M.I.T. Laboratory for Computer Science Local Network Note No. 28
March 3, 1981

ON NAMES IN NETWORKS--ONE MORE TRY

by J. H. Saltzer

Despite the best efforts of John Shoch [1] to put confusion over names, addresses, and routes to rest, discussion continues to be confounded and confused by different perceptions of what names and addresses in networks mean. This confusion seems to stem at least partly from making too tight an association between various types of network objects and the most common form for their names. This note makes one more pass at trying to straighten things out.

Operating systems have a similar potential for confusion concerning names and addresses, what with file names and unique identifiers, virtual and real memory addresses, page numbers, block numbers, disk addresses, and such, but most of that potential has long been rendered harmless by explicit recognition that the concept of binding provides a systematic way to think about naming. In an operating system, one does not argue about the distinction between a name and an address. One looks at both as an example of a single phenomenon, and asks instead where is the context in which a binding for this name (or address) will be found, and to what object, identified by what kind of name, is it therein bound? This same approach is equally workable in networks, as shall be seen.

To start with, let's review Shoch's terminology:

- a name identifies what you want,
- an address identifies where it is, and
- a route identifies a way to get there.

There will be no need to directly tamper with those definitions, but it turns out that they leave a lot of room for (mis-) interpretation.

In a network, there are several types of things around for which there is more than one instance, and therefore it seems helpful to attach names to distinquish one instance from another. Of these several types, there are four that turn up quite often:

1) Services and Users. These are functions that one might want to use, and the clients that might want to use them. Example of services are one that tells the time of day, one that performs accounting, or one that forwards packets.
2) Nodes. These are computers that can run services, or run user programs. Two examples are a PDP-11/40 and an Alto. Some nodes are clients of the network, while others help implement the network by running forwarding services.
3) Network attachment points. These are the electrical connectors of a network, the places where a node can be attached.
4) Paths. These run between network attachment points, traversing forwarding nodes and communication links.

As mentioned, there are other concepts, but these four are the ones usually being discussed when confusion over naming strikes. We can make two observations concerning these four types of objects. First, since for each type there are usually many instances, we want to name the instances, and are free to choose any form of name that seems helpful--unique identifiers, human-readable character strings, or hierarchical-looking names. Further, there may be more than one form of name for a single type of object--a node might, for example, have both a character string name and a unique binary identifier. One of the worst sources of confusion comes from too closely identifying

some conventional form of name for a particular type of object as a property of that type. For example, service names might be assumed to be named by character strings, nodes assumed to be named by unique ID's, and network attachment points assumed to be named by hierarchical addresses. When one participant in a discussion assumes identification of name form with object type and another doesn't, the resulting conversation can be very puzzling to all participants.

The second observation is that most of the potential naming requirements in a network can be simply and concisely described in terms of bindings and binding dynamics among the four types of objects. To wit:

1) A given service may be run at one or more nodes, and may be moved from one node to another without losing its identity as a service.
2) A given node may be connected to one or more network attachment points, and may be moved from one attachment point to another without losing its identity as a node.
3) A given pair of attachment points may be connected by one or more paths, and those paths may change with time without affecting the identity of the attachment points.

Whether or not all of the flexibility implied by these possibilities should be provided for in a particular network design is a matter of engineering judgement. A judgement that a particular binding can be made at network design time and will never be changed (e.g., a particular service might always run at a particular node,) should not be allowed to confuse the question of what names and bindings are in principle present. In principle, to get a message to a service one must make three bindings:

1) choose a node on which the required service is operational,
2) choose a network attachment point to which that node is connected,
3) choose a path from this attachment point to that attachment point.

There are, in turn, three conceptually distinct binding services that the network needs to provide:

1) Service name resolution, to identify the nodes that run the service
2) node name location, to identify attachment points that reach the nodes found in 1)
3) route service, to identify the paths that lead from the requestor's attachment point to the ones found in 2)

and at each level of binding, since there are generally several alternatives, a choice of which node, which attachment point, and which path must be made. These choices are distinct but interacting. For example, one might choose the node after first looking over the various paths leading to the possible choices.

Confusion

We can find several examples in the network world of places where the naming mechanisms provided are an incomplete set, and some bindings are thereby permanent rather than flexible. The Xerox Ethernet protocols, for example, assign a 48-bit unique ID as the name of each node, but also make that same 48-bit unique ID part of the name of the network attachment point of the node. This required binding, together with the convention that the rest of the name of the network attachment point is the number of the network, makes it impossible for one node to connect to two attachment points on the same Ethernet.

For another example, the ARPANET NCP style of protocol provides character string names that appear, from their mnemonics, to be node names or service names, but in fact they are the names of network attachment points. Thus the character string name MIT-Multics is bound to the network attachment point at IMP 6, port 0, so reattaching the node (the MIT 68/80 computer) to another

network attachment point requires changing tables in every other node. Also, a parallel attachment of the MIT 68/80 to a second ARPANET port would be achievable only by assigning a second character string identity; this requirement emphasizes that the name is really bound to the attachment point, not the node. Again, because of their mnemonic value, the ARPANET NCP host name mnemonics are often thought of as service names. Thus one expects that the MIT Multics service is operated on the node reached by the name MIT-Multics. That assumption doesn't produce any surprises. But any of the MIT ITS machines can accept mail for any of the others, as can the groups of TENEX's at BBN and ISI. If the one to which one tries to send mail to is down, the customer must realize that the same service is available by asking for a different node. The need to give a different name to get the same service when a node goes down comes about because in the ARPANET the name is not bound to the service but to the attachment point.

Finally, confusion can arise because the three conceptually distinct services (service name resolution, node name location, and route dispensing) have not been thought of as mechanically distinct. There is usually suggested only one identifiable service, a "name server". The name server starts with a service name and returns a list of network attachment points that can provide that service. It thereby performs both the first and second conceptual services, and may leave to the customer the final choice of which attachment point to call. Path choice may be accomplished by a distributed routing algorithm that does not resemble a binding service at all.

Correspondence with names, addresses, and routes

With this model of binding among services, nodes, network attachment points, and paths firmly in mind, we can now interpret Shoch's names, addresses, and routes as follows:

1) Any of the four kinds of objects (service, node, network attachment point, path) may have a name.

2) The address of an object is the name of the thing it is bound to. Thus, an address of a service is the name of some node that runs it. An address of a node is the name of some network attachment point to which it connects. An address of a network attachment point (a concept not usually discussed) can be taken to be the name of a path that leads to it.

3) A route is a more sophisticated concept. A route to either a network attachment point or a node is just a path. Because a single node can run several services at once, a route to a service has to be a path to the network attachment point of a node that runs the service, plus some identification of which activity within that node runs the service (e.g., a socket identifier in PUP or IP.) But note that a route actually consists of a series of names, typically a list of forwarding nodes and the paths between them.

Reference

1. Shoch, J.F., "A Note on Inter-Network Naming, Addressing, and Routing," ARPA Internet Experiment Note IEN-19, SRI International, Menlo Park, Calif., January, 1978.