Why Loc/Id Split Isn't the Answer

John Day
2008

"Well sir," said old Fairford, "they're English. And my experience with the English is that you can get them to do anything if you put it to them right.The trouble with the English is they try all the wrong ways first."
- John Masefield
The Bird of Dawning, 1933

"How Can You be in Two Places at Once, When You are Not Anywhere at All"
- Firesign Theater, 1969

Introductory Discussion

After a recent meeting, there was the usual bar discussion: This time on naming and addressing. The Internet has had major addressing problems for over 35 years. Mostly stemming from not having a full addressing architecture. Thanks to Moore's Law and studious rationalization we have convinced ourselves that we could ignore them. But it is not nice to ignore Mother Nature, eventually she gets even. And the longer it takes the more painful it will be. The high priority concern right now is that the kludge for multihoming is causing router table size to increase. This is only the most immediate problem. Others equally severe are waiting in the wings. We always say that the root of the multihoming problem is that IP addresses name the interface and we need to name the node to solve multihoming.

Doesn't loc/id split do that?

That has been the theory pursued by most of the field since the mid-90s. Although, it never had the feel of a right answer. Now after over a decade of everyone assuming that a solution based on loc/id split would be the answer, it can be shown that any solution involving loc/id split doesn't scale, and apparently can't be the answer. Let us consider what this is all about, and why Saltzer's distinctions of application name, node address and point of attachment do solve the problem and others, but loc/id split doesn't. First lets consider what the problem is:

Starting with the ARPANet and even in the phone system before that, addresses had always named the interface. There was not much reason for redundancy with phones, not to mention that was difficult to do electrically. For the ARPANet, it was also understandable. It was an experimental network, the first of its kind. It was going to be a bit of the old and a bit of the new. There were a lot of things to figure out. There was much about the ARPANet that was "beads-on-a-string." It was and wasn't layered, at least not in the way we think today. But it was modern enough, that we could think of it in the new model we were trying to work out.

We realized early that naming and addressing would have subtle effects on a network even if we weren't quite certain what they would be. The first of these came in 1972. I

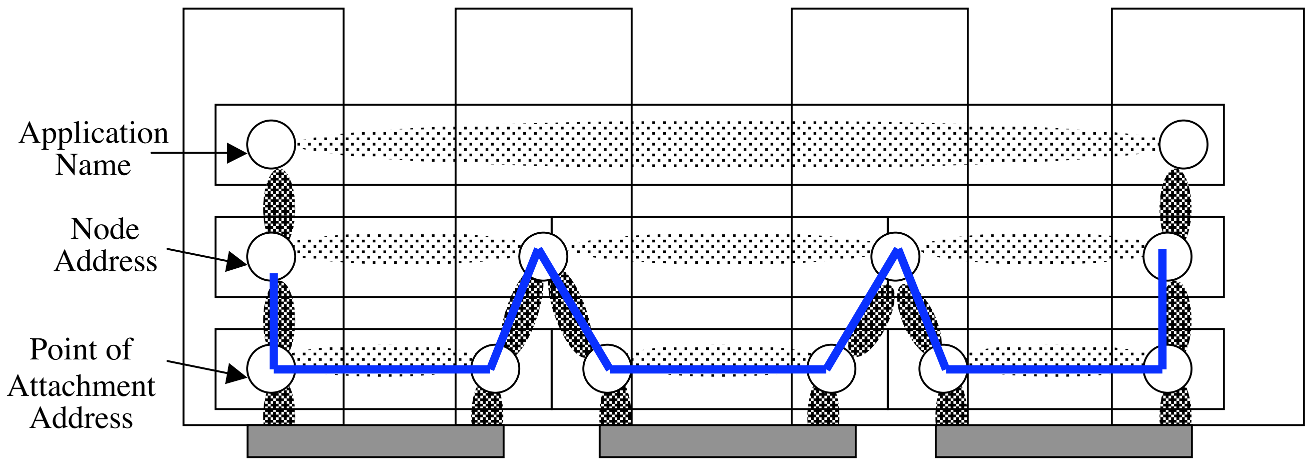

Figure 1. Saltzer's Model of Network Naming and Addressing
▬ A route in Saltzer's paper

