XEROX SDD ARCHIVES
I have read and understood
Pages________To________
Reviewer________Date________
# of Pages______Ref. 78SDD-187

XEROX
BUSINESS SYSTEMS
Systems Development Department

To:	Distribution	Date:	22 September, 1978
From:	Yogen Dalal	Org:	SDD/SD System Architecture
Subject:	Network Addresses and Routing	Filed:	[Iris] < Dalal > memos > NetworkAddresses.memo

The notion of a *network address* is fundamental to the operation of the OIS communication system, and the architecture of the distributed system that uses it. A network adddress is the basic description for the source or destination of OIS packets, the transmission of which, over an interconnection of networks, provides communication among distributed processes. A *network address* as currently defined is a triple *(networkID, hostID, socketID).* HostIDs are unique over all space and time, since every system element is guaranteed to have a unique one. SocketIDs are unique within a host, and their allocation is the responsibility of Pilot. The tuple (hostID, socketID) is, therefore, sufficient to uniquely describe a source or destination of packets. If the distributed system is small, then the routing of packets through an interconnection of networks can be achieved using this tuple. However, in order to reduce the state information needed to permit routing in large topologies of interconnected networks, among a large number of hosts, the networkID serves as a *very strong hint* to the actual location of the host. Since a host can be connected to more than one network, a unique source or destination of packets can have more than one network address, but a network address cannot describe more than one source or destination. This memo discusses the many different issues related to networkIDs, hostIDs, socketIDs, and the use of network addresses in the architecture of the OIS distributed system. This memo also proposes solutions to some of the problems discussed.

A *network* is a communication medium connecting together two or more system elements, and providing a communication path among *all* of them. An *internet* is an interconnection of networks.

A network has a *networkID* and a set of properties. The network provides point to point, multicast (multiple destination) or broadcast routing, has certain error characteristics, and defines a protocol hierarchy used to encapsulate OIS packets as they traverse the network. The protocol hierarchy can consist simply of a line control discipline, or complex rules for sequencing, fragmentation and reassambly of data. Each network has a *local* addressing scheme (used by the encapsulation protocol) by which data is delivered from source to destination, within the same network. If a network's local addressing scheme is different from the OIS hostID, then the hostID must be translated into the local address for transport through the network. Examples of networks are Xerox Wires, Ethernets, leased lines, the Bell Telephone System, Telenet Communication Corp., SBS etc. This memo discusses the size of networkIDs, mechanisms for assigning them, and techniques for making them known throughout the distributed system.

A *hostID* is 32 bits long and we assume without further discussion that they are unique within OIS. Their uniqueness is guaranteed for D* system elements by Xerox during manufacture, and for non-D* system elements by careful administrative procedure.

A *socketID* is 16 bits long and is unique within a host. Its uniqueness is guaranteed by Pilot. A small number (~1000) of these socketIDs are *well known* and constant on every host. These socketIDs are used only by the OIS communication system, and their values are assigned statically by SDD. Pilot will assign socketIDs to client software, and guarantee that they are unique since the time the machine was last booted.

Users and Servers Network Addresses:

A distributed computing system must permit resources to move from machine to machine if need be. Similarly, it must be able to determine things like "where is a particular person currently working?". Generalizing this, the system must permit the binding between *keys* and *values* to be dynamic. As a consequence, binding between keys and values is done at run time through a data base which we call a *clearinghouse.* This feature is not only useful for the distributed computing system, but for the OIS communication system as well, since it must be able to determine the networks on which a particular host is connected, i.e. determine another network address for a unique tuple (hostID, socketID). In this example, the unique tuple (hostID, socketID) is the key and the network address(s) is the value.

For the time being let us assume that the OIS communication system can route a packet to any network address. We will return to the problems it faces when attempting to do so. We now examine the problems of how a resource gets a unique identity and network address, and how the (resource identity, network address) pair, (i.e. the (key, value) pair) is made known to a clearinghouse.

