, putting it there is probably more convenient.

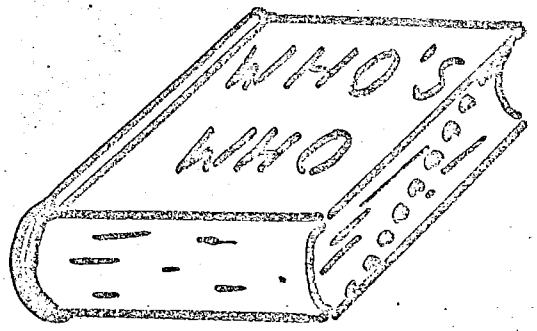
6. Network control center :

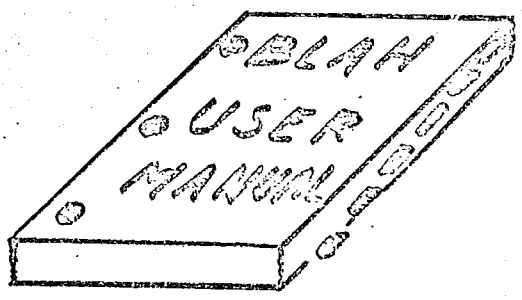
Most tasks entrusted to the Network Authority cannot be accomplished without computerized aid, such as running diagnostics tests, monitoring traffic, cross-billing, etc... For that purpose a computer installation is needed, and it can be linked to the communications system on the same basis as any other installation of the network. Owing to its obvious privileges, this installation can be considered as "internal", in the sense used previously.

E - Information centers

The diversity and the geographical distribution of resources in a computer network create the need for various information repositories. This belongs to network sociology, rather than technology. Indeed, like in any association conducting business, some media are needed to facilitate an encounter between supply and demand. This can take the form of ads, news, manuals, indexes, mail service, broadcast, etc... Probably at some point in the future the video-technology will allow live meetings to take place over networks, and be canned in play-back files. An example of such a center is developing in Arpanet at Stanford Research Lab. Some of these services might be provided by the Network Authority, but other specialized installations may do it as well. Ideally, it should be thought of as set of coordinated interactions between installations, and again some dominant patterns are likely to match the partitionning into clubs. (Fig. 8)


NEWS


Hello's
Hero


BELL
USER
MANUAL

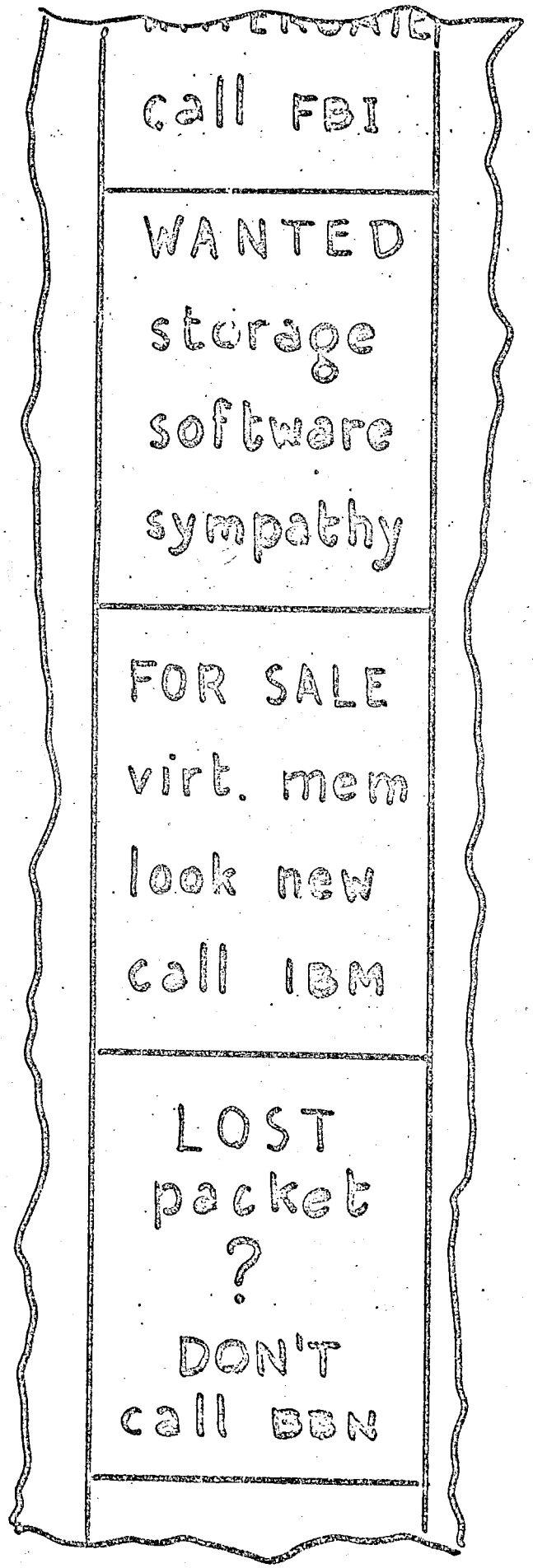

call FBI
WANTED
storage
software
sympathy
FOR SALE
virt. mem
look new
call IBM
LOST
packet
?
DON'T
call BBN

Figure 8

III - CONCEPTS AND FUNCTIONS

A - Data switching machine

Various techniques may be considered when it comes to set up some communications system between computers. What they have in common is that they must comply with some government regulated monopoly, and they require existing communications facilities to be ruled out, but the initial cost would be unacceptable, unless intense pressure is put on, like in a period of war.

