HEWLETT PACKARD
Information Networks Division
19240 Homestead Rd.
Cupertino, CA

From: Craig Wassenberg
Bob Carlson

Date: 2 February 1984

Subject: Path Report Revision

To:	John Balza (CNO)	Rick Bartlett (IND)
	Steve Booth (CNO)	John Bugarin (CNO)
	George Carter (IND)	Joe Devlin (IND)
	Carl Dierschow (CNO)	Joel Dunning (POD)
	Frank Fiduccia (IND)	Atul Garg (IND)
	Anne Hathaway (IND)	Clark Johnson (IND)
	Alex Lau (IND)	Brian Lynn (IND)
	Lissa Martin Sesek (IND)	Jack Repenning (IND)
	Mike Robinson (CNO)	Lynda Korsan (IND)
	Dave Tribby (IND)	Mike Wenzel (CNO)
	Jim Willits (CNO)	Ed Yang (IND)
	Gregg Levin (IND)	Chris Fuggitt (CNO)
	Carl Morgenstern (IND)	Deanna Osborne (CNO)

Attached is a new version of the path report specification. The changes from the previous version include (1) a reorganization of the group pid membership lists, (2) changes in the values of some of the group pids, and (3) longer discussions of the syntax and semantics of nodal path reports. Please direct any comments or corrections you might have to either of us at IND.

Table of Contents

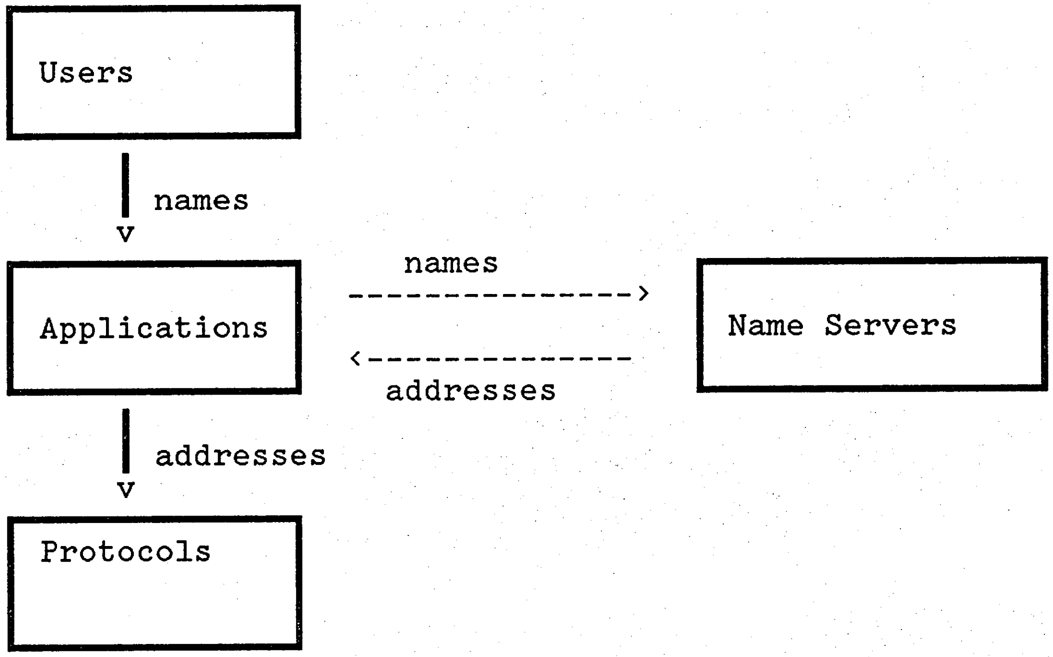
Path Reports

A Proposed Solution to the Communicable Address Problem

Bob Carlson
Craig Wassenberg
14 June 1983
Revised: <840208.1102>

Introduction — CHAPTER 1

A conflict arises in computer network design between names and addresses. When referencing objects like files, nodes, and IPC sockets, humans prefer to use names while communication/transport protocols prefer to use addresses -- names being more mnemonic and addresses more efficient. OneNet's transport protocols can only reference remote objects by address. Despite this limitation, OneNet applications will permit their users to reference remote objects by name. Under OneNet, the names that users use must be mapped into the addresses that the transport protocols require before remote objects can be successfully accessed.



The PROBE protocol, the MONAD network directory, and the IPC socket registry will all support remotely accessible name serving functions at OneNet first release. PROBE and MONAD will map node names into information about nodes, while the IPC socket registry will map socket names into information about sockets. Sometimes name servers will be clients of other name servers, for example, the IPC socket registry will sometimes invoke either PROBE or MONAD to resolve node names.

Name servers perform two basic functions: one of binding object names to protocol/addressing information, another of retrieving such information when subsequently queried with object names. Before it can send a message to a remote object, a node must first identify the protocols through which that object can be reached, then it must determine which addresses are needed in conjunction with those protocols. Having a name server available with the needed protocol/addressing information relieves nodes from having to supply it themselves. Furthermore, having a single name server which can be accessed remotely relieves users and nodal managers from having to place all of the name to protocol/address bindings they need into their own local data bases -- a remotely accessible data base can be shared across many nodes and many users. A problem, however, with such remote name servers is that the protocol/addressing information that they bind to names must somehow be communicated to them, and they in turn must somehow communicate the information when queried for it.

1.1 Where Path Reports Fit In

A path report is a data structure used to carry address information pertaining to one or more network objects. Types of network objects that may be described fall into two broad categories: nodes and connect-sites. Some name servers, notably the IPC socket registry, will traffic exclusively in connect-site reports. Other name servers, notably PROBE, will traffic in nodal reports.

Connect-site path reports carry very specific address information. A "connect-site" is defined here to mean a logical location within a machine to which messages intended for a particular network object can be addressed. The majority of active connect-sites in a OneNet network will have IPC sockets bound to them: it will be through these sockets that applications and services will receive their messages.

From a connect-site report, a source node can discover which protocols and addresses should be used to send a message to a particular remote connect-site. Such a report could be used to describe how to address messages to, say, socket BARNEY at node FRED. Typically, such a report would reveal FRED's internet address(es) as well as the protocols and corresponding addresses needed to address socket BARNEY.

A nodal report provides general information about a node. It describes the node's VNAs, the well-known services it supports, the protocols it supports, and the machine addresses that it can be addressed by. Nodal path reports do NOT describe the particulars of the protocols and addresses that must be used to send a message to a particular object within a node, for example, a nodal report may reveal that a node supports, say, NFT, but won't reveal the well-known TCP port to which NFT's socket is bound.