remember Grossman[1] coming in one morning and told me that Tinker was joining the 'Net and wanted connections to two IMPs for redundancy. My first reaction was, "O, good idea." And a split second later, "O, #$&*!@, that isn't going to work!" To the 'Net it would look like two different hosts. The routing algorithm had no way to know that the two wires went to the same place! But a second after that the solution was obvious. We were OS guys. We had seen this problem before. We needed to name the node not the interface, we needed a logical address space over the physical address space. The lesson was clear: unlike with telephones and computers, in networks there is *always* the possibility of more than one way to get between two points. (I was sure Grossman had figured all this out when he told me.)

We rationalized that even though, we saw the problem and knew the answer, we didn't fix it when IP was split from TCP in the mid-70s, because, we rationalized, it was still too early to commit ourselves to a solution. We still had much we needed to understand about naming and addressing. However, we didn't realize that the Internet was settling into a period of stagnation and newcomers were assuming the existing model was right.

In 1982, Saltzer published his paper arguing that a network addressing architecture must have application names, node addresses and points of attachments yielding routes. Clearly, Salzter was using naming and addressing in operating systems as his model, a subject he was quite familiar with. The three levels correspond to application names, virtual addresses and physical addresses, with routes at the bottom, in the case of the computer, in the hardware.

In his view, networks were the same. There was a mapping from application names to node addresses, and a mapping from node addresses to points of attachment, and routes were sequences of nodes and points of attachment (the blue line in Figure 1). It is

[1] Gary Grossman was our boss, leader, friend, a composer, and the best system designer I ever knew, outside of maybe Grothe.

important to note that each kind of identifier names a distinct entity and that there is a mapping (that may change) between the names. This has the ring of the right answer and builds on the unity we all saw between networks as IPC and operating systems. [2]

Routes, of course, are calculated from complex graph theory algorithms the computation of which increases combinatorially with network size. However, it isn't the route that we want. We only want the address of the next hop from the route calculation to generate the forwarding table for this router. Our hope is that all other routers will do the same calculation with mostly the same data and come to the same or nearly the same conclusion. This is how the Internet does it. In Saltzer's model, the Internet looks like Figure 2.