A distributed computing system typically consists of resources accessible through servers, and users of these resources. Servers have unique, well advertised, *identities* assigned to them through some mechanism, the exact details of which are not necessary for this discussion, but one similar to that used for unique *fileIDs* in Pilot will do. The OIS communication system is able to route data to a server once its *address* is known. The binding between the identity of the server and its address is provided by the clearinghouse. As stated earlier, it is sufficient that this address be only the tuple (hostID, socketID) at which this server accepts requests, or the tuple (hostID, socketID) through which this server can be approached (server dispatcher or RPCP listener). If the networkID is included in the address then the network address defines a partial path, or route to that server. Should this path be unavailable the OIS communication system may have an (adaptive, heuristic or table-lookup) algorithm for computing a new one. In addition, the source might be informed of this change so that it uses the new network address rather than the old one, thus reducing the overhead in the communication system, which in general does not keep much history of data transmission.

From a server's point of view, when it *first* gets created, it must be assigned a unique server identity, and then must find out its network address and register this (server identity, network address) pair with the clearinghouse. There may be an additional *name* associated with the server in which case there must be a (name, server identity) binding that is also stored in the clearinghouse. Hence, it is conceivable that two clearinghouse lookups are performed before the address of the server is known. For example, a file server may have the name Iris, a unique server identity 234519, and a list of network address 12#234#2789#, 34#234#2789#, 321#234#2789#.

If the address of the server did not contain the networkID, then a third clearinghouse lookup is performed by the communication system in order to find out the network number(s) for that hostID. The routing machinery then attempts to deliver the packet, possibly with recomputation of the network address.

A server determines its address (network or otherwise) from Pilot each time the machine is booted. The hostID is the number of that particular system element, and the socketID is the value that Pilot decides to assign this time. *What about the networkID? This memo attempts to answer this important question.* The server must, therefore, store its (server identity, network address) pair in the clearinghouse each time the machine it resides on is booted.

A user knows the name or unique identity of a server (by consulting some directory of resources) and determines the network address of the server by querying the clearinghouse. Mechanisms by which data is entered into the clearinghouse, updated and read is out of the scope of this memo, although some techniques may be inferred from this discussion.

Network Number Assignment:

NetworkIDs are 16 bits long. Every network in an internet must have a unique networkID, otherwise the routing mechanism will not operate correctly. This scheme allows networks in two separate independent internets to have the same networkIDs. The problem only arises when two independent internets merge. NetworkID assignment is accomplished by an SDD *czar* using some well formulated principles:

1. The networkID space is partitioned into *chunks* which are multiples of some basic size.

2. Public data networks have networkIDs assigned from one chunk, and their networkID is the same in any two *independent* internets.

3. An internet consisting of only one network (not a public data network) has a networkID of zero.

4. Internets which will be independent (with extremely high probability) may be assigned overlapping chunks for their networkIDs.

5. Internets that may merge are assigned unique non-overlapping chunks.

This is a simple scheme for assigning networkIDs. The exact details of chunk management and assignment are out of the scope of this memo. A technique by which networkIDs are propagated to the system elements is now discussed.

A system consisting of one network (that is not a public data network) has a network number of zero. System elements need not discover the ID of the network to which they are connected, and the network addresses of all objects is relative to this network. System elements will, in general, not know that they are connected to a network whose networkID is zero, and so they will try to discover their networkID (in a manner described below) and, upon failure, they will assume it to be zero.

Now consider how a large internet is first initialized. In order to examine this in detail let us consider an internet topology consisting only of Xerox Wires, since there are many other problems associated with the introduction of other communication media. The installation manager must install (initialize) each *internetwork router* before installing any of the other system elements on that network. Installing an internetwork router consists of (among other things) assigning networkIDs to the different Xerox Wires connected to the system element. *Mechanisms by which the installation manager can identify a particular Xerox Wire interface and assign it a networkID must be determined.* As other system elements are added to the networks, the routers in them will attempt to discover the networkIDs by asking the internetwork routers using a broadcast protocol. Routers must take care when determining the networkIDs, and a sufficiently long timeout must be used, so that a networkID is deemed to be zero only if it really is zero! If a system element is connected to more than one network (it need not be an internetwork router), then servers requesting a unique network address will be assigned by Pilot a list of network addresses--each one differing only in its networkID. What happens in the case that a system element is not an internetwork router but is connected to more than one network? Are the networks unusable until their networkIDs have been determined? What network addresses are assigned to clients that request them prior to determination of networkIDs?.