For the most part, nodal reports are only useful to nodes having considerable a priori knowledge of the HPDSN Canonical Addressing Standard [2]. Only with such knowledge, can messages be addressed to well-known services.

When talking about path reports, it is useful to make a distinction between two types of routes: those between machines and those within machines. In this paper the term "path" is used to refer to routes within machines, and the term "route" to refer to routes between machines. Paralleling the path-route distinction is the distinction between "machine" and "dispatch" addresses. In this paper, a "machine address" is one which references a particular machine, and a "dispatch" address is one which references a particular protocol. A protocol like ARPA's IP carries both machine and dispatch addresses in its header. TCP carries only dispatch addresses. A route can be expressed as a sequence of machine addresses, a path as a sequence of dispatch addresses.

Path reports carry both machine and dispatch addresses. The machine addresses uniquely identify a machine, while the dispatch addresses, presented as parts of sequences, describe the paths by which an inbound message may reach one or more of the objects on that machine. Path reports really perform two functions: they describe which nodes objects reside upon, and they describe where within those nodes the objects may be accessed. Any nodal or connect-site report might describe several paths because any node might support several protocols at any given layer.

Described below is a typical event sequence involving a connect-site path report.

(1) A user on node WISCONSIN creates a socket and names it BuckyBadger.

(2) Another user, not on WISCONSIN, but wishing to communicate with BuckyBadger calls IpcLookUp(), a routine which interacts with the IPC socket registry.

(3) The IPC socket registry prepares a name query message, puts the name BuckyBadger in it, and then sends it off to some socket name server -- a server which probably, but doesn't necessarily, resides on WISCONSIN.

(4) The name server receives the query and is able to resolve the name BuckyBadger. The name server prepares a connect-site path report which describes all possible paths to the socket BuckyBadger (let's assume the socket may be accessed either via TCP or the NBS transport). The server

puts the path report into a query reply and sends it back to the query initiator.

(5) When the query reply arrives, its path report will be extracted. Then when the user is ready to initiate a connection to BuckyBadger the necessary addressing information will be available.

The example above is interesting because the returned path report describes two alternate paths, one involving TCP and the other involving the NBS transport. A node wishing to initiate communication to the object that a path report describes must evaluate that report to determine which path is best. Sometimes the choice will be simple, e.g., when one of the described paths requires a protocol which the initiator doesn't support. At other times the choice will be more involved, requiring the application of relatively sophisticated heuristics (aside: source routing bears many similarities to "source pathing" in that both operations require a source node to choose some portion of a route to a destination object, in the case of source routing the source node chooses a route to a node's entry point, in the case of source pathing the source chooses a route from the node's entry point to the object's entry point).

1.2 Objectives

The organization of a data structure can -- and usually does -- have strong implications for the types of algorithms that can be used to manipulate it. Recognizing this, we defined a set of objectives that we felt a usable path report structure would have to meet: these are listed below. While considering the list, readers should understand that we hold the definition of packet formats as a "non-objective," i.e., we have no intention of defining what such packets look like.

A successful path report structure should ...

(1) satisfy the needs of the IPC name server, the PROBE protocol, and the MONAD network directory. We envision these services devoting portions of their packets for the conveyance of path reports.

(2) be both easy to generate and easy to decode. The programming techniques required should be realistic, efficient, and reasonably simple; recursion, for example, would probably not be realistic.

(3) complement rather than subvert the canonical addressing standard [2].

(4) be extensible. OneNet will grow, new protocols will be introduced, and improved methods for offering distributed services will be discovered. The path report format should accomodate changes not impede them.

(5) be flexible. It should permit experimentation with alternate protocol implementations. More specifically, it should permit development groups to experiment with layer skipping and the support of alternate protocol stacks.

(6) meet the needs of multi-homed nodes. Path reports should not hinder the support of multi-homed nodes.

1.3 Compromises, Trade-offs, and Assumptions

The adoption of any particular path report format would probably represent a compromise. Trade offs exist between generality and specificity, space and time, static and dynamic configurations, etc. Deciding between the various compromises and trade-offs would be impossible unless some assumptions were made. In this section we list the assumptions we made. If any of them are unreasonable then portions of the design are probably inadequate and should be changed. Readers are eagerly encouraged to submit comments regarding areas for improvement.

(1) The ARPA IP protocol won't be a participant in all paths -- some paths may skip the IP layer altogether. In addition, it is likely that IP could be replaced by some other internet protocol, especially in light of HP's customer's interest in newly developing international protocol standards.

(2) Path reports should carry information about protocols existing at layers below the internet (3i) layer, for if they did not then either nodal configuration would be more involved or network operation would be more inefficient. When MONAD replies to node name queries it will return a path report. This arrangement allows a user on, say, node FRED to request communication with a peer user on node CHARLIE even though node FRED has no information configured about CHARLIE. FRED would send a query to MONAD referring to CHARLIE. MONAD would resolve the name CHARLIE into a path report and return it to FRED. Were the path report not to contain sub-3i information then FRED would have to derive this information from some other source -- either from configuration tables or from additional queries, e.g., PROBE.

(3) It is safe to permit path reports to depend on unique internet address assignments. The major network architectures that we're aware of (ARPA, ISO, Xerox NS, etc.) rigorously follow

the convention that all network nodes have (one or more) unique internet addresses. HP plans to assist its users in choosing unique assignments. The problems raised by non-unique assignments are very troublesome and any solutions would probably be complex (possibly involving the use of source routing, a service not included under current OneNet plans).

(4) Path reports should be capable of handling multi-homed nodes. A query concerning a multi-homed node should result in a response which contains path reports relative to each of the node's internet identities.

Path Report Structure — CHAPTER 2

At first release, MONAD, PROBE, and the IPC socket registry will return path reports in response to queries. While MONAD and PROBE will probably traffic exclusively in nodal path reports, the IPC socket registry will traffic in connect-site path reports. This paper is not an attempt to define the protocols that these name servers will use, it is only a proposal for the format of the data to be returned in their query replies.

The path report structure has been organized around a few central concepts. Understanding of these concepts is prerequisite to understanding the path report structure. One central concept is that of the Domain. A large catenet may contain several domains. Although domains are roughly separable on the basis of the protocols that are spoken within them, their truly distinguishing feature is the way in which they permit unique identification of member nodes. Nodes within the HPDSN domain, for example, may be uniquely identified by their ARPA IP addresses. Similarly, nodes within the Xerox domain may be uniquely identified by their IDP addresses (IDP is the Xerox NS internet protocol). Usually it is quite easy to identify the nodes within a domain, because usually, although not always, some form of protocol address can be used.

