Network Working Group Garrett A. Wollman, UVM
INTERNET-DRAFT 16 December 1992

NAP-based Addressing

A Strategy for Encoding Hierarchical Addresses in Internet Name Services

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the 1id-abstracts.txt listing contained in the internet-drafts Shadow Directories on nic.ddn.mil, nnsc.nsf.net, nic.nordu.net, ftp.nisc.sri.com, or munnari.oz.au to learn the current status of any Internet Draft.

Abstract

This document describes NAP-based addressing, a scheme for abstracting and encoding hierarchical addresses in a name service in such a way as to allow the entities responsible for various levels in an address to change their portions independently and scaleably. As a byproduct, it also defines a particular addressing scheme which the author believes to be appropriate for the existing model of Internet connectivity.

1 Introduction

This document is the result of some thoughts about the problems that host mobility and automatic address reconfiguration can give rise to in the Internet environment, particularly with regards to the Domain Name Service [DNS]. It was developed primarily in the PIP mold, where an address is a variable-length byte string which is basically hierarchical in nature [PIP], but it should be applicable to any other proposal with a sufficient number of levels in the addressing model. We believe that this model—because it was developed with an existing network structure in mind—can aid considerably in the transition from the current IP version 4, through EIPIP, to PIP [EIPIP].

This memo breaks naturally into two sections. In the first section, we will introduce the NAP addressing model and explain how it arises naturally from the real-world Internet. In the second, we will explain the adaptation necessary for this model in the context of the Internet Domain Name Service; the same ideas could easily be adapted for other name services, such as Sun's NIS.

Some of the ideas in this document come from Dave Clark's "route fragments" idea, posted to the IETF mailing-list several months previously; other ideas come from Steve Deering's geographic addressing model. The model described here is our attempt at providing the benefits of both ideas while allowing the actual numbers involved in the addresses to change in a way which is transparent to end-users, and still scales to meet the IP version 7 requirements.

1.1 Glossary

absolute level: The hierarchical level or tree depth of a particular node, relative to an arbitrary root node. When viewed from the root of the addressing hierarchy, the absolute level is *not* unique; for every "parent" of the node, there is an absolute level corresponding to the length of the path from the root to the node going through the specified parent.

address: We use the term "address" in the sense of Noel Chiappa; i.e., as an object with a well-defined meaning with respect to routing. An address does *not* necessarily correspond to an endpoint of a connection; a particular endpoint may have multiple addresses, or multiple endpoints may share the same address.[1]

EID: Endpoint Identifier.

Endpoint Identifier: We think of an endpoint identifier as a number which uniquely identifies the end of a transport connection. While this proposal does not actually require EIDs to operate, we feel that it makes more sense in terms

[1] For example, a multicast group. As a more far-out idea, consider a subnet which uses ES-IS-style address resolution on the basis of EID; all the hosts on such a subnet could be assigned the same address, with de-multiplexing by EID done at the level-1 routers.

of what we are trying to accomplish, to have EIDs, and so some of our examples will make use of them.

Level: See absolute level and relative level. When used unqualified, the latter is usually meant.

NAP: A NAP is a Network Attachment Point, in the most general sense of the word. That is to say, a NAP is any point where two or more networks meet. In our proposal, we are most concerned with the NAPs which are currently located at the regional internet providers, such as (for example) NEARnet, rather than the interface between the customer network and their provider. (However, customers which have more than one provider are considered ‘‘interesting’’ for our purposes, as well.)

Protoaddress: The 3-tuple (*local-part*, *provider-part*, *NAP-identifier*).

Pseudoroot: A tree root which corresponds to no actual node in the graph, but which (in conjunction with the relative level comparison) can be used to turn the graph into a valid tree.

Relative level: This refers to the hierarchical level or tree depth in the abstracted connectivity graph developed at a NAP or set of NAPs. Relative level cannot easily be quantified, but can be compared; we define A as *higher* than B if A is a long-haul provider connected at B, or if B is a local customer which can use one of the providers connected at A. (*Lower* is defined symmetrically.) Two nodes are said to have the same relative level if both are connected, but neither is higher than the other. (Thus, all routers at a specific regional or local network are considered to have the same relative level.)