The techniques described above also works in environments consisting of other types of communication media in the internet topology, as long as *all* the system elements that are connected to communication media other than Xerox Wires or Ethernets function as internetwork routers. This is because system elements that are not internetwork routers can still easily determine their networkIDs by using a broadcast protocol. We postpone the problems introduced by connecting system elements, which are not internetwork routers, to any communication media other than Xerox Wires until a later section.

There will be some environments with many Xerox Wires and other communication media interconnected from the start, while there will be others which start from one or a few wires and then incrementally grow. In the case that two internets with networkIDs from unique non-overlapping chunks merge, there is no real problem other than merging clearinghouse data.

The problem arises when the chunks overlap--as is the case when two or more independent Xerox Wire networks merge, or when two or more internets with networkIDs that overlap merge. The networks in the merged internet must have unique networkIDs (except for any public data networks that they share). *A consequence of this merger is that the entries in the clearinghouse must be updated to reflect new network addresses, and the servers must also become aware of their new addresses.* In addition, we assume the existence of mechanisms for merging the information from two independent clearinghouses. The different ways by which networkID uniqueness might be maintained are:

1. The installation manager that is performing the merge renumbers all the networks manually. System elements learn of their new network numbers. Entries in clearinghouses and any caches need to be updated appropriately. Processes on system elements may need to be told that some of the network addresses they know about are changing. Servers will typically not have to be told since their network addresses have been updated in the clearinghouse, and they will determine new network addresses on reboot.

2. The internetwork routers that connect merging internets serve, in addition, as *merger* routers. Such a router performs the additional task of networkID translation. Each of the original internets is seen by the other internets participating in the merge as *one* super-network with a networkID that is *unique* with respect to those in all the internets that are merging. Clearinghouse entries of objects in one internet as seen by other internets must have the networkID changed to reflect the ID associated with the super-network. Routing and networkID translation takes place as follows: When a packet arrives at a merger router, it examines the networkID and determines if this corresponds to one of the directly connected super-networks. If so, it examines a data base, provided by the installation manager, to determine the networkID relative to this super-network, modifies the packet and then ships the packet to an appropriate router or internetwork router within the super-network. If the networkID in the arriving packet is for a super-network not directly connected to the merger router, then the packet is sent to the appropriate merger router. This may be a little difficult, since the packet networkID may have to modified in order to get the packet to the other merger router, thereby losing the networkID present in the packet when it arrived. This is the age old problem associated with performing tasks hop-by-hop, without using a separate hop-by-hop encapsulation. Assuming that there does exist a merger router to merger router protocol, all works well.

   This solution is rather complex and requires special software. The solution does not generalize very well if two super-network internets were to merge, and there were super-network networkIDs common to both super-network internets! One would have to have a super-network merger router and associated protocol, and so on!

3. There is an automatic procedure by which the network numbers are renumbered. This is simply an automated version of 1. above, and all the other changes must also be performed.

Internetwork Routing:

We now return to the problems of routing within the OIS communication system itself.

First some open issues: How does the routing machinery determine that the networkID hint in a destination network address is inaccurate? When the networkID hint in the network address fails what kind of recovery action should be made? Should the routing machinery inform the source which interrogates the clearinghouse for an alternative networkID, or should the routing machinery do so itself? If the latter, how does it let the source know about this change? If this happens once a sequenced packet protocol connection has been active, will this change upset the protocol machines at both ends?