Another important concept is that of the Virtual Network Address (VNA). A VNA is nothing more than a (relatively simple) way to uniquely identify a node within a catenet -- even in a multi-domain catenet. Since no single internet protocol pervades all domains the VNA can not be a true network or internetwork address, instead it must be a "virtual" network address. A node's virtual network address is formed by concatenating an identifier for its domain to its unique name/address within its domain.

Both nodal and connect-site reports contain one or more domain reports. Each domain report contains a VNA address and a set of one or more paths which may be used within the context of the domain referenced in the VNA.

2.1 Syntax: Backus-Naur Form (BNF)

Note: The curly brackets denote repetition of the enclosed symbols zero or more times.

```
<path report>  ::= <report length> {<domain report>}

<domain report>::= <domain report length> <version>
                   <virtual network address> <path list>

<virtual network address> ::= <domain>
                              [<ARPA IP addr>] | [<ISO addr>] |
                              [<Xerox IDP>]

<path list >   ::= {<path>}

<path>         ::= <path length> {<path element>}

<path element> ::= <pid> <element length>
                   [<address info>] [<pad byte>]

<pid>          ::= <individual pid> | <group pid>

<address info> ::= [<service map>] [<sap>] [<machine address>]
```

2.2 Semantics

Path Report: A list of zero or more domain reports.

Report Length: The sum length, in bytes, of all the domain reports in the path report. This length field, just like all the other length fields in path reports, is not self-referencing, i.e., does not include the number of bytes which it occupies. A non-zero length guarantees that the path report contains at least one domain report, while a zero length means the path report describes no paths. The meaning of an empty path report, returned in a query reply, is that the name specified in the query could not be found.

Domain Report: There are two major categories of domain reports: nodal reports and connect-site reports. At present it doesn't seem as if there will ever be a case where both kinds of domain reports will be returned in the same path report.

Domain Report Length: The length of the entire domain report expressed in bytes. Domain report lengths must be an even number of bytes in order to accomodate machines which don't support byte addressing well. This field is not self referencing.

Version: A value describing the format adherred to by domain report. All first-release domain reports should carry a version number of zero. Later releases will carry higher version numbers if the domain report format they use differs from the version zero format. Introducing a new format version into a functioning network could be disruptive. Were a node to attempt to decode and act upon a domain report having a version number for which the node wasn't configured could produce unexpected results. Requiring that all of a network's nodes be upgraded simultaneously would be unacceptable to many customers. In light of these two observations we recommend first, that no node attempt to decode a domain report carrying an unsupported version number, and second, that newly upgraded nodes support the most recent version as well as (at least) one previous version.

One final note: packets in which path reports are conveyed may have their own version numbers independent of path report version numbers.

Virtual Network Address: This field uniquely identifies the node which the path report references. Virtual network addresses are hierarchically arranged. The topmost level in the hierarchy is termed a "domain." Domains qualify the type of internetwork environment in which a node resides, e.g., ARPA, Xerox NS, etc. Most internetwork domains employ quite different sets of protocols, thus, some nodes which receive path reports may be capable of communicating with nodes in some domains but not others. Nodes can use the domain value to quickly weed out domain reports from incompatible domains.

The lower hierarchical level of a virtual network address will usually contain an internet address of the type carried by the domain's internet protocol. In the "ARPA domain" virtual network addresses would consist of an an ARPA domain identifier prepended to an ARPA IP address. This splitting out of internet addresses from the remainder of a domain report has at least one advantage which isn't immediately obvious -- it permits an evaluating node to determine which network a connect-site report applies to even when the report describes an INTRAnet path (i.e., a path which skips a domain's internet protocol entirely).

Some further points about virtual network addresses are the following: (1) They don't permanently bind us to ARPA's IP -- path reports refer to IP only when expedient. (2) Although independent of all internet domains they nevertheless enable us to take advantage of conventions employed in those domains. (3) They result in some space savings since ARPA IP path elements (explained later) need not carry network and node numbers, and Router path elements need not carry node numbers.

Domain: Defines the internet domain that the path report applies to. Currently defined domain values are shown in Table 2.1

below. The domain values are intended to correspond with those used by the HP/DSN IPC service [6]. Other domains may be defined as necessary. The UNIX domain applies to communication between programs running within the same UNIX machine and so might never appear in a path report -- the entry is included here for completeness only.

Domain Name	Domain Number	Associated Virtual Network Address Protocol
HPDSN	1	ARPA IP
UNIX	2	None
NBS	3	???
XEROX	4	Xerox IDP
ISO	5	???

Table 2.1: Domain names, numbers, and associated internet protocols.

ARPA addr: The thirty two bit ARPA internet address of the node that the path report references.

Path List: Path lists for connect-site path reports will consist of one or more "individual" path elements (see below). Path lists for nodal path reports will consist of one or more group path elements and one or more individual path elements.

Path: Contains one or more path elements that "belong together." The path elements of a connect-site path describe all the protocols and associated addresses needed to send a message to a particular connect site. The path elements of a nodal path report describe a node's location as well as the services and protocols which it supports.

Path Length: The length of a complete path expressed in bytes. This length does not include the length of the Path Length field itself.

Path Element: "Individual path elements" reference a protocol and an address to be used in conjunction with that protocol. "Group path elements" reference a particular protocol group and the services offered by that group. Although two different nodes might support the same general group they might not both support all the possible services of that group.

Pid: "Pid" is short for "protocol identifier." There are two kinds of protocol identifiers: individual pids and group pids. Individual pids appear in individual path elements, and group pids in group path elements. An individual pid identifies a particular protocol, e.g., TCP, UDP, IP, etc. Path reports sent from IPC name servers will always be connect-site reports and will only reference individual pids, never group pids. Table 2.2, below, shows the values to be used for individual pids; these values correspond to the ones defined for use by HP/DSN IPC users [6].

Protocol Name	PID	Service Map?	Service Name	Bit Index
Ethernet	1	no	none	*
X.25	2	no	none	*
MAPLE	3	no	none	*
TCP	4	yes	TCP CHKSUM	0
			none	1-15
UDP	5	no	none	*
HPPXP	6	no	none	*
IEEE-802	7	no	none	*
IP	8	no	none	*
telephone	9	no	none	*
NBS Transp.	10	no	none	*
HP RPM	20	no	none	*
HP NFT	21	no	none	*
HP VT	22	no	none	*

Table 2.2: Individual pid assignments.