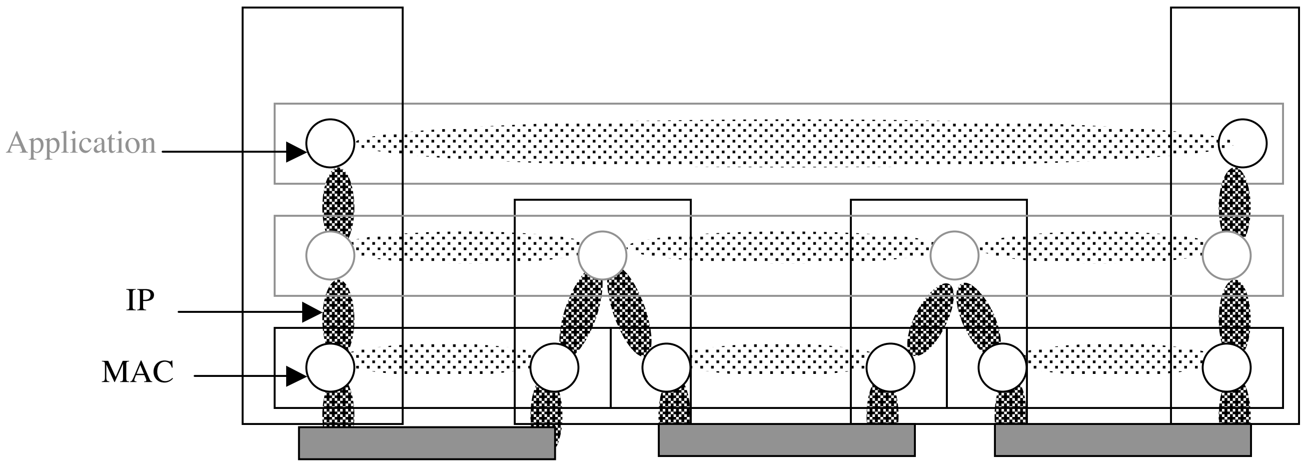

Figure 2. The Internet according to Saltzer's Model

The first thing we notice is that Internet only names the point of attachment and to add insult to injury, does it twice: The IP address and MAC address are naming the same thing. Now you may find this strange, but the *Internet* Protocol address is naming what appears to be a *subnetwork* point of attachment.

Shouldn't it be naming something that has to do with internetworking, not the subnet? You would think so. But it is clear that if we are only naming the points of attachment, then there is no way to do multihoming without some sort of kludge. The analysis of the Network Layer in OSI found that it divided into 3 sublayers: 3c) Subnetwork Independent Convergence, the so-called Internetwork Layer; 3b) Subnetwork Dependent Convergence, transition from the technology-dependent to technology-independent; and 3a) Subnetwork Access, nominally the Point of Attachment.[3] From this model we can

[2] It is important that in these figures, the "layers" are not given names.

[3] We will ignore as a rathole the obvious question of what was the difference between subnet access and the data link layer.

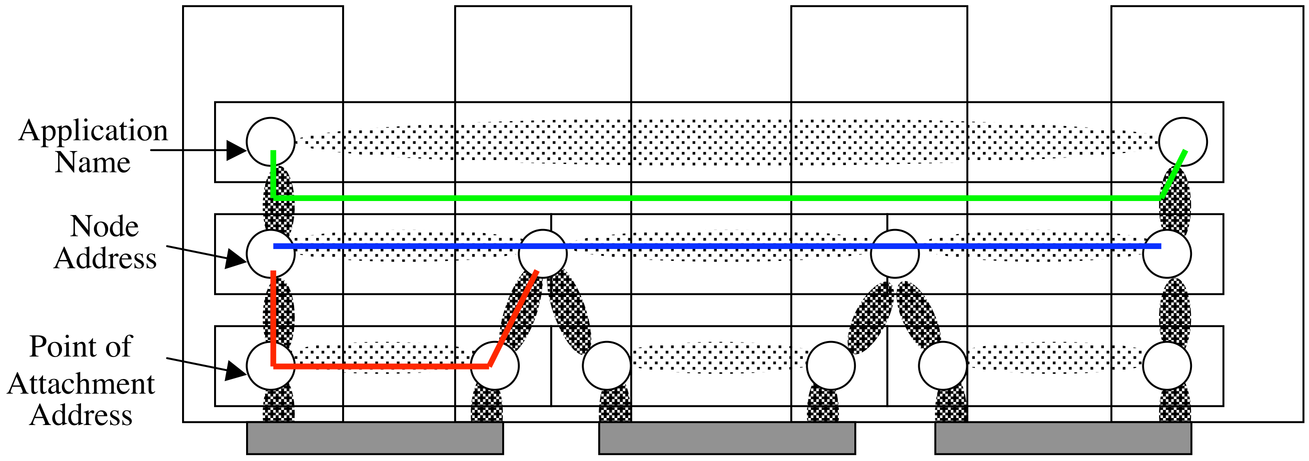

Figure 3. Saltzer model extended to handle multiple wires between nodes.
🟦 A route is now a sequence of nodes
🟥 There may be more than one path between adjacent nodes.
🟩 The application name to node address mapping, or directory.

see the somewhat schizophrenic nature of IP: While IP resides at 3c, its address is at 3a. This seems made for creating problems.

But Saltzer missed something, through no fault of his own. It hadn't occurred yet. There may be more than one "wire" between next hops. In fact, there might be whole networks. In networks, there is often more than one way to get some place. The simple version of this occurred in the late 80s, when Internet traffic was growing so fast that it was not possible to get links of sufficient bandwidth to carry the load between adjacent routers. The obvious solution was to run multiple wires between adjacent routers. This creates real problems. If we approach the problem naively the number of routes skyrockets. Assume the above figure is a string of 4 routers with one wire between each one. How many routes are there? One, of course. Now put 4 wires between each node. How many routes are there? $4^3 = 64$. And that is only a simple series of routers, if there were branching in the graph the number becomes very large very quickly.