In order to perform routing in large internets, internetwork routers must maintain routing tables indicating, for a remote network, the best internetwork router on a directly connected network, and the expected delay to get to the destination network from it. This information is propagated between internetwork routers periodically so that they can adaptively modify their routing tables based on possible changes in traffic patterns. This algorithm is similar to that used by the Imps in the Arpanet. Every host has a router, which has a routing table indicating the best internetwork router on a directly connected network for a given remote destination network. This table gets periodically updated since routers on a Xerox Wire can overhear the exchange of routing tables between internetwork routers over the Xerox Wire--a use of the broadcast feature of such networks. [Recall that we are still operating under the assumption that all system elements connected to networks other than Xerox Wires are internetwork routers]. When internet topologies get large, internetwork routers may not be able to hold the entire routing table, and similarly routers in hosts may also not know where to direct a packet destined for a remote network. Hence, there must be a recursive strategy by which a simple router in a host can ask one of its internetwork routers to find the appropriate information. This request percolates through the internet using some kind of broadcast routing algorithm with information being cached in internetwork routers once it has been found. This strategy seems most suitable even in large network topologies over hierarchical routing (examine Kamoun's thesis, or look at the Bell Telephone System).

Recall that internetwork routers on the same network, i.e. adjacent through a communication media, must exchange routing information periodically if the routing machinery is to be adaptive to topology and traffic changes. This becomes extremely difficult when networks other then Xerox Wires or Ethernets are added to the internet. When leased lines are added, there is no additional problem since the internetwork routers at the two ends exchange this information.

When packet switching, or circuit switching networks like Telenet, Arpanet, Autodin II, and the Bell Telephone System are added to the system, then internetwork routers (i.e. every system element connected to such a network) must exchange this information. The situation is not so simple any more since each one of these internetwork routers must know who the others are. *Mechanisms for providing this information to such internetwork routers must be determined.* There was no similar requirement with Xerox Wires since internetwork routers could resort to broadcast.

Internetwork routing has another level of complication associated with encapsulating the OIS packet for transport through the communication media. Recall that every network has its own local addressing scheme. That of the Xerox Wire is identical to hostIDs, but this will, in general, not be true for networks like Ethernets, Arpanet, Telenet, Bell Telephone System etc. Assume that an internetwork router has determined, from its routing tables, the internetwork router to which the packet should be forwarded. The internetwork router now has the networkID and hostID for that internetwork router. In an attempt to reach that internetwork router, its address on the network must be determined before it can be carried on that network towards its destination. *Mechanisms for providing this translation table to internetwork routers must be determined.* A rather troublesome situation arises when the internetwork router is connected to a public data network that supports X.25 via the Bell Telephone System. The internetwork router must now supply both the telephone number of the source DCE and the address of the destination DCE--a la source routing!

System Elements on other Communication Media:

In this section we consider the problems introduced by permitting system elements that are not internetwork routers to be connected to any communication medium. It is anticipated that for the most part there will be only a small number of such system elements in comparison with the number that are connected to Xerox Wires. The situations where a Xerox customer will want this to happen are those where he already has his own communication network, the number of OIS system elements is fairly small and they do not communicate very frequently.

The problems that must be solved are:

1. How does such a system element discover its networkIDs?
2. How does the routing table in the router of each system element get updated?
3. How does the hostID-to-local-address translation table get created in each router?

This problem is taken to its extreme when system elements are connected to the Bell Telephone System through simple dial-up telephone connections. The reason for the additional complexity is that since the Bell System is available everywhere, the number of system elements potentially connected to it is large (Victor Schwartz's estimates).

If we assume that all such system elements are internetwork routers, then we have solved the three problems since we have postulated that internetwork routers are told their networkIDs at installation time, they exchange routing information, and have the translation table provided at installation time (or something equivalent)! This is not an ideal solution because we have now increased the number of internetwork routers that do not do very much, thereby burdening all of them. We have also increased the amount of work that an installation manager must perform--the bookkeeping involved may become a burden. Such *fake* internetwork routers could exchange information with one another at a lower frequency such as when they communicate with another in order to transfer data. A number of short cuts are made possible by introducing a level of indirection. First we postulate that there is at least one internetwork router on every network. Every system element is given the hostID and local network address of these internetwork routers at installation time. The system elements can now communicate with these internetwork routers using the networkID zero. This connection can be used to determine the networkID of the network, and the routing and translation tables. The frequency of determining new routing information is based on the frequency with which the system element communicates with other system elements.