2 NAP addressing model

The NAP-based addressing model is based on the widely known maxim, ‘‘Any problem in Computer Science can be solved by a layer of indirection.’’ This is combined with a few observations about the nature of the Internet as it stands today (and as we believe it is likely to remain throughout the lifetime of IP version 7), to develop the scheme we present here.

2.1 The Internet as it stands

The present global Internet can best be characterized as a loose confederation of regional and local networking providers, interconnected by some number of long-haul backbone providers. In many cases, these groups overlap; for example, AlterNet is both a backbone provider and a local provider in the regions which it serves.

Also in the present Internet, there exist what are known as Internet Exchanges, which provide for connectivity between various providers. There are some national Internet Exchanges, such as the East and West FIXes in the United States, and there are local Internet Exchanges located at many regional and local networks.

Let us consider a concrete example. The New England Academic and Research Network, NEARnet, is located in and around Cambridge, Massachusetts, and has tentacles reaching into every New England state. This is our typical regional network. Some of the areas served by NEARnet are also served by Performance Systems International, or by Global Enterprise Systems; neither of these companies interconnects directly with NEARnet, so we will not consider them. More importantly, some of the areas served by NEARnet are also served by AlterNet, which *does* interconnect with NEARnet. This means that there is a logical ‘‘location’’ somewhere in the structure of *both* networks which is the point at which they are attached.

However, NEARnet and AlterNet[local] are not the only carriers which meet here—if they were, New England would just be an isolated internet, with no connection to the worldwide Internet. So, in addition, we must also consider the long-haul carriers which interconnect at Boston or Cambridge; these include ESnet, NSFnet[T-1], NSFnet[T-3], AlterNet[lh], TWBnet, and some others which connect to individual sites only.

As it happens, AlterNet, ESnet, and NSFnet[T-3] are all connected at the same location, so that we can then say that they meet at a NAP located in Cambridge, Mass., on the campus of MIT. Furthermore, we can say that all other sites on NEARnet are ‘‘below’’ this NAP, and that the three networks just mentioned are ‘‘above’’ it. We could also pick on TWBnet and NSFnet[T-1], which meet at a NAP located at BBN; since there exists no natural ordering between the two NEARnet NAPs, we consider them to be equal in level with respect to the long-haul networks and client sites.

2.2 Problems of variable hierarchical addressing

To solve the twin problems of route aggregation and administrative overhead, it seems natural to use the connectivity mesh as a basis for address assignment. If we assume that there is a strict tree representing connectivity—with all nodes connected and a single root—the scheme for assigning addresses is trivial. Unfortunately, in today’s multi-provider Internet, this is emphatically not the case. Instead, we have a partially-connected mesh, with no well-defined root.

By taking advantage of the network characteristics we noted above, it is possible to define an arbitrary pseudoroot from which we can base our addressing hierarchy. Unfortunately, there is still the problem of partial connectivity, not represented in this structure, but we can assume for the moment that this is not a problem. (In the particular case of PIP, since a PIP address is not strictly hierarchical, it is relatively easy to work around this problem.)

We now have an addressing scheme which is capable of providing the desired scaling and abstraction properties. Great! Now we can all stop running all those nasty routing protocols! Sadly, however, this is the real world, and the connectivity mesh is constantly changing. Entropy ensures that any efficiency gained through an addressing scheme will quickly be wiped out by the routing overhead caused by exceptions. Hierarchical addressing can only work efficiently if nodes can easily be renumbered to reflect changes in mesh topology. Unfortunately, none of the name services in common use in the Internet community can deal efficiently with a situation in which large numbers of hosts change numbers at will. So, it looks like hierarchical addressing may not be the panacea after all.

2.3 Policy routing and other complications

The most important cause of the partial connectivity we mentioned above, is the rather nebulous concept of ‘‘policy,’’ which subsumes everything from acceptable use restrictions to national security regulations to budgetary constraints. In IETF discussions, several kinds of policies seem to be the most important:

- Don’t traverse provider *X*’s network.
- Only traverse provider *X*’s network.
- Don’t send any traffic through country *X*.
- Don’t send any traffic that costs me money.

There does not seem to be an easy solution to this problem which can be encoded in a packet and then sent off to be carefully interpreted by every router which might be affected. It is clear from the work in [NIMROD] and [SDRP], however, that it does make sense to allow hosts to request abstracted routing information for their own use; such hosts can then compose a source route (NIMROD) or use a special route-setup protocol (SDRP) to get the desired treatment. It is important for a new protocol to make this process reasonably efficient; in the current IP, it ranges from difficult to impossible.