It is not the intent of this paper to discuss communications techniques. Let us remark only that computer input-output is message oriented, in digital form, and that existing communications facilities have been engineered for something else than data communications, (voice, telegrams, television). More recently a technique called "packet switching" has been proposed and experimented, (see Dr. Davies' paper). It seems to make the best out of existing technologies, and it can be supported by telephone, radio, or satellite transmission. Observing the trend in networks shows that transmission systems evolve towards a message store and forward technique with short messages, and short transit time, in order to meet the requirements of conversational traffic. With mini-computers and the availability of high speed lines, it looks as if the day of packet switching has come, at least for a decade, till new transmission systems are widely offered. In the following, assumptions made about communications systems will be restricted to packet switching.

As we have seen before, we want a communications system independent and insulated from computer installations. Consequently, a packet switching system has to be a computer network in its own right, but a highly specialized one. What it has to do is to switch short messages (packets) on the best possible way, i.e. fast, efficient, reliable. Of course, each of these adjectives might take pages of discussion. Since every other additional service is to be implemented in some internal or external installation, the communications network should do no more than packet switching. Externally it looks like a black box to which messages are handed for delivery to a specified destination. (Fig. 9)

The internal structure of a communications network could be looked at conventionally as reflecting its physical structure, i.e. a set of computers exchanging messages, to carry either data or control information. There is nothing invalid in such a model. But some concepts, in addition to represent a physical structure, have also the power to expand and generalize the model in a way that brings new light, sets guidelines, and eventually helps re-thinking the structure itself. Considered as a black box, a communications network can be viewed as an abstract machine, which as usual has an instruction code, processors, I - O, and internal states. Passing a message over to a communications processor (node) is tantamount to inputting an instruction (op. code + operand). Executing the instruction is in effect transferring the message into another component of the machine, and output it. In a short-hand form this could read :

input A	input message
B := R(A)	evaluate destination
(B) := (A)	transfer message to destination
output B	output message

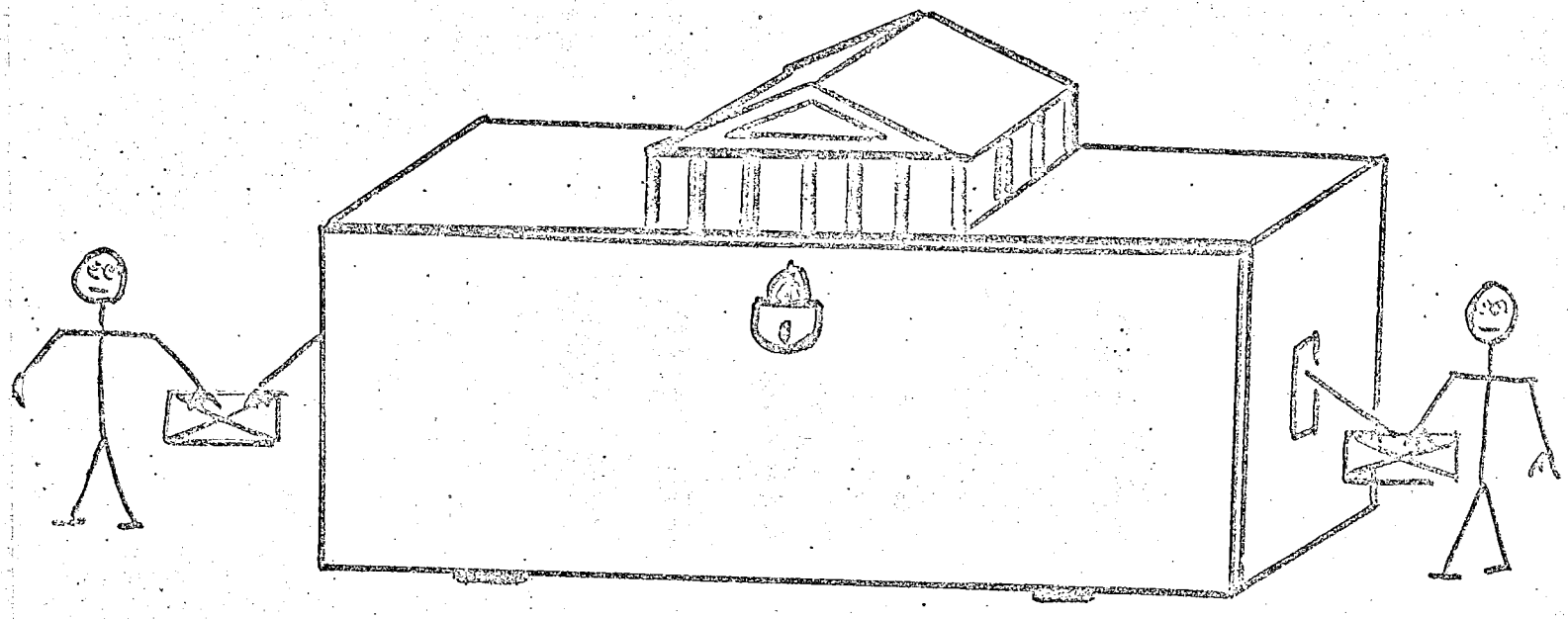


Figure 9.

Some instructions may spawn several outputs, if they fail (e.g. R(A) undefined), or if some options are exercised, like tracing the transfer path. Internal states are mainly used for internal control of the machine (monitoring, maintenance) or as a result of failure diagnosis. Other instructions may act upon internal states, which can be thought of as switches, special registers, or panel lights, in a conventional computer.

Describing a communications network as an abstract machine is a way to draw a line between its functional specifications, as visible from an external user, and its physical implementation. Of course, it would be non-nense for a designer not to have precise ideas about the mechanisms underneath. But as far as possible, inner workings should be kept invisible at the functional level, because computer installations on one hand, and the communications network on the other hand should conserve maximum autonomy.