Obviously workarounds were found to avoid this explosion, but multi-path routing was a big topic in the late 80s. But they were just that: workarounds, not solutions. Basically the Internet doesn't support multihoming for routers, let alone hosts.

To incorporate this into our model, we need to extend Saltzer's model a bit. Saltzer's model describes networks. We need to describe *internetworks*. This is a more general case than operating systems or networks. The obvious solution is to make it a two-step process: First, routes are sequences of node addresses (the blue line in Figure 3). Second, once we know the next hop, then we need to know the node address to point of attachment address mapping of our nearest neighbors (the red line in Figure 3), so that we can pick which path to the next hop. (This does not mean that forwarding is a two-step look-up. It means we logically go through these two steps to generate the forwarding table. Although it does imply that we might load balance by simply doing the last step more frequently, rather than recalculating the route every time.) This is the model for internets. Already we see that the Internet is really only a network, or at best a catenet, but not an Internet.

But before we leave this, note that the mapping to determine path to the next hop (the red line in Figure 3) is the same mapping as the application name to node address mapping (the green line). What we normally refer to as the directory. The green line and the red line are different instances of the same mapping! They are both mappings involving nearest neighbors, relative to their layer. The structure repeats? How curious.

Question: *Is there an analog to this in OSs? We haven't moved the routing in this case. There may still be routing among the PoAs (it is a relative relation), it is just that there is routing among node addresses too. The PoAs are routing between "next hops" among the nodes. The nodes are routing about the "next hops" of the applications. Is there an analog in OSs?*

Little Known Facts about Addressing
Before we go on, let us get a few misconceptions or not generally known bits of knowledge about naming and addressing cleared up (See Chapter 5 of PNA):

1) Virtually all identifiers used in computer science are used in some sense for *locating*. In fact Saltzer appears to agree. In his earlier work on addressing [1977] (primarily on memory management), he defines "resolve" as "to *locate* an object in a particular context, given its name." (emphasis added). The "particular context" is the scope of the name space or the set of objects whose names are taken from the name space. To use a name is to locate an object. Basically, an object can't be located without identifying it and vice versa. This not good. Already there is evidence we are pursuing a false distinction. If "identity" is useful, then it is locating.

   This includes application names. The hierarchical pathname is a locator in a quite different kind of topological space. But it still has the property that given two names, we can tell if they are "near" each other for some concept of "near." We have been somewhat sloppy in characterizing them as location-independent. Location-dependent does not need to be spatial.

2) There are only two and half means to resolve names, i.e locate an objects given its name: 1) exhaustive search, or 2) information in the name itself provides hints for narrowing the search. In addition, these means may be used for indirect resolution of names (the half) where a name or part of the name is used to find a reference to the object (recursively).

3) Addresses need only be unambiguous within the scope of the layer/DIF they are used in. Scope tends to increase as we go up through an architecture. This has been clear since at least the early 80s.

This says that MAC addresses are 3 times longer than they need to be. But okay, no harm. MAC addresses being globally unique, allows them to double as a serial number and makes configuration easier. However, addresses are not serial numbers and as we will see this also creates security problems. (Concerns over addresses being used as personal identifiers occur when addresses have greater scope than necessary. And we develop bad habits of using addresses for things they were never intended.)

4) Addresses in adjacent layers should be completely independent. Lower layer addresses should not be used to build higher level addresses. In other words, don't form an address by concatenating it with the address of the lower layer.

   It seems a natural thing to do. Make it hierarchical. This was the general thinking in the 70s. Early drafts of the OSI model made this mistake. But work in the early 80s on the Naming and Addressing Addendum to the OSI Reference Model turned up the error and fixed it sort of. (There are other things wrong with the OSI concept of address.)