2.4 Named NAPs as a solution

The solution we propose is one which allows variable hierarchical addressing to operate within the constraints of the existing DNS. In addition, it allows for provider-based route selection in the manner which seems to be the most useful from the policy point of view.

Our solution is a fairly simple one. Rather than storing an entire hierarchical address in the name service, we store a 3-tuple consisting of a *local part*, a *provider part*, and a *NAP identifier*. This 3-tuple is called a *protoaddress*. The local part is assigned by the site at which the system is located, by whatever means that site happens to prefer. The provider part is assigned by the site’s provider, in coordination with the other providers at the NAP, to designate the route from the physical NAP to the customer’s site. The NAP identifier is assigned by IANA (perhaps delegated to the Internet Registry), to designate the particular NAP in question.

It is assumed that there is some host at every NAP which listens to the IGP of every provider interconnected there. (This should not be very difficult, since the IGPs will only be carrying NAP identifiers and address fragments, not networks or host addresses.) It might use an EGP instead of listening in on an IGP, although Noel Chiappa wouldn’t like it. It may be necessary to reserve a well-known address or EID for this purpose (it is unclear at present).

In order to determine the address of a particular host, the following procedure is employed. First, the address is looked up in a name service, giving the 3-tuple above. The sending host can then query its *own* NAP for an address or addresses which will make it to the foreign NAP (assuming it doesn't already have one or more cached already), and by querying the foreign NAP it can turn the provider part into another address fragment. Given these fragments, the host can construct an address and begin to send datagrams.

A potential problem with this is that large numbers of sites getting connected to multiple providers could lead to NAP proliferation, potentially having the same problems as network-number proliferation in IP version 4. It is not clear whether the number of such sites would be so great as to cause a problem, but a simple extension to this mechanism to make NAP identifiers themselves hierarchical in nature, should solve this problem if it occurs.

There is an obvious optimization that can be made here, but it turns out to be not as useful as it might first seem. Since we are assuming that every NAP is omniscient (i.e., it knows how to get to every other NAP), we might as well just make the NAP identifier the top level in our hierarchical address, thus saving ourselves a step in address resolution. Unfortunately, this would disallow the long-haul providers from having their own internal address structure, which is clearly a Bad Thing. This sort of optimization also eliminates one of the easiest and most useful interfaces for doing policy-based routing semi-automatically.

Another obvious optimization is to replicate copies of the NAP identifier to address fragment mapping table nearer to source hosts. This seems to be much more useful, although some sort of database replication protocol would probably need to be invented for this purpose. (Question: how fast can we expect these fragments to change, and what kind of replication delay are users willing to accept?)

As a result of viewing addressing in the manner, the most common forms of policy routing simply fall out of the addressing structure. More complex forms of policy routing can still be done, provided that the network protocol is sufficiently flexible to encode the routes so generated. Also as a result of this form of address resolution, we can achieve a network in which it really doesn't matter what a host's address prefix is, because it is never explicitly stored in the name service; thus, regions of the network can be renumbered at will, without any site needing to change any part of their databases. This is big win.

2.5 Isolated internets

One problem which arises is the question of how can this scheme be made to work for isolated internets, which don't have connectivity to global name service and NAP databases. One possibility might be to simply permit the NAP identifier and provider part of a protoaddress to be null; in that case, a host would treat the local part as a site-local address fragment, and proceed from there. Thus, a protoaddress in an isolated internet would simply degenerate to a fairly standard hierarchical address, which is what is needed. This also solves the obvious bootstrapping problem (how does a host find out its servers' addresses without querying those servers to find out).

3 DNS extensions

In order to implement this new address-resolution scheme, some changes would have to be made to the domain name system. We will be primarily concerned with the conceptual changes (new record types and so on) rather than the actual format of data on the wire, which will vary depending on the particular network layer in use.

The DNS will need to be updated to contain protoaddresses instead of addresses, and for completeness, it would also make sense to have NAP identifier to and from NAP name mappings stored there as well.

3.1 New record types

There are two different kinds of new records: those dealing with host protoaddresses, and those dealing with NAP information; these are mostly independent of one another.