Opening the black box discloses a new dimension, (Fig. 10). Our abstract machine is actually composed of a set of components, say processors, which must be put to work in a coordinated fashion to get a message through. In other words, executing an instruction of the global machine is a multi-processor action. Then it might seem that well known process coordination techniques would apply nicely. Actually they don't. Although we are dealing with an abstract model, we are nevertheless in a real world. There is no common store, no indivisible operation, no well-mannered processes. Components are geographically scattered, and they may fail. In this machine there is a communications problem between its own components. We are facing a more general case of multi-processor, i.e. a distributed machine.

Rather than attempting to devise a complex model of a distributed machine, an easier and more powerful approach consists in breaking it down into simpler components along its geographical distribution. Each node is again a local machine, whether it be uni or multi-processor. Its abstract model is identical to the global one, with an instruction code, I-O, etc... Messages are instructions as previously. Executing an instruction of the global machine appears now as a sequence of instructions on local machines. Each one performs a step of the global instruction, and outputs a result towards a neighbor machine. The process propagates up to a point where output is directed to the external world, and there it stops. (Fig. 11)

Actually this process can be other than a sequence of local machine instructions. Indeed, components may fail, particularly the communications lines, and a local instruction may be repeated a number of times, with different results, because internal states may change. Thus, outputs may be sent to various neighbors, and the set of local instructions becomes a tree, with loops and branches executing in parallel. In the end there may be one, none, or several results. It is clear that some error control scheme is needed to make our machine reliable.

This global instruction moving step by step through successive local machines is reminiscent of the pipe-line structure of high speed processors, or time-slotted multi-processors (CDC 6000 peripheral processors). The similarity is not accidental. With very fast and cheap micro-components, future machines will contain hundreds or thousands of processors. They will be micro-networks.

As seen previously, the local machine model is the same as the global machine. Our structure is recursive. But only one level of recursion is not much after all. Let us carry this approach one step further, by applying the distribution concept

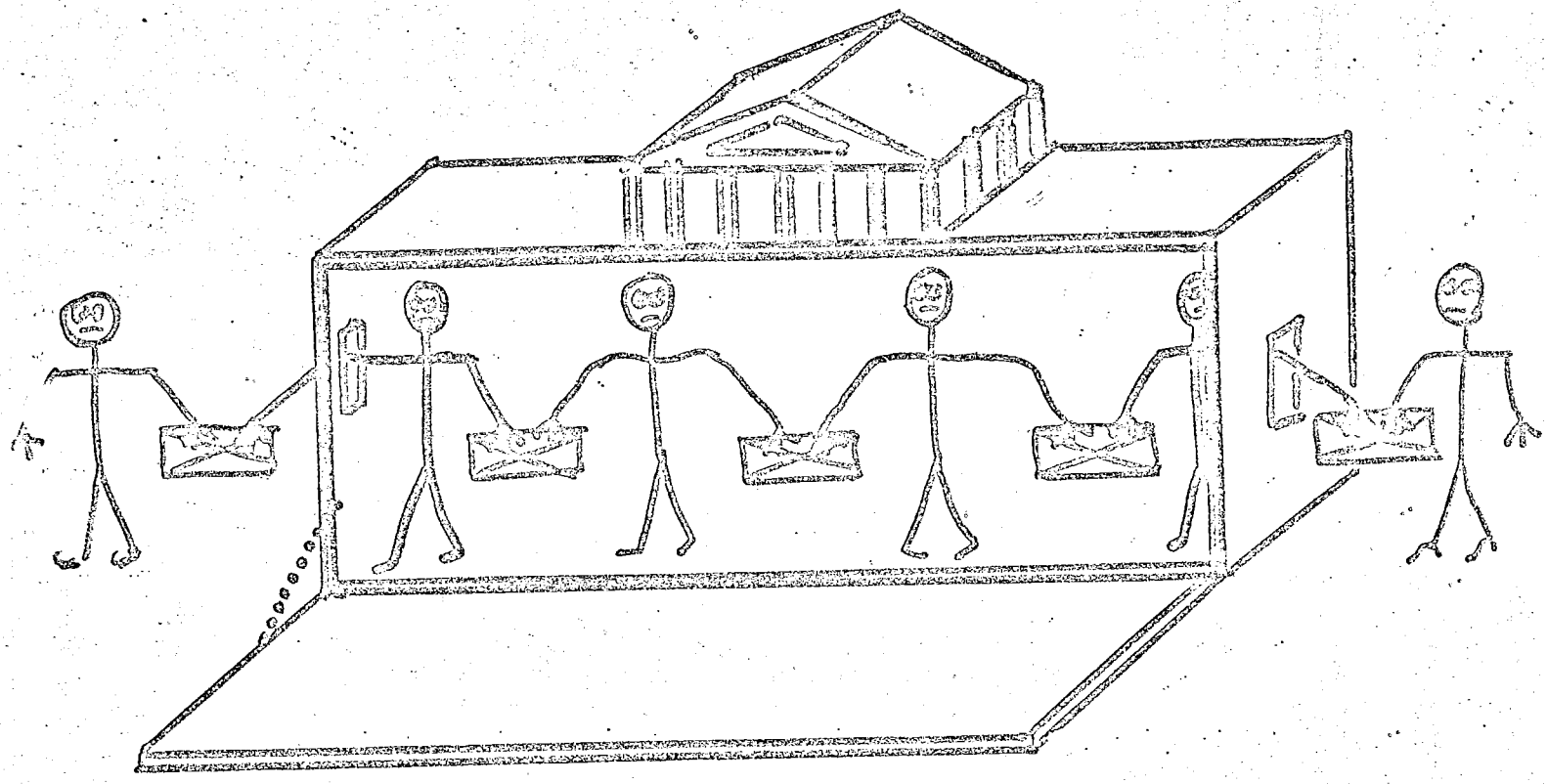


Figure 10.