The OIS communication system and (to a certain degree) the distributed system that uses it, rely on the fact that a broadcast network is available. The Xerox Wire makes performing broadcast searches very easy--though sometimes expensive. Installation of system elements in such an environment becomes simpler, since some key installation parameters need be given to only a few system elements and the rest discover them automatically. Such a scheme makes every system element software identical and the installation procedure simple because installation parameters are bound at run time. Such schemes may make the changing of parameters a little difficult without executing the installation procedure or something similar.

Most other forms of communication media are not broadcast in nature and have their own local addressing scheme which differs from hostIDs, and therefore installation of system elements that are connected to such networks requires a little extra handling. Such networks make the design of clearinghouses a little difficult as well.

Internetwork Measurement and Performance Monitoring:

It is anticipated that as the OIS communication system grows it will be necessary to have continuous performance monitoring of the system, and remote control of internetwork routers. A scheme similar to that in use at PARC for measurement data retrieval, or the Arpanet Network Control Center and Network Measurement Center will have to be designed.

The function of this system is to gather data on traffic measurement and flow, in order to modify topology or capacity of links and/or internetwork routers. In the event that some internetwork routers malfunction, it must be possible to debug and restart them remotely.

A side effect of this system is that network topology and location of system elements can be determined from any point, thus enabling more sophisticated routing algorithms, and providing information as to physical location of resources both to application software and to maintenance personnel.

Clearinghouses:

This memo has defined the clearinghouse to be a general lookup data base. In reality there may be two such data bases--one for the private use of the OIS communication system, and the other for the general requirements of dynamic binding in distributed systems. This discussion has, in addition, assumed that there is one logical clearinghouse per internet. This need not be true, as there may be structure to a clearinghouse's data base, in order to limit the scope of certain lookups. The clearinghouse may be implemented using a distributed system or a centralized system.

In any case, there is still the problem of informing system elements of the network address of their clearinghouse (there may be more than one clearinghouse in the system). [For the moment let us not worry about clearinghouse use by the OIS communication system's routers.] We assume that, during system installation, internetwork routers have already been installed and that the routers know how to route packets to any network address. The different ways in which a system element gets bound to a clearinghouse is:

1. Each system element is told the network address of its clearinghouse at installation time.

2. The clearinghouse is told the network addresses of the system elements that will interrogate it, and the clearinghouse tries to contact these system elements and introduces itself to them, using an appropriate protocol.

3. The system elements are given the identity (some unique description) of their clearinghouse, and they try to find out the network address of the clearinghouse using broadcast (or exhaustive) searches.

Consider the suitability of these three schemes:

1. The first scheme requires determining the network address of the clearinghouse, and then having the system element remember it across machine booting. This binding to a clearinghouse network address may of course change during system operation. The scheme requires considerable effort on the part of the installation manager. The scheme has the advantage that it is applicable to system elements connected to any communication media, and that new system elements can be added incrementally once it has been decided with which clearinghouse they are associated.

2. The second scheme requires determining in advance all the system elements associated with a clearinghouse before the system elements are installed. Installation of new system elements at some later time requires updating this list in the clearinghouses. This scheme is also suitable for systems in which system elements are connected to any communication media.

3. The third scheme has the advantage that network addresses of system elements need not be known by anyone but the system element during installation, thereby simplifying the job of the installation manager. Broadcast searches are, however, difficult to perform. This scheme has the weakness that it is difficult for a system element connected to a non-broadcast communication medium to determine the network address associated with its clearinghouse. The OIS communication system (consisting only of Xerox Wires) can be designed to provide *global* or *directed* broadcast. The reliability of this broadcast cannot be guaranteed.

Conclusions:

This memo provides a uniform model and framework within which to view binding, addressing and routing within arbitrarily interconnected communication media. There are a number of problems associated with binding keys to values. This memo attempts to show how a large distributed system would be installed--with respect to initializing the OIS communication system, i.e. assigning networkIDs, propagation of routing information etc., and with respect to informing system elements of the network address of the clearinghouse with which they are associated.

Distribution:
SDD: *Bob Metcalfe, Bill Lynch, Hugh Lauer, Steven Abraham, Ron Crane, Pitts Jarvis, Communication Software, Archive*
PARC/SSL: *John Shoch*
PARC/CSL: *Dave Boggs, Will Crowther, Ed Taft*
ASD: *Steve Butterfield*