Group pids identify sets of protocols or services, e.g., the set of application protocols running above TCP, PXP, and UDP on first release OneNet machines (see Figure 2.1 below). A group pid will generally have a "service map" associated with it. The bits of the service map indicate which of a group's services/protocols a subject node supports. Group pids would be appropriate for use in reports from MONAD's and PROBE's node name servers.

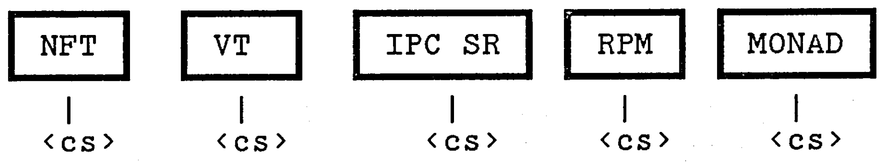


Figure 2.1: Protocols which are members of the OneNet_Services group pid. The group element containing the pid does not describe the addesses and protocols necessary to send a message to any of the service's connect-sites.

Table 2.3, below, shows all of the currently defined group pids along with the bits defined for their service maps. Other group pids can be defined as needed, e.g., if the desktop computer family decides to offer its own set of services.

Group Name	PID	Service Name	Bit Index
OneNet_Services	255	NFT	0
		VT	1
		IPC SR	2
		RPM	3
		MONAD	4
		RFA/3000	5
		RDBA/3000	6
OneNet_Transport	254	TCP CHKSUM	0
		TCP	1
		HPPXP	2
		MAPLE	3
		UDP	4

Table 2.3: Group pid and associated service map bit assignments.

It is only appropriate to include group pids in a nodal report. A nodal report is not intended to describe the particulars of specific connect-site paths, but rather to describe a node's machine address(es) and the protocols/services which it supports. The path elements of any path contained in a nodal report must be arranged in a top-to-bottom fashion, i.e., the services groups must be placed at the top, the transport groups lower down, and individual pids -- for the protocols that the transports run over -- at the bottom. A single path may contain multiple service group pids but only one transport group. It will normally be true, however, that the paths carried in a nodal report will be simple, containing a single services element, a single transport element, and one or more individual protocol elements.

Service and transport elements appearing within the same path are related in that all of the marked services are guaranteed to be accessible via all of the marked transport protocols. This means that more than one path must be included in a nodal report if one service, say RFA, can run across either TCP or UDP, but another service, say NFT, can only run across TCP.

Individual pids are included in nodal path reports to carry machine addresses, e.g., IEEE-802 station numbers, and X.25 addresses. When used in conjunction with a working knowledge of the HP canonical addressing standard, these machine addresses make it possible for one node to initiate communication to the services of the target node. The sub-internet addresses which may be included in a path report won't be of use to initiators that aren't connected to the same local network as the target. Nevertheless, the sub-internet addresses may save considerable work for initiators that are on the same network as the target (e.g., they can permit the initiator to forego invocation of the PROBE protocol).

Element length: The length, in bytes, of the path element's address info field. This value may be odd, in which case the address info field will be followed by a pad byte. Note that including the one byte length of the pad byte as part of the element length would make it impossible to unambigously communicate variable length addresses.

Address info: A protocol-dependent (pid-dependent) field which may contain any or all of the following: a service map, a machine address, and a sap. Path elements of group pids generally contain only service map fields and not sap, location address, and pad byte fields. The service maps of group pids describe the protocols and/or services supported by those groups.

Path elements of individual pids almost always contain either sap fields or location fields or both. This information will usually end up in a protocol header if the path report is accepted as the best connect-site path to a service access point.

Service Map: A bit map roughly equivalent to a capability mask. Group path elements usually include service maps describing protocols and/or services supported within the group. The OneNet_Services bit map has bits indicating which of the standard OneNet application services the machine supports. The OneNet_Transport bit map has bits indicating which of the standard, first-release protocols the subject machine supports; in addition, the map indicates whether the node's TCP supports optional check-summing.

Some individual path elements will contain service maps. TCP, currently has one, and it has one bit defined, bit zero, which indicates whether check-summing is optional. All bits which are undefined should be set to zero.

Sap: Most subject protocols of path reports carry some dispatching information: TCP has a destination port field, IP a protocol number field, Xerox IDP a socket field, etc. The sap fields of path elements are set up to carry this dispatching information in the same format as the protocol referenced by the path element would carry it (see Table 2.4 below). Because sap fields are sixteen bits long it will be necessary to right justify this information for protocols which carry dispatch fields narrower than sixteen bits. If some future protocol carries dispatch field wider than sixteen bits then either the path element of the protocol will be given a new format, or, more likely, a new path report format and version will be defined.

Protocol	Field Name	Field Length in Bits
TCP	Destination Port	16
UDP	Destination port	16
ARPA IP	Protocol	8
Xerox IDP	Destination Socket	16
Router	Dest Protocol ID	16
X.25	Call User Data	16 (proposed)
IEEE-802	DSAP	8

Table 2.4: Correspondences between protocols and sap fields.

Machine Address: The sap field satisfies a protocol's intranodal dispatching requirements. It does not, however, satisfy the needs that some protocols have for specifying nodal locations. The Machine Address field, therefore, is included in the path elements of those protocols which require it. As a general rule, most protocols - but not all _ that carry node numbers, network numbers, station numbers, etc., will have Machine Address fields included in their path elements. Exceptions to this rule are made in cases where the necessary location information for a protocol can be derived from the Virtual Network Address field -- protocols in this later category include ARPA IP, and Router.

Protocol	Machine Address Field Length	Name of Corresponding Field(s) in Protocol Header
IP	0	Comment: Carried in Virtual Network Address field
X.25	8 bytes	DTE address - 14 BCD digits
Router	0	Comment: Carried in Virtual Network Address field
IEEE-802	6 bytes	Station address

Table 2.5: Descriptions of Machine Address fields for selected protocols.

2.3 Syntax Revisited: Message Form

This section illustrates path reports as they would be blocked out as part of a message. The bit numbering convention used matches the one used on HP3000s: the MSB and LSB are numbered zero and fifteen respectively (the reverse of the order used on HP1000s).