   Why is this a bad thing? It is obvious once you think it through. This is a *pathname*. It hard allocates the bindings between layers. It names a single path through the layers. Precisely what we are trying to avoid. Furthermore, it assumes that the mappings between the addresses are static, which we know won't be the case. But it is still proposed, most recently by IPv6 and in several academic papers. If the device is multihomed then there will be multiple addresses because there will be multiple paths.

   The counter-argument is made that this is pedantic. We don't have to *interpret* the address as a pathname. No, we don't. But some not-so-bright hotshot will do it and scream bloody murder when his application breaks because he made an unwarranted assumption. Better to do it right so no one is tempted to do it wrong.

   This also creates a security problem if it is done with MAC addresses as IPv6 does. Since MAC addresses are serial numbers and uniquely identify the interface/system, they can be used to associate data with particular originators. Normally this is not much of a problem. The MAC header is removed when the data leaves the local LAN. But thanks to IPv6, the identity of the source system stays with every IP packet until it is delivered. Of course, if the MAC address is only unambiguous within a LAN segment, then it wouldn't be a problem. Better to do it right so that no one is tempted to do it wrong.

5) Naming the host has nothing to do with the problem of forwarding PDUs.

   Tradition makes it an easy confusion to make. But naming is one of those topics where we must be very precise. Sloppiness will get us in trouble. All addresses are used to deliver PDUs to some Protocol Machine (PM), at most the IPC Process. The fact that the PM resides on a "host" is purely coincidental. We can now define what an address names:

An address names the locus of processing that removes the PCI (header) of the PDU containing the address.

This observation is important for mobility and ensuring that all address assignments are potentially transient. If the semantics of the address is overloaded in any way, it is the idea that it names the "host." Addresses are not serial numbers for equipment. If an identifier for the system or the hardware is desirable then define such an identifier, but it has nothing to do with *addressing*.

6) Addresses are identifiers *internal* to a DIF/layer. The address is an internal identifier used by members of the layer to coordinate their behavior.

   OSI defined the intersection of a layer and a system as a "subsystem." (One of the few useful definitions they came up with.) In other words, all the stuff including whatever protocol machines and management, that was associated with the operation of the layer in a system. In PNA, a subsystem is an IPC Process. Its external name is an application-name. An internal name is assigned to each IPC Process to facilitate coordination with other IPC Processes. There is no reason for an address to be visible outside the layer/DIF.

   Addresses are synonyms for the application-names of the IPC Processes of a DIF taken from a name space with less scope. They are short names for the IPC Processes that are members of the DIF. Application names are globally unambiguous, hence very long, and structured to be resolved in the scope of the global application name space. IPC is a very performance sensitive distributed application. Using application names in all PDUs could incur considerable overhead and look-ups would be more inefficient. Hence considerable advantage can be gained in the internal operation of the DIF, if application names are not used for internal coordination of DIF operation. Instead, names (traditionally called addresses) with much less scope (shorter) and structured to facilitate resolution (located) within the DIF are used.

   What about addresses being location-dependent? That turns out not to be an inherent requirement, but only one that is generally introduced for large DIFs. In other words, for small DIFs exhaustive search (a flat name space) is adequate. For larger DIFs there are advantages to structuring the addresses. In general, this "structuring" takes the form of imposing a topology on the name space, such that a "nearness" function can be applied to addresses.

   In an early draft of the PNA book, I had used another term for these internal identifiers. Reserving "address" as a term for location-dependent identifiers to be introduced when their interpretation actually depended on them being location-dependent. I wanted to make the point that the requirement for location-dependence only arises under specific conditions. I finally dropped it thinking the subtlety would be lost on most readers and would only confuse them. But it is still the case that addresses for small DIFs may be assigned simply by enumeration, i.e. flat, while addresses for large DIFs will often indicate location, relative to other members of the DIF, which may or may not have any relation to spatial location. This is where considering topology as an abstraction of a graph may be useful.