INSTRUCTIONS

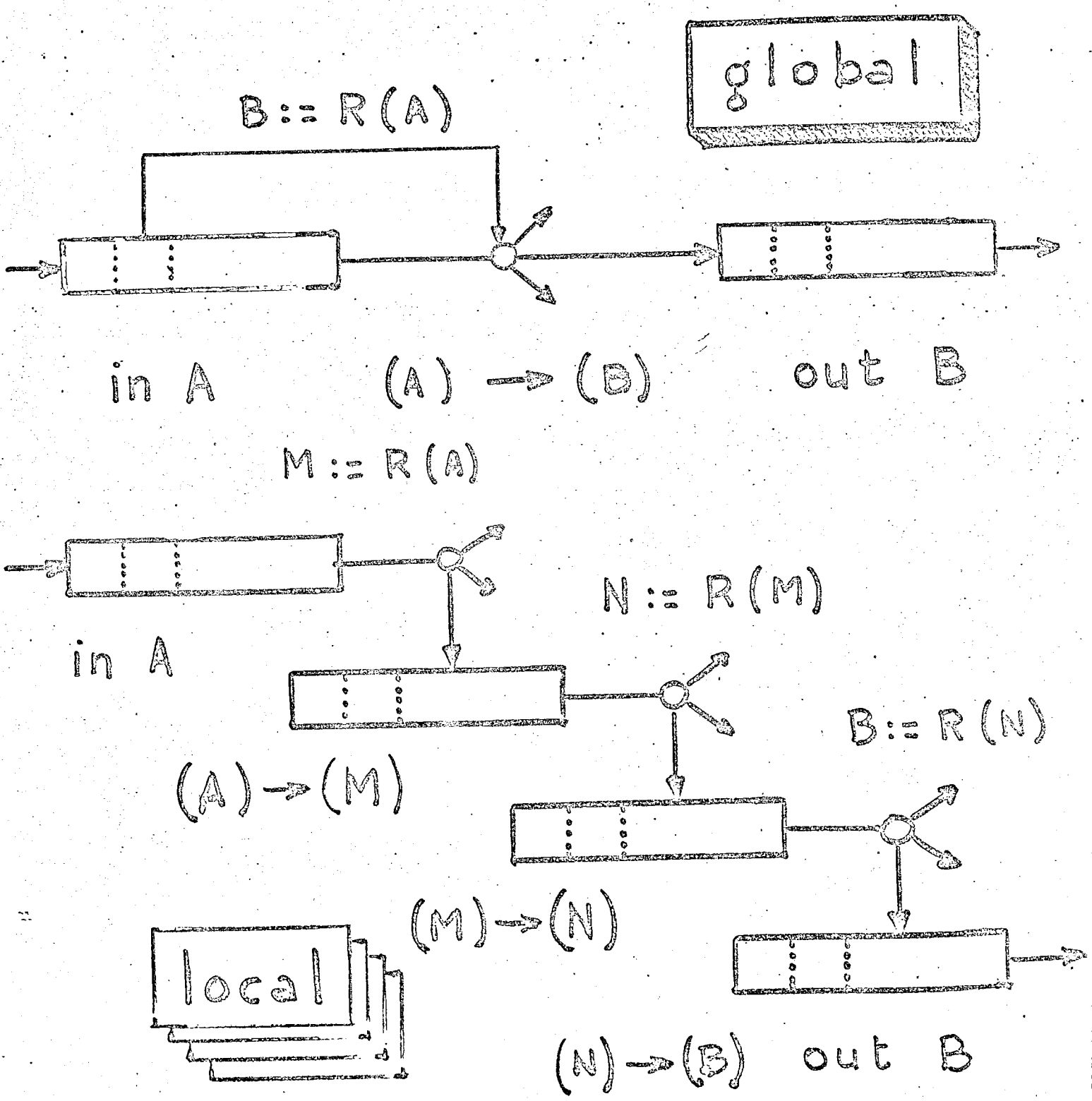


Figure 11.

to the instruction code. In conventional machines, the instruction set is closed, due to the number of bits reserved for the op. code. In our network machine we can consider that each instruction type is executed by a logical component, located somewhere in the global machine. An op. code is then simply the address of the appropriate instruction processor. Executing a global instruction in a first phase propagates it toward its specific processor, where it is executed with possible further propagation. By putting logical network components within the destination name space, the instruction set can be as large as desirable without practical limits. In other words, the instruction set is open-ended. This is a nice feature for an experimental system. The network machine is now defined in terms of logical (software) components, instead of local (hardware) machines. Practically these components have to be physically located within some or all nodes. Consequently, the same structure applies as well to a local machine (node). (Fig. 12)

A consequence of this regular structure is that communications between logical components (software machines) is the same whether they be geographically distant or located within the same node. But there is a difference in delay.

Another consequence is that several networks of that type can be put together and make up a single network. This is the recursive structure property applied outwards. Indeed, every network can be modeled as a node, hence several networks are a network. Partitionning and coalescence of networks may occur during normal operation as a result of line failures. Therefore this sort of self-adapting structure is most desirable, as it saves on down-time and service disturbance.

B - Hosts

So far we have used the term "installation" to mean vaguely a computer of the set making up a heterogeneous general-purpose computer network. This requires further attention, as we know that computers can reveal various metamorphoses. Since the Arpanet literature introduced the term "host", it has been used widely, although not always in the very same sense as Arpanet. For the sake of consistency, we use it here also, in some loose sense, as we shall see.

As seen from the communications network, a host is a source and a sink of messages. It is able to input and output messages of a predefined format, and it has an address, i.e. it is known within the name space of the communications network. This is enough, but one can notice that not the slightest assumption is made about the logical or physical nature of a host. This means that the structure of a computer network is totally independent of that of a properly designed communications network.

At the host level, discounting some special cases, no one is interested in a computer per se, because it is not a usable logical entity. Usable entities in a computer are resources : files, processors, peripherals, user accounts, etc ... In order to get some productive work, activities are run to which a sub-set of resources are assigned, exclusively or in some shared mode. Customarily, computers are used in a multiplexed fashion, allowing several activities in parallel. Thus a host is merely a container of resources and activities.