For host protoaddresses, we propose a record type of `PA` (for ProtoAddress). These will act in exactly the same manner as the current `A` records in providing a mapping from host name to protoaddress. The textual representation we would make as follows:

```
my-host    IN    PA    ( local-part provider-part NAP )
```

with encoding TBD. We feel that mobile host support can best be provided in this framework by simply providing a mobile-host server which can be queried for the current true protoaddress of the mobile host; this is represented with a `PM` record.

```
my-mobile IN    PM    my-mobile-server
```

This record would trigger the same auxiliary processing as a `CNAME` record, to reduce round-trips to the name server. For the purposes of transition, it would also be useful to be able to turn an IPv4 network number into two-thirds of a pseudoaddress; the host part of the IPv4 address would be added (as the local-part) to form the full pseudoaddress. Here's the `NPA` record to do that:

```
my-network      IN    NPA   ( provider-part NAP )
```

Finally, we need the equivalent of `PTR` records. This is somewhat difficult to do for general addresses—they vary from host to host—but easy enough for protoaddresses. The general address problem is one of the main reasons why EIDs are needed in an architecture such as this, so we will define two record types, one for EIDs and one for protoaddresses:

`*reversed-EID-text*.IN-EID.ARPA. IN REID *my-host*`

`*local-part*.*provider-part*.*NAP-id*.IN-PA.ARPA. IN RPA *my-host*`

The particular format of a *reversed-EID-text* will obviously depend on the nature of the EID as defined by the network protocol. The *local-part* and *provider-part* would be in the same format as they were in the `PA` record, but it is unclear whether the *NAP-id* should be the textual NAP name or the numeric NAP number, or perhaps both.

The NAP mapping is much simpler. We would define a new top-level domain, `NAP`, to contain all the NAP data, and add two new record types, `NAP` and `RNAP`, to contain the mapping information. These would be represented as follows:

`*NAP-name*.NAP. IN NAP *NAP-number*`

`*NAP-number*.NAP. IN RNAP *NAP-name*.NAP.`

This provides all the information needed to perform the mapping in both directions.

3.2 New query types

One problem that several people have noticed with any scheme where one host may have several different address types encoded in the DNS is that the different records may have different TTL values, so that some address types expire before or after others. The way we propose to work around this difficulty is to define a new query type in addition to the record types discussed above. We assume that the name server has some understanding of what kinds of records contain addresses and which do not. When a server receives a query for type `ADDR`, it is required to return *all* such records in its possession. Furthermore, it *must* adjust the TTL values of all records that it returns to reflect the minimum TTL of the collection. (Writers of SMTP agents might also want to have the same treatment for `MX` records; if anyone thinks this is a good idea, please write to the author.) Finally, when it caches an address as a result of a query for a specific type of address, such addresses must not be used to respond to `ADDR` queries (although the reverse is allowed).

Applications wishing to contact a host out in the Internet will then be able to make `ADDR` queries and be assured that the responses which are returned reflect the complete set of addresses that one particular host resides at.

3.3 Benefits for the EIPIP transition plan

One of the difficulties identified in the EIPIP transition plan was that of mapping an IP address to a usable PIP FTIF chain for forwarding. If this can't be done reasonably, then we can't solve the Routing Problem. (The current EIPIP specification turns an IP address into a two-element chain, of the form (IP network, IP host). This clearly suffers from the exact same scaling problem as IPv4 addresses themselves, with respect to routing.)

This scheme, by way of the address resolution mechanism and the `NPA` records defined above, provides a scalable mechanism for an EIPIP host or NAT box to turn an IPv4 address into a usable FTIF chain with enough hierarchy to solve the Routing Problem.

4 Conclusion

By breaking up the formation of hierarchical addresses into three independently-named segments, we can allow local routing protocols to operate without requiring the Domain Name System to contain rapidly-changing addresses or policy information. This mechanism can be used to help the EIPIP transition plan in translating IPv4 addresses to PIP addresses, and the DNS extensions that we propose will also help the multiprotocol Internet deal with single-homed but multiple-addressed hosts.

5 Security Considerations

Security considerations are not discussed in this memo.

Author's Address

For more information, the author of this document may be contacted via Internet mail:

Garrett A. Wollman
EMBA Computer Facility
250 Votey Building
Burlington, VT 05405

Work: +1 802 656 2926
Home: +1 802 864 8386
E-mail: Garrett.Wollman@UVM.EDU