The common routing algorithms do not use the location dependent nature of an address in their calculation. The addresses are only used as labels for the nodes of a graph. As the DIF becomes larger, we impose structure on these identifiers so that we can aggregate them into fewer cases so that our calculations remain tractable. I conjecture in Chapter 8, if we develop algorithms for generating forwarding tables that do use location information embedded in addresses it will greatly simplify the forwarding table generation.

Then the primary purpose of an address is to distinguish where the PDU is delivered? Correct. *So it is primarily an identifier?* Correct again. *So loc/id split is a red herring?* Some what, but that alone doesn't make it wrong. There are distinct objects to be identified and if the scope within which the objects exist is sufficiently large to warrant imposing a location dependent naming scheme on them then fine. The important thing to be clear about is what each name names.

7) The designation of node address and point of attachment is relative. One layer's node address is the layer above's point of attachment. Or to put it differently, to any layer its addresses are node addresses. Problems only arise when a layer/DIF tries to use someone else's identifiers. (The repeating structure we saw in Figure 3.) With a repeating structure, the distinction becomes moot. We can throw away another ladder.

8) The last two points implicitly imply that the collection of addresses at the same rank (level) will form a graph. For the lowest rank, the collection of graphs of the DIFs will be the physical graph of the network. For the other levels, the graphs will be an abstraction of the physical graph.

9) Given that every time we don't do it we get in trouble, we must obey the rule that an address should be used for one purpose and one purpose alone: to identify the entity to which PDUs should be delivered. Nothing else. Any overloading of the semantics of an address leads to trouble.

Back to the Problem

Now we are ready to get back to the problem at hand.

Just a minute. Okay, so you didn't understand enough in 1975 to fix multihoming but you certainly did, say after 1985! Why wasn't it fixed?

It is curious, isn't it? A Department of Defense network touted to be able to survive nuclear attacks and it doesn't support redundant connections to the network! It takes the concept of military intelligence to new heights, doesn't it? Over the years, various excuses were given:

- *Not that many hosts need to be multihomed.* That was certainly not true of DoD networks, who were paying for the development. It might have been true of the operational Internet, but even in the Internet the ones that do need to be multihomed are the ones all the others want to get to. Typical of spoiled boomers,wanting the benefit without the cost. ;-)

- *Since so few need to be multihomed, it is "unfair" to require the ones that don't to incur the cost.* This is follows on the last one putting constraints on the solution. It is curious for two reasons: First, it represents a patch mentality that assumes a new mechanism is required to solve the problem, rather than a solution inherent in the structure that would solve the problem at no cost. Second, casting the problem in this way (that any burden only be on the multihomed system) virtually guarantees an asymmetrical (ugly) solution and hence not acceptable, giving this rationale an air of disingenuousness.

- *The multihomed lines will be from different providers. Because peering points are so sparse by the time the routing tables respond, the data will have to travel so far the TTL will expire anyway.* (Providers currently have very few peering points between networks. For example, traffic between two companies in the same office park in a Boston suburb on different providers will be routed through northern Virginia.) There are at least two problems with this argument: First, not all multihomed lines are from different providers, in fact, not even most of them. Second, it assumes that the number and density of peering points will never change. That there will never be sufficient traffic to warrant peering points at a finer granularity. Actually such sparse peering is not good for the survivability of the network in the event of disasters. There are many reasons to not expect these assumptions to persist.

Why does having lines from different providers matter, other than getting from one provider to the other? If all hosts on the same provider had the same prefix, then all of them could be handled by storing a single route in the routing table, rather than for every host on the provider. Hence when CIDR (Classless Interdomain Routing) was adopted, large blocks of addresses were assigned to providers for them to assign to their customers. This meant that addresses to the same provider could be aggregated to a few routes: the nearest peering point to that provider. A host multihomed to two different providers will have addresses with two different prefixes and increase router table size. Or a so-called provider independent address (in IPv6) and/or an AS number (in IPv4) is assigned, which also increases router table size.