In a computer network environment, useful host-host communications actually occur between activities, in order to access remote resources, or set up distributed activities. Due to the heterogeneity of the hosts, there is no common or compatible definition for a local activity. On the other hand, communications between hosts cannot take place unless some mutual agreement has been agreed upon. To get around this dilemma, we introduce a new kind of network-wide entity called a subscriber. They are purely abstract things, bearing a name known by every host, which activities can plug into when they want to start some host-host communications.

Data switching machine

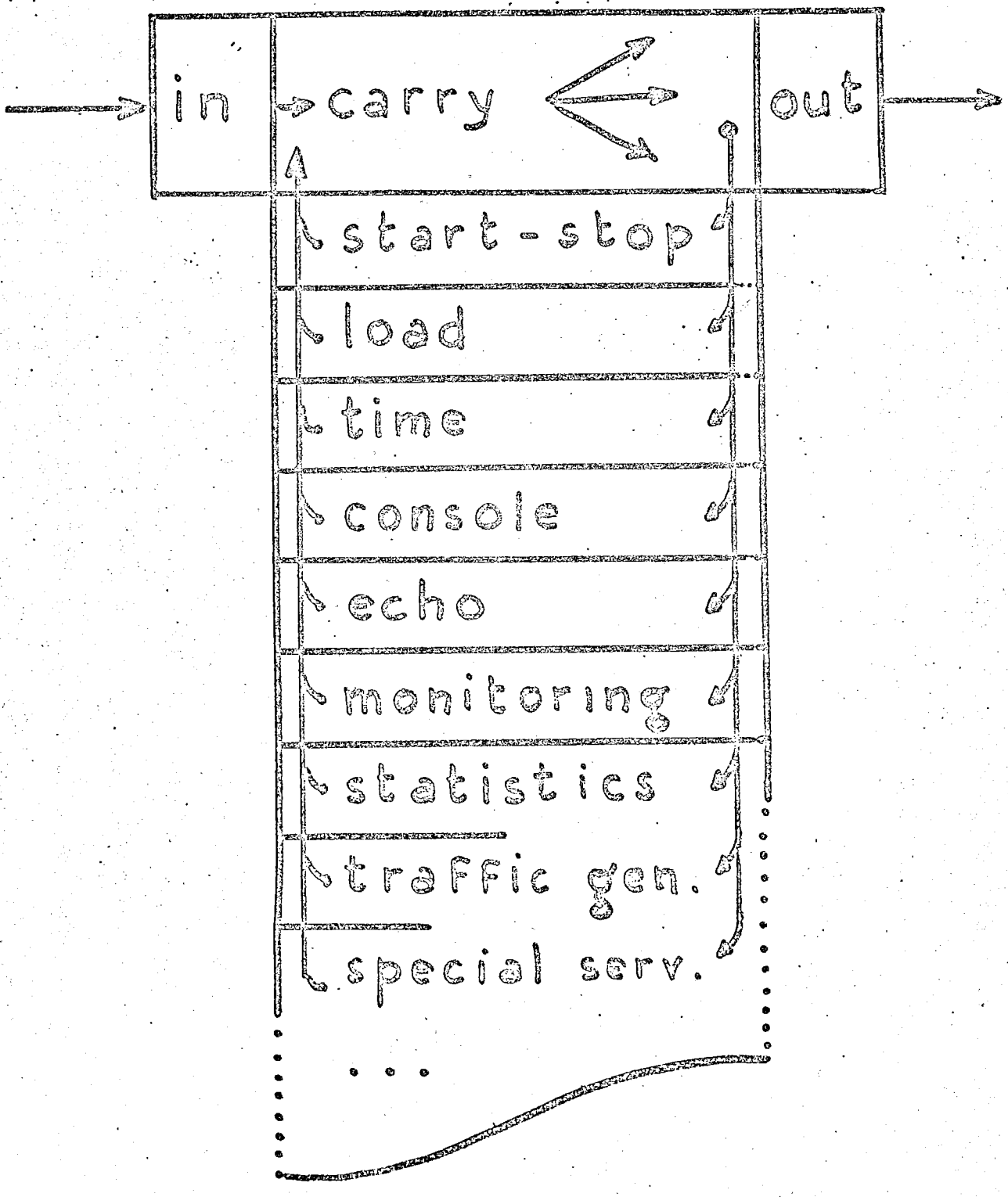


Figure 12.

In many ways subscribers are like phone sets. (Fig. 13)

The number of subscribers per host should be large enough to accomodate the needs of simultaneous network activities. But if we want to avoid intricacies and inconvenience associated with a dynamic assignment, subscriber names should be assigned permanently to real world entities such as users, terminals, services, etc ....

Usually, all messages exchanged between a host and the communications network are multiplexed onto the same transmission lines, typically two for reliability. Thus a logical component is necessary to collect and deliver messages fro and to the proper subscribers. It is typically a host operating system module, of the access method species, implementing "the" network protocol.

However, a unique protocol is quite constraining. New versions must be tried out occasionally, user clubs want their own protocol, an experiment is contemplated with a host in a different network, and for a handful of other good reasons, a host should accomodate various protocols, which may or may not be compatible. This results in a more sophisticated structure with two levels of multiplexing, one for protocols, one for subscribers.

If messages received by a host must be fanned out to the proper protocol, this means that some general convention has been agreed among all protocols about the selection algorithm. This would not be too realistic, as protocols may just happen independently, and it would be in contradiction with the concern for installation freedom. Consequently, we use a different scheme. Within a host there can be several virtual hosts, each one with its own set of consistent protocols. Incompatible protocols are in as many virtual hosts, for which there are different addresses in the name space of the communications network. Thus, messages are dispatched according to their virtual host destination within a single host. (Fig. 14). This technique applies as well when a host is composed of several computers hooked together, or when a host is operating under a virtualizing system like CP-67.