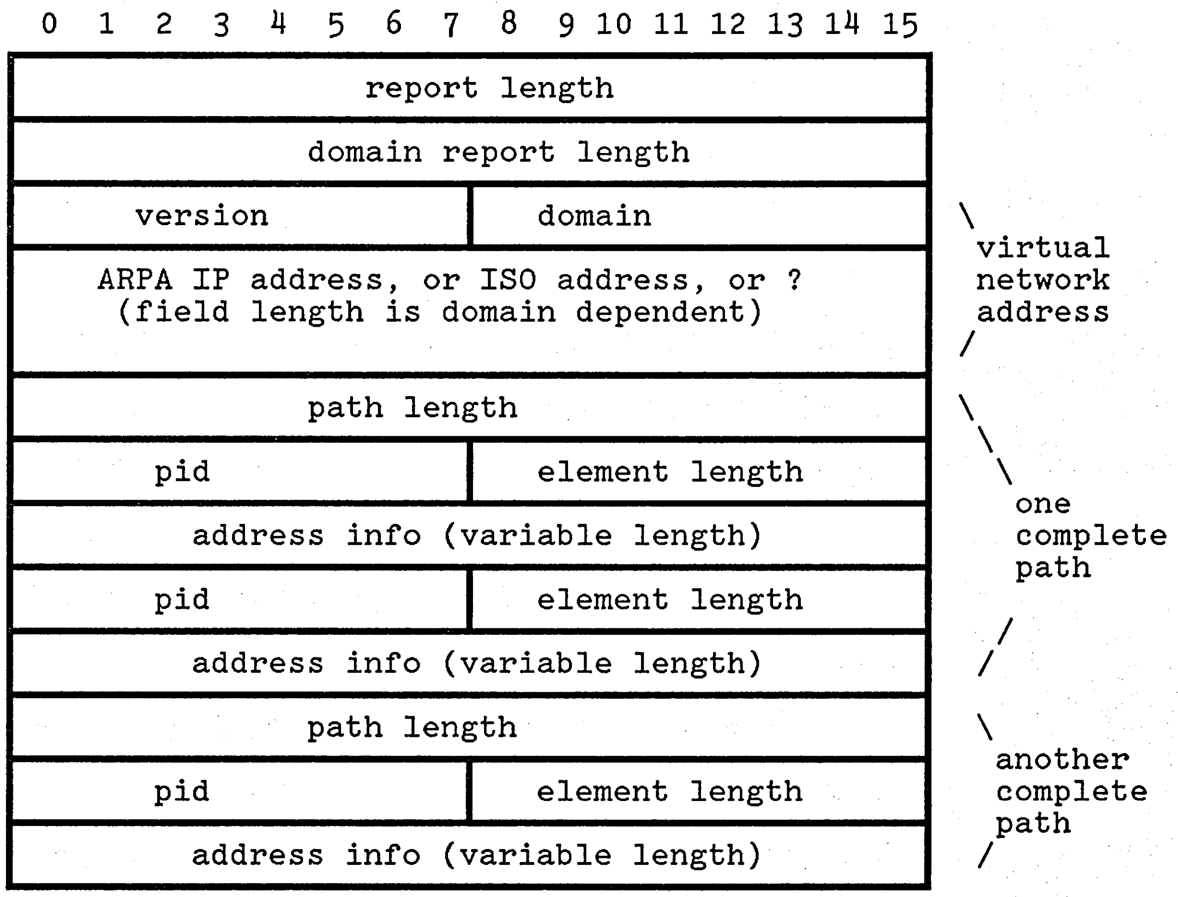


Figure 2.2: Path report message form.

2.4 Practical Applications of Path Reports

OneNet services probably won't use the full flexibility of path reports very often. The majority of path reports will probably only contain one or at most a very few alternate paths. Most OneNet nodes won't support layer skipping or multiple protocol stacks during the near term; this implies that these nodes won't have any alternate paths to report about. The most frequently reported paths will probably be the following: TCP-IP-802, TCP-IP-ROUTER, TCP-IP-X.25, UDP-IP-802, and UDP-IP-X.25.

The path report format hasn't been fully optimized to reduce space. Reducing path report space would increase the time required to generate and evaluate path reports. To trade off space for time would seem counter-productive given that path reports as they are now defined require so little space.

2.5 Example Path Reports

In this section we present a few path report examples. The first example, shown in figure 2.3, is a nodal path report. The node in question supports only the protocols of the HPDSN domain. The nodal report reveals that the node supports NFT, IPC socket registry, RPM, and MONAD but not VT -- this information is contained in the topmost (group) path element. The second from the top path element reveals that the node supports almost the entire set of first release OneNet protocols (i.e., all but MAPLE). The last path element reveals that the node is a member of an IEEE-802 LAN. Note that the sap field of the IEEE_802 path element is set to "meaningless" in this example since this dispatch value will take on different values depending upon the which upper level protocol an inbound message is sent to.