So we need provider-independent addresses as well? Why are IP addresses provider dependent in the first place? It seems that Internet *addresses, if anything should be provider-independent!* Partly because they are point of attachment addresses, like physical addresses in a computer and partly because the IETF makes decisions based on emotion, not logic.

Surely not!? 'Fraid so. The change was attempted in 1992 with the ROAD process. The proposed replacement for IP would have named the node instead of the interface. But there were probably less than two dozen people in the IETF who understood the problem, even though it had been known since 1972 and the analysis published in 1982. The protocol, CLNP, being proposed to solve the problem was developed by Internet people working in OSI. They had taken Saltzer's paper literally and were proposing to name the node to solve the problem uncovered in the ARPANet. The IETF would not adopt anything from OSI, and not do anything that protocol did, even if it was right. Consequently, one of the ground rules for the IPng was not to change the semantics of the IP address. It would continue to name the interface. If routes need to be aggregated (which they do) and naming the interface is a requirement (not really) then provider-based addressing is the consequence.

In other words, they cut off their nose to spite their face. Pretty much. Especially given that the protocol was already deployed in the routers, the last 15 years of machinations to get to the current crisis with IPv6 were completely unnecessary.

Wait a minute! You mean that this protocol wasn't just a proposal. It was already implemented and deployed! Yes. *So billions of dollars have been spent and valuable time wasted unnecessarily for something that won't work and now threatens to pull the rug out from under the world economy!* You can't make this stuff up.

Nine months after the decision was made, it dawned on the IETF participants that naming the interface implied that addresses would have to be provider-based so that routes could be aggregated, which further implied that if an organization changed providers, they would have to renumber their network. There was a huge hue and cry, more than 250 messages over a holiday weekend. But they had done it to themselves. Naming the node would have allowed aggregation on provider-independent addresses and avoided the problem entirely.

After a few years and considerable concern about the problem, O'Dell proposed GSE [O'Dell, 1997], which I always assumed was a case of 'if they wouldn't do it right, maybe they might accept backing into it a bit.' O'Dell proposes a solution that says the IP address should be separated into "Routing Goop" and an "Endpoint Identifier" but tries to avoid saying explicitly what each names. That wasn't accepted either. The rejection of GSE basically convinced the clueful to give up and move on to other things. If the IPv6 group was going to insist on digging themselves a hole, give them a shovel. Now they want a bail out.

Things lay dormant for a few more years festering, but any time there was a discussion of naming and addressing Saltzer's paper was mentioned as required reading, as well it should be. Then Chiappa begins to write notes arguing that naming the interface isn't enough, we need to also identify the "endpoint." Although, he is a bit vague on what an "endpoint" is. As we saw, the IP protocol is considered to be the Internetworking Layer (the top of layer 3) but it names the subnet point of attachment at the bottom of Layer 3. This doesn't leave many places for an endpoint identifier.

So where is it? The Transport Layer? Not really, each transport flow/connection is independent, there is no common entity there. Ironic, but the most canonic of layers isn't really a layer at all. *Huh?* Sorry, rat hole, an interesting rat hole, but a rat hole for this discussion, lets leave that for later.

But the addressing problem was still there. Then in 1999, the IRTF initiates the Name Space Research Group. It became clear that there was a lot of folk knowledge about naming and addressing, but not much critical thinking. Then abruptly about the end of the NSRG deliberations, references to Saltzer's paper largely disappeared and loc/id split is all that is mentioned. Of course, no one noticed that Saltzer had covered the network case, but not the internetwork case. There was a missing case. When I asked Chiappa why the references to Saltzer had disappeared, he said that they had 'moved beyond' that paper. *'Moved on'? But to where?*

Finally, we can go back to what is wrong with loc/id split.
First what is loc/id split? As we said, the official story is that we have been overloading the semantics of the IP address, requiring it to be both locator and identifier of an endpoint. So we should separate the IP address into a locator part, which is hierarchical and an identifier, which is flat, like a MAC address.

Wait a minute! Saltzer said that using a name, i.e. resolving it, is locating the object named, so what is this distinction between locator and identifier, if every name locates something? You are catching on. Maybe they just don't use the identifier?