So far there has been no assumption about the number of transmission lines used to connect a host to the communications network. It has only been mentionned that two would be appropriate for reliability. It is not said either that they should go to the same node. For reliability it is better to connect to different nodes, but this is immaterial as far as the host is concerned. However, it matters to have several lines, particularly if the host is a distributed system, i.e. a network. In this case it is very likely that several paths should be established between both networks.

To recap, it appears that the concepts of host and subscriber are flexible enough to elude any precise definition, except being names. Practically they can be associated with a variety of objects. Let us say that a host is a container and a supplier of resources, while a subscriber is usually a set of resources intended for an activity. E.g. a subscriber can be a file, a terminal, a user, a sequential process, a sub-system, an operating-system, a virtual machine, etc... Some entities might as well be considered hosts or subscribers, depending on convenience.

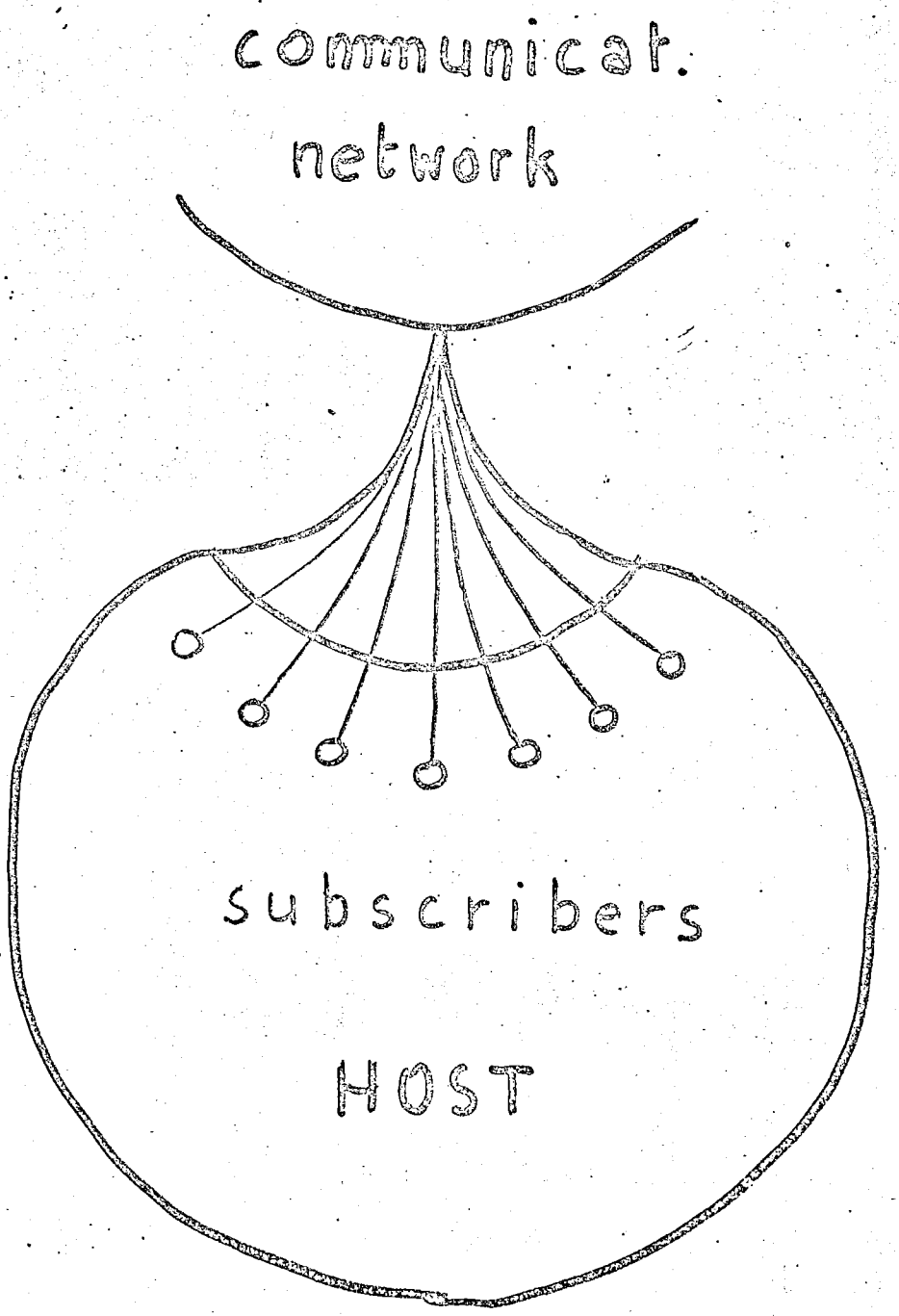


Figure 13.