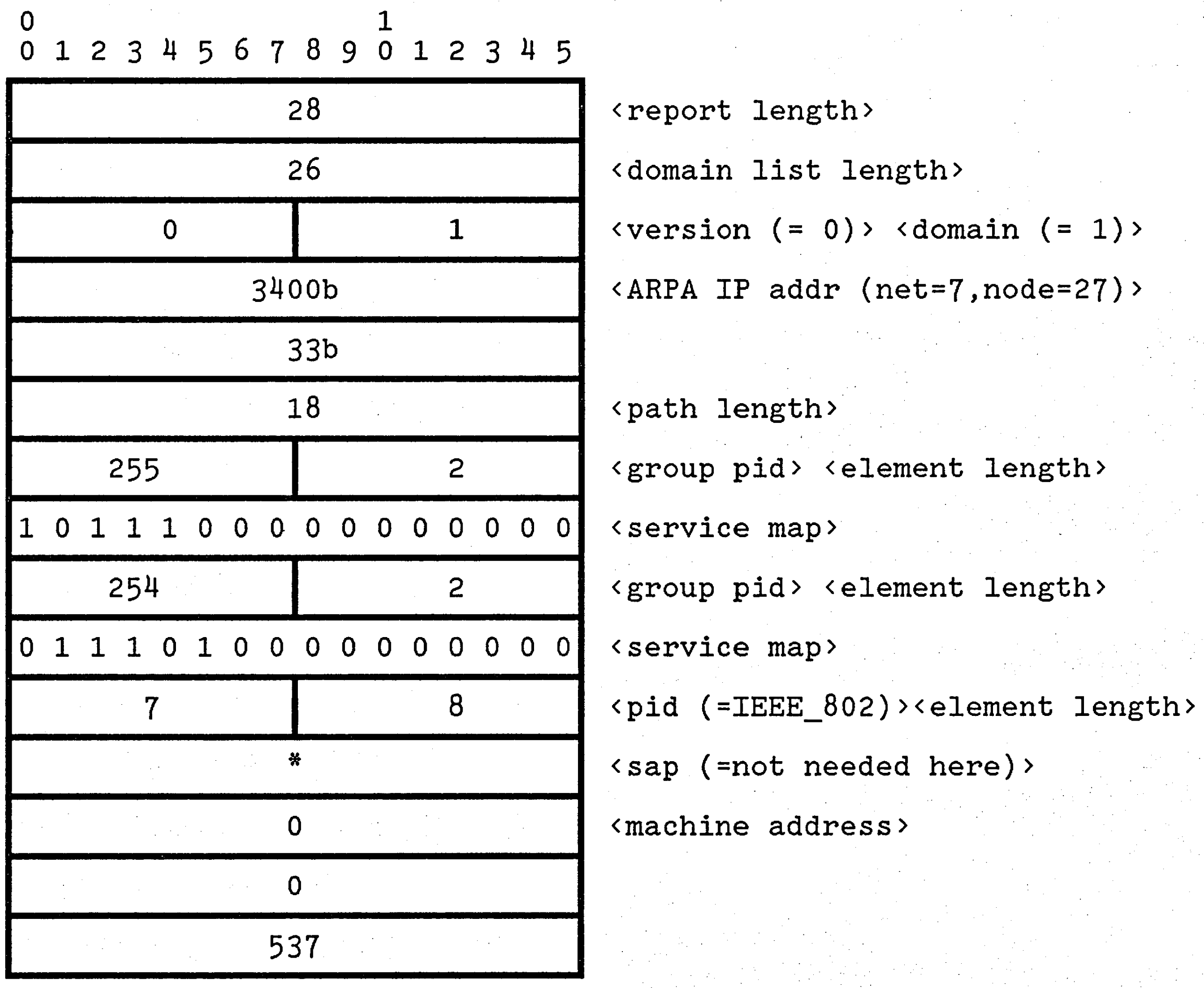

0 1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
28 <report length>
26 <domain list length>
0 1 <version (= 0)> <domain (= 1)>
3400b <ARPA IP addr (net=7,node=27)>
33b
18 <path length>
255 2 <group pid> <element length>
1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 <service map>
254 2 <group pid> <element length>
0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 <service map>
7 8 <pid (=IEEE_802)><element length>
* <sap (=not needed here)>
0 <machine address>
0
537

Figure 2.3: Format of a path which describes a node running on an IEEE-802 LAN.

The connect-site path report shown in Figure 2.4 is typical of one that would be returned by the IPC Socket Registry in response to a query. The single path in this report contains sufficient information to describe both an intra- and an inter- net path -- the intranet path would include TCP, IP, and X.25, while the internet path would include only TCP and IP. If the path report's subject node supported layer skipping then the path report might also list a second path, one showing TCP running directly on top of X.25.

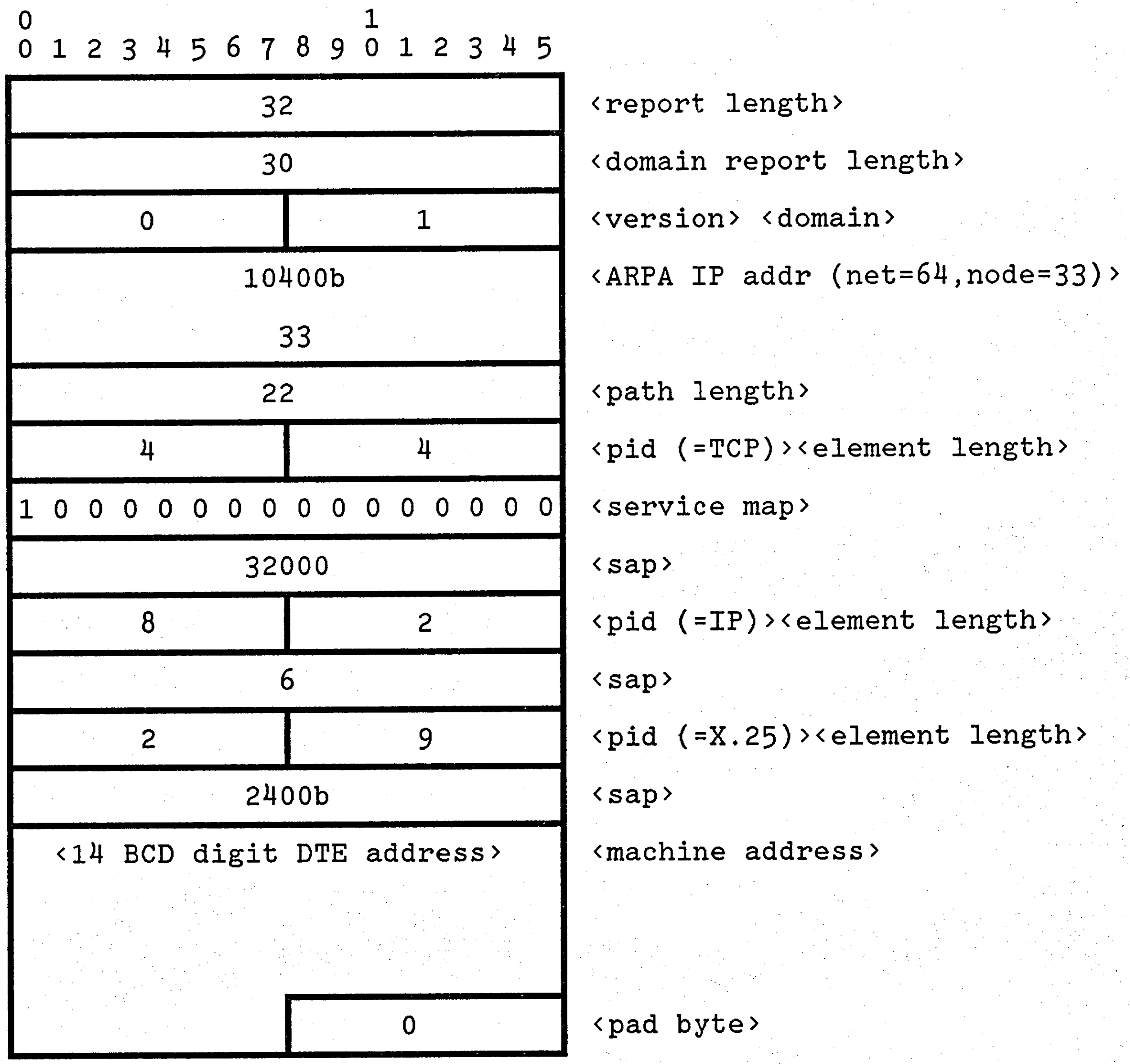


Figure 2.4: Format of a path to an IPC call socket.

Connect-Site Path Report Evaluation — CHAPTER 3

This chapter examines the problem of evaluating connect-site path reports. Such a report might be received, for example, in response to a query to the IPC socket registry. The report would describe all the ways by which a particular IPC socket could be accessed through the protocols of the internet. The objective of path evaluation is to find the best path that our source node can support.