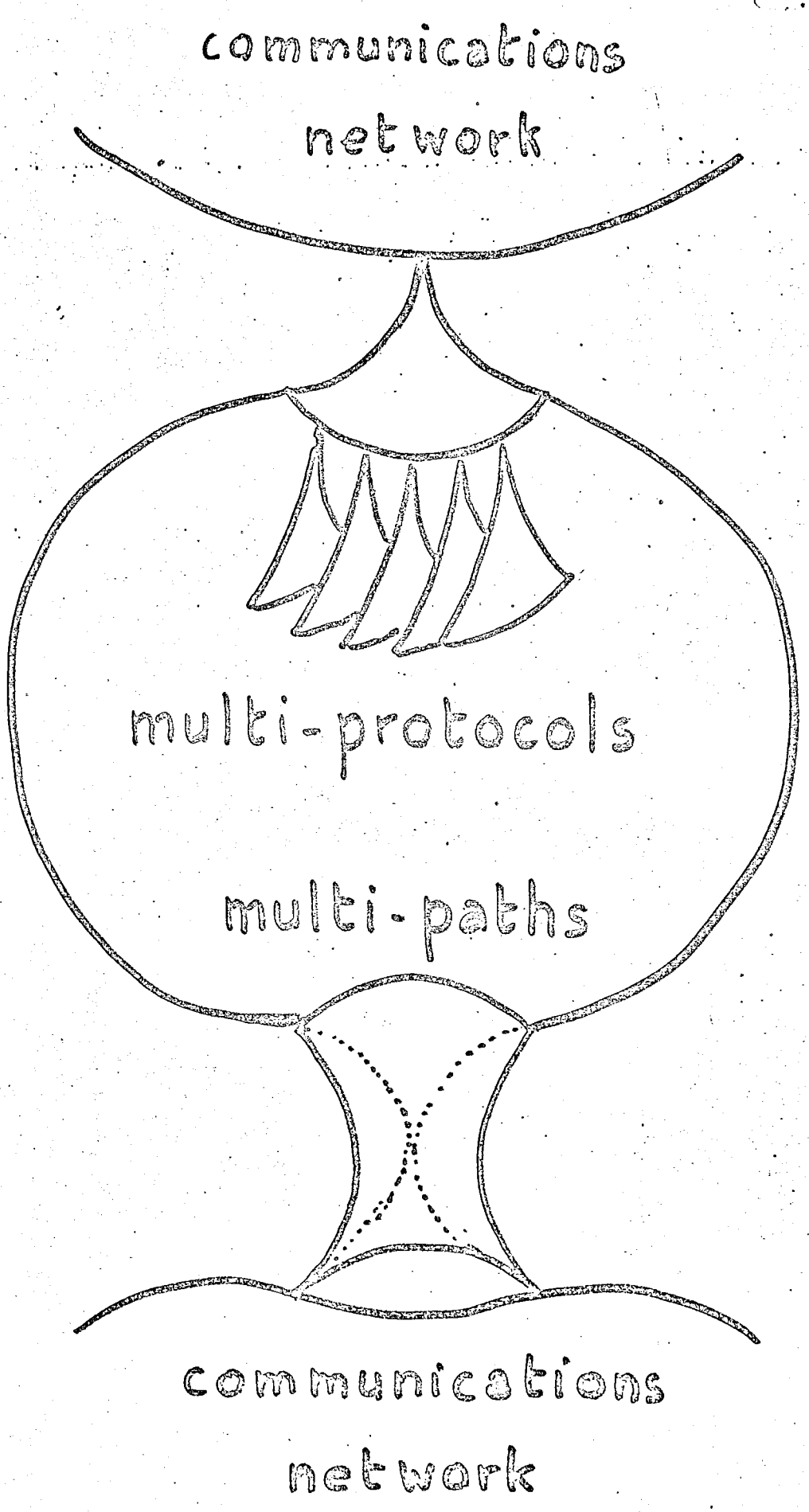

communications
network
multi-protocols
multi-paths
communications
network

Figure 14.

Using distant resources, communicating with distant processes require naming them. But since hosts are heterogeneous, there is no consistent scheme to name distant resources. Furthermore existing names may conflict. A first attempt to circumvent the problem is to define a new name space global to all hosts. This may be the case for subscriber names. Therafter, every host organizes on its own way a mapping of its local names onto the part of the global name space allocated to it. But this means that one must know beforehand how a distant host maps its local names in order to access its resources. Again, this is heterogeneous.

A second technique is to use a hierarchical name space. Names are 2-component, one is a global name (host, subscriber, or any other), the other is the local name designating a specific resource in the host naming scheme. This method is usually more costly for computers, but more convenient for human handling.

There is a dilemma in networks when it comes to locate an entity from its name. Due to delays inherent in communications, it is costly to search for a name in several or even all hosts. Consequently one tends to make up names with a host name component, which serves as a geographical pointer. But this scheme is not well suited to resources which belong to virtual or distributed hosts. There is more flexibility if resource names are independent of host names. But to reduce search time it is desirable to reinstate a geographical locator, which we may call a region name.

Regions can be defined with the objective of cutting through a minimum traffic, so that region and traffic clusters would roughly match. In addition, within the confines of a particular region there is no need to use global names, since the region name can be implicit. Both effects would compound to make most names shorter, hence more convenient.

Other more sophisticated variations may be considered, in order to simplify naming across the network. They are inspired from dialing schemes in the telephone system, which are quite interesting indeed. See a phone directory for more details.

D - Communicating entities

In a message oriented communications network sending individual messages is a basic capability which does not require any initial set-up. This can be put to an advantage in a computer network when cooperating activities exchange short self-identifying messages, e.g. transactions, samplings, control information. In this case, the only entities required at host level for activities to communicate are subscribers. Once an activity has a subscriber name assigned to it, it can just send messages to any other subscriber in the network under the same protocol.

But more traditionally, interchanges between activities tend to be modeled after sequential I-O. It is a transposition at network level of programming systems designed for sequential devices. In effect most inter-process communication schemes are data stream oriented, stemming from an I-O-minded approach. Thus it is convenient to have mechanisms at host level geared to data stream handling. For that purpose new kinds of entities are required, to provide something like data pipes between activities. There are a number of possible variations in the functional design of such tools. Also the terminology is not stabilized. In the following we designate by pipe such an entity capable of channeling a data stream between two distant activities, and ports the names used at each end by host activities. Here are some typical pipe properties which may be encountered.

Uni - or bi-directional :

Since a data stream always requires some feedback for error and flow control, it is most convenient to have bi-directional pipes.

Half - or full-duplex :

On bi-directional pipes full-duplex mode is much more simple and efficient, as it does away with the contention problem that plagues half-duplex procedures.