3.1 The Relevant Questions

There are several criteria against which an offered path should be evaluated.

(1) Does the path reference any protocols which our source node doesn't support. Obviously, a node which speaks only TCP and IP on top of a LAN can't communicate with another node which supports only the NBS Transport on top of Xerox IDP. Any path requiring a protocol which our source node doesn't support must be rejected.

(2) Can our source node support all of the path's protocol adjacencies? Some machines may be capable of running, say, HPPXP directly over IEEE-802 while others may not. Although the two machines might both support the same protocols they might not be capable of supporting the same protocol adjacencies. A path specifying an unsupported protocol adjacency must be rejected.

(3) Does the report describe an intra- or an inter- network path? An intranetwork path, for our purposes, is one which can only be used between two nodes coexisting on the same network. Intranet paths can be optimized to take advantage of the proximity of the two communicating nodes, e.g., in the OneNet domain they could skip - for efficieny - the IP protocol layer entirely. This is not to say that intranet paths must skip the IP layer or that intranet path reports may not include IP path elements. Intranet path reports will contain path elements for protocols running below the internet (ISO 3i) level and may include path elements for internet protocols like IP. Note that an internet path (see below) may be extracted from an intranet path that contains an internet protocol path element.

An internet path is one which can be used to support communication between two nodes existing on different networks. Internet paths reports must include path elements for an internet protocol. An internet path may be derived from an intranet path report if that report contains an internet element, e.g., a path report with elements for TCP, IP, and IEEE-802 specifies an intranet path which includes all three protocols, but also contains an internet path which contains only TCP and IP.

(4) If a path report contains multiple paths how does one determine the best alternative? Disregarding multihoming for the moment, finding the best path is equivalent to finding the "first fit" path if the paths have been placed in the path report in their preferred order, i.e., best performing paths first. A few simple assumptions make preference ordering possible -- faster protocols are preferable to slower ones, and intranet paths are preferable to internet paths.

The path report list for a multihomed node will contain multiple path reports -- one for each of the node's virtual network addresses. The evaluator of such a list might discover that each of the path reports contains a viable path or paths. Preference ordering of the paths within a report means that discovering a single report's best path won't be a problem. Deciding between the paths of competing path reports is more troublesome however. Having the name server that delivers the path reports put the alternative path reports in some kind of preference order does not appear to be a viable solution to the problem unless quite a bit of intelligence about network topology is placed into the name server.

Consider a multihomed node, ABN, residing on both networks A and B. ABN's path report list will contain one path report which is A-based and another which is B-based. An A-based intranet path would be preferable to an A-resident node and a B-based path to a B-resident node -- even though both the A- and B- residents might be capable of supporting either of the A- or B- based internet paths. If ABN were to generate the path report list it could conceivably order the path reports relative to the node that requested it. If, on the other hand, a generalized network name server not residing on node ABN were to serve up the path list it would have to evaluate the relative positions of both ABN and the query initiator. This problem, sometimes referred to as the "three node problem" has motivated us to place the burden of choosing between the viable paths of competing path reports up to the query initiator and not up to the name server.

3.2 Example Evaluation Algorithm

In this section we present an example path evaluation algorithm. The algorithm uses a first fit acceptance criterion while attempting to extract a viable path from a path report. The example is supposed to show that the path report format is reasonable to work with. This algorithm is included here only to illustrate a few general concepts; different implementers will probably create and tailor similar algorithms to satisfy their own specific needs.

```
PROCEDURE ExtractPath ( VAR report         : ReportType;
                            intended_user  : Integer16;
                        VAR pathstart      : Integer16;
                        VAR pathlen        : Integer16;
                        VAR result         : Integer16    );

{}
{ Abstract:
{  ExtractPath tries to extract a viable path from a path
{  report. The length fields in the path report are used
{  to determine where the end of the path report is, where
{  paths begin and end, and where path elements begin and end.
{  This algorithm assumes (naively) that our machine has
{  reasonable byte addressing capabilities and that the path
{  report it is analyzing has its domain field set equal to
{  HPDSN.
{
{ Input parameters:
{
{  report: A fully formed path report -- presumably one obtained
{     from a remote name server.
{
{  intended_user: The protocol id (pid) of the user that intends
{     to use this path. Values for this field might include
{     <IPC user>, <NFT>, <Net Management>, etc. A value of
{     <IPC user> would indicate that the caller was trying to
{     find a path that an IPC user could use to access a remote
{     socket. A value of <NFT> would indicate that NFT wanted
{     to find a path to its peer NFT.
{
{ Output parameters:
{
{  path_start: An offset into the path report which indicates
{     the starting position of the located path. This value
{     is meaningless if the returned "result" parameter has
{     value NO_PATHS.
{
```

```
{  path_length: The length of the loca+ed path. Note that this
{     length could be less than the length specified in the
{     report's "path length" field -- this would happen if an
{     internet path was extracted from an internet path.
{
{  result: Returns SUCCESSFUL if a patn was found and NO_PATH if
{     no path was found.
{}

TYPE
   Int16 = -32768..32767;

   ReportType = RECORD
      CASE BOOLEAN OF
      TRUE : ( words  : PACKED ARRAY [1..N] OF Int16 );
      FALSE: ( bytes  : PACKED ARRAY [1..N] OF Byte  );
   END; {RecordType}

CONST
   CURRENT_VERSION  = 0;    {OneNet first release version number}

VAR
   end_of_report : Int16;
   i             : Int16;
   intranet_path : BOOLEAN;
   nets_match    : BOOLEAN;
   nextpath      : Int16;
   pid           : Int16
   satisfied     : BOOLEAN;
   user          : Int16;

BEGIN
   i := 2;
   end_of_report := report.words[i DIV 2] + 1;
   i := i + 1;

   <<< evaluate the version and virtual network address fields.
       If the network specified in the virtual network address
       field matches a network that this node is connected to
       then set nets_match to TRUE. Initialize i so it indexes
       path length field of the report's first listed path.
   >>>

   satisfied := FALSE;

   WHILE ((NOT satisfied) AND ( i <> end_of_report )) DO
   BEGIN
      i := i + 1;
      user := intended_user;
      nextpath := report.words[i DIV 2] + i + 1;
      i := i + 1; {points to pid field}
```

```
pathstart := i;

internet_path := FALSE;
rejected := FALSE;

{ Examine path elements in the currently indexed path.
{ Reject the path if we don't support one of the
{ protocols or adjacency relationships
{}
WHILE ((i <> nextpath ) AND (NOT rejected)) DO
BEGIN
  {}
  { Get the path element's pid and then make i index
  { the next path element.
  {}
  pid :=  report.bytes[i];
  i := i + 1;

  { Advance index to the first byte after this path element.
  {}
  i := i + report[i+1] + 1;

  IF (NOT Supported(pid)) THEN
  BEGIN
     rejected := TRUE;
     i := nextpath;
  END
  ELSE IF (NOT Adjacent(user,pid)) THEN
  BEGIN
     rejected := TRUE;
     i := nextpath;
  END
  ELSE IF ( pid = IP ) AND (NOT nets_match) THEN
  BEGIN
    {}
    { Our target node is on a remote network. We've
    { got an IP address. Therefore, we've got all the
    { information we need.
    {}
    pathlen := i - pathstart;
    i := nextpath; {So we can leave the while loop}
    internet_path := TRUE;
  END; {IF pid}

  user := pid;
END; { WHILE i <> nextpath }

satisfied := (NOT rejected) AND (i <> pathstart) AND
             (internet_path OR
                (NOT internet_path AND nets_match));
```

```
      IF (satisfied AND (NOT internet_path)) THEN
      BEGIN
        pathlen := i - pathstart;
      END; {IF satisfied}

  END; { WHILE }
END; { PROCEDURE ExtractPath }
```