Sequential :

This is the property of delivering messages in the same order as they are put into. For sequential data streams, sequencing is mandatory. But a pipe may be used by a sub-system which contains its own scheme for tagging and controlling the message flow, in which case sequencing is not necessary.

Error control :

It is the ability to insure that every message sent has been received only once. It can be a by-product of sequencing. It is not questioned that error control is needed at some point when data travel from host to host. But where to put it is an issue far from settled. There is more elaboration on that in the following.

Flow control :

It is a mechanism whereby the receiver can choke up the sender to prevent overflooding. Same comments as for error control.

Parameters :

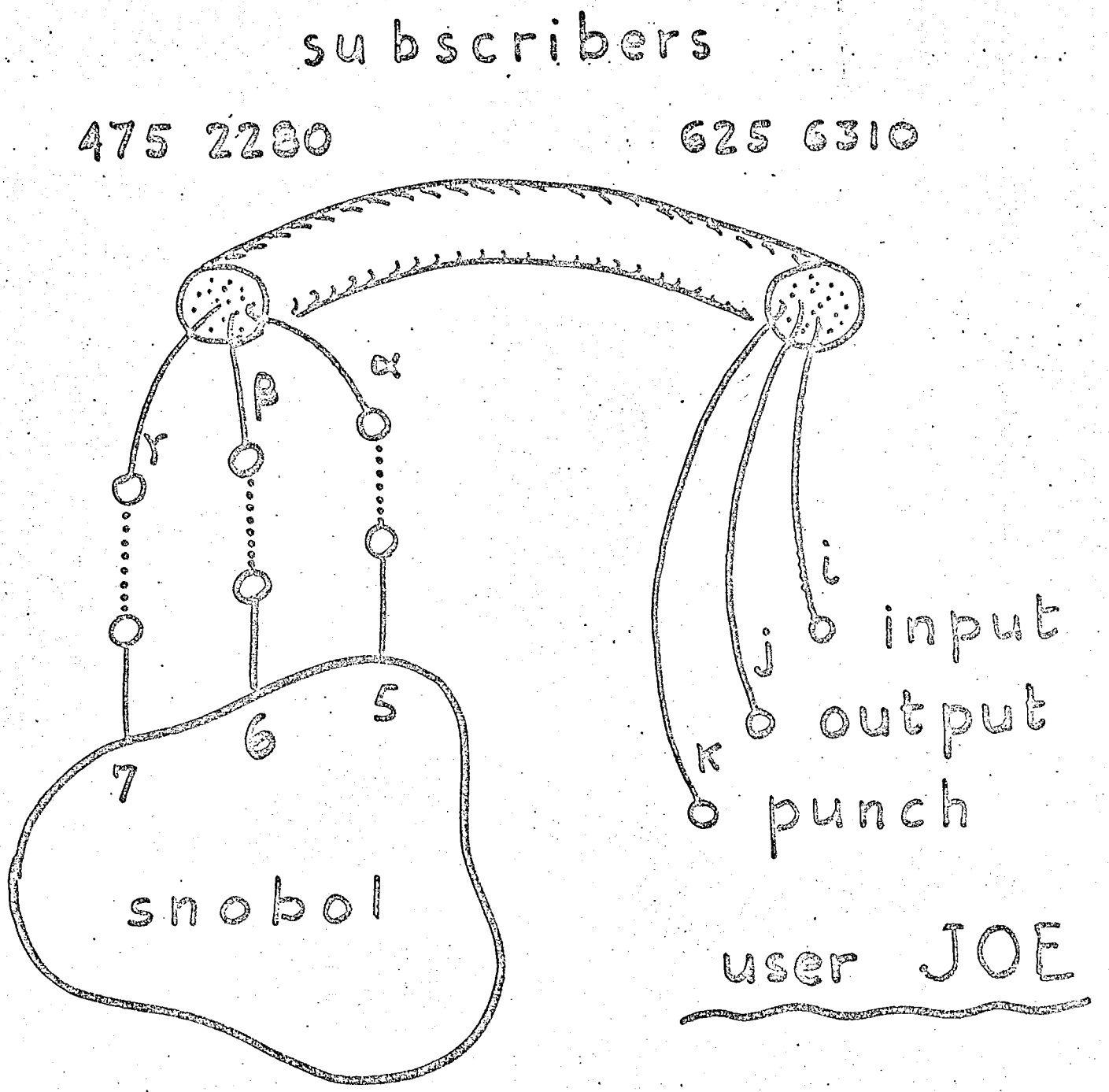
To regulate the data flow and tune it to the amount of resources available to both sender and receiver, it may be appropriate to associate parameters such as : message length (mini. + maxi.), traffic rate, time-outs, etc... These parameters can be negotiated between both parties as part of the pipe initial set up.

It should be emphasized that pipes are only a construction implemented at host level. In some cases they are extended end to end through the communications network. The implication is a strong coupling between the design of host protocols and communications network. This may be justified for specific applications (e.g. Tymnet, a service bureau time sharing network), but in the general case, this is in contradiction with our principles of insulating the two domains, due to the undesirable constraints attached.

Data pipes are to be set up (opened) and destroyed (closed) at the request of host activities. As one can figure out easily, it would be a tricky problem if both activities had to agree on a common pipe name, because there is no referee to break a tie up. It is more convenient to name a pipe from both ends with local names as seen from each activity. These are port names. Each activity refers to a port when sending a message. But this is its own local name. In order to deliver messages from the appropriate pipe at the other end, they must arrive with the destination port name. Hence, port names of both ends must be exchanged between hosts during initial set up.

Pipes and ports are particularly useful when host activities are programmed with I - O streams left unassigned to any specific device, and can be referred to by labels, say by port names. Using a command language, or some form of declarative statements allows a user to bind network and program port names at execution time. Depending on his requirements, a user is able to use either local or distant files and I - O devices, just by assignment commands. (Fig. 15)

PIPES



PORT names

Figure 15.