Report Generation — CHAPTER 4

There are probably an infinite number of algorithms and data structures that could generate path reports. The best of these would be fast and require little space. Unfortunately, space and time reductions usually represent conflicting goals. We think that with any reasonable data structure the storage space needed to generate path reports will be small. Implementers should therefore try to reduce the time required to generate reports.

4.1 One Possible Algorithm

One simple algorithm which could be used to generate path reports relies on the use of "path templates." A path template contains a skeleton structure for a path -- it is actually an incompletely specified path.

A path differs from a path template in only one major respect. The address info fields of a path template might not all be initialized. Each path has a topmost path element. Typically, the address information carried in this topmost element is relatively dynamic while the address information carried in lower elements is static and derivable from the canonical addressing standard or some other source. To convert a path template into a viable path one must initialize the path template's dynamically determined address info fields.

There will only be a limited number of possible topmost path elements. Path reports generated from the IPC name server, for example, will usually have paths whose topmost elements are any of TCP, IP, UDP, or PXP. Path reports generated from MONAD's node name server routines will have topmost paths which describe protocol groups. Path templates can be stored which describe the common paths to be used to access individual protocols (IPC case) or to access protocol groups (MONAD case). When the IPC name server, for example, needs to generate a path report for a connection-oriented call socket it only needs to find all path templates which have topmost elements whose protocols are bound to the call socket. One way to do this would be through a path template table.

A path template table is designed to make generation of path reports easy. To use the template table for this purpose one must know the pids of the topmost protocols or protocol groups which will be topmost in the paths to be generated. The template table itself consists of a string space and an array of records indexed by pids. The records in the array contain (minimally) two fields: the first a pointer to a string in the string space, and second the length of the string pointed to. Each string will be a path template list, i.e., it will be a list of path templates. The topmost path element in each path template will have the same pid value as was used to index the path template table. If the address info field associated with the pid must convey dynamic address binding information then these fields must be initialized inorder to produce complete path descriptions.

Normally, only the topmost path elements carrying individual pids will have uninitialized address info fields. The path templates carrying group pids won't have uninitialized address info fields and so won't ever require dynamic configuration.

GLOBAL DECLARATIONS

```
TYPE
   TemplateRecord = RECORD
      string_addr  : AddressType;
      string_len   : Int16;
   END;

VAR
   Templates : ARRAY [1, LAST_PROTOCOL] of TemplateRecord;


PROCEDURE GenerateReport (     topmost_pids      : PidArray;
                               associated_addrs  : AddressArray;
                               num_pids          : Int16;
                           VAR report            : ReportType );

{}
{ Abstract:
{  Many liberties are taken in specifying this algorithm.
{
{ Input parameters:
{
{  topmost_pids: An array of protocol identifiers representing
{     the topmost protocols to be found along paths to the
{     object about which the path report is to be generated.
{
{  num_pids: The number of pids in the topmost_pids array.
{
{  associated_addrs: An array containing the addresses to be
{     used in association with the pids in the topmost_pids
{     array.
{
```

```
{  report: A partially completed path report -- presumably the
{     version, report length, and virtual network address fields
{     have been initialized.
{
{ Output parameters:
{
{  report: A complete path report.
{}

VAR
   i           : Int16;
   j           : Int16;
   path_offset : Int16;

BEGIN
   FOR i := 1 TO num_pids DO
   BEGIN
      path_offset := report.report_length;
      pid := topmost_pids[i];

      CopyString ( templates[pid].string_addr,
                   templates[pid].string_len,
                   report[path_offset],
                   templates[pid].string_len );

      InsertDynamicBindings ( report[path_offset],
                              templates[pid].string_len,
                              associated_addrs[i] );

      report.report_length := report.report_length +
                                 templates[pid].string_len;
   END; { FOR }

END; { GenerateReport }
```

References — CHAPTER 5

[1] David D. Clark, "Names, Addresses, Ports, and Routes," RFC 814, MIT Laboratory for Computer Science, July 1982.

[2] Clark Johnson, "DS'83 Canonical Addressing Standard," HP-Internal Document, 2 December 1982.

[3] Bob Carlson, "HP-DSN Addressing Standard," HP-Internal Document, Revision 2 (proposed), 15 October 1982.

[4] Rick Bartlett, Atul Garg, and Craig Wassenberg, "DS'83 Network Architecture," HP-Internal Document, Draft- 7 April 1983.

[5] George Carter, "Node Naming Using MONAD," HP-Internal Memo, 8 March 1983.

[6] John Bugarin, Bob Carlson, Brian Lynn, and Craig Wassenberg, "HP-DSN Interprocess Communication External Reference and Protocol Specification," HP-Internal Document, revision of 14 March 1983.

[7] David C. Plummer, "An Ethernet Address Resolution Protocol," RFC 826, November 1982.

[8] Derek C. Oppen and Yogen K. Dalal, "The Clearinghouse: A Decentralized Agent for Locating Objects in a Distributed Environment," Xerox Office Products Division, Palo Alto, CA., October 1981.

[9] Andrew Birrell, Roy Levin, Roger Needham, and Michael Schroeder, "Grapevine: An Exercise in Distributed Computing," Communications of the ACM, Vol. 25, No. 4, April 1982, pp. 260-274.