But even so Saltzer said we needed three things, this gives us two. How does this solve anything? If a host is multihomed, it would have at least two IP addresses (still points of attachment) with different locator parts, but the same identifier part. The locator part locates the POA; the identifer, the endpoint.

Doesn't this break your rule #4 above? So it would appear although strictly speaking it is all in the same layer or is it? Lets see where this goes?

Okay, What does the endpoint identify? It is hard to say. Chiappa first distinguishes between the "Network Layer" and the "Internetwork Layer." This is analogous to

OSI's distinction between Subnet Dependent and Subnet Independent. But Noel has a problem, if he stops here then the locator is the Network Layer (IP) address and the endpoint identifier, the Internetwork identifier/address. This clearly won't do. An IP address can't be just a Network Address. (The problem is clearly telling him to route on the node, not the interface. But since the horrible blow up of IPng, he knows that won't be acceptable.) So then Chiappa proposes that that the Internetwork Layer is subdivided into two sublayers. (epicycles within epicycles) This gets the IP address into the Internetwork layer, but he still has a problem.[4] He has two things that should be one. And he is on the verge of an infinite regress. So now he invokes "fate-sharing" to make the two things one, i.e the host or the stack.

That sounds reasonable.

There is a flaw. Fate-sharing assumes that when there is failure among entities in the same fate-sharing region, they all fail. They all share the same fate, like a system crash. *Right*! That means for fate-sharing to be applicable to this problem, we require the converse, if part (interface) of the system fails, then the whole system fails. In other words, the failure of any interface implies the whole system fails. Precisely the property we don't want. (Fate-sharing tries to be the shamrock, but no luck.)

So the argument is flawed . . .rather badly it seems. 'Fraid so. He seems to be trying to find a way to have a name for the host while still routing on the interface.

But you said naming the host was irrelevant? Yea, interesting, eh? Some proposals say it is the "Transport address," but no model including OSI has had anything that could be called a Transport address. The independent nature of the Transport Layer protocol machines makes it hard to imagine what it would identify. A globally unique flat identifier for something as transient as a transport flow doesn't seem to be what was intended. One recent paper [Baker, 2008] characterizes the endpoint as the end of a transport connection, but that is identical to an IP-address and port-id. But if the port-id is extracted then endpoint-identifier (EID) must name something common to all of the transport flows and to all of the interfaces, which it doesn't.

So it is not clear whether the loc and id name the same thing or different things. Are there two names for the same thing or two names for different things? To solve multihoming we know we need to route on an identifier for the common place that

[4] This whole thing reminds me of a scene in the movie "Nuns on the Run." Eric Idle and Robbie Coltrane are two low level thugs on the run from their gang, the police, and the opposing gang. They find themselves holed up in a girls convent school disguised as nuns. (Yes, the possibilities are endless!) Idle has been picked to teach a religion class on the Trinity. Of course, he knows nothing about it and Coltrane is trying to explain (based the lessons as a boy by his Irish priest) how if there is one God, can there be a Father, Son and Holy Ghost (It is like a shamrock: one thing split 3 ways). After going around and around, the scene ends with Idle saying, "But that doesn't make sense!" and Coltrane retorting with a shrug, "It doesn't have to make sense! It's religion!" At which point the crucifix on the wall behind them falls off , a great rimshot! (Later in the class, Idle becomes flustered and blurts out, "God is like a shamrock: small green and split 3 ways.")

all of the traffic comes to from different interfaces. That argues for two names for two things. The only question is what are the two things?

Since there may be more than one locator related to a single EID, that would seem to imply that the interface is the provider dependent part the traditional view of the post-CIDR IP address. Then the locator corresponds to the point of attachment. It would seem so, wouldn't it?

That means that the node is the id! Given the huge IPv6 address, we have plenty of room. Rather than create a whole other protocol, we just carry both parts in the same field. What is wrong with that? What does the id name? *I don't know. The . . . the host?* Uh . . .

O, you are being overly pedantic, splitting hairs! We can build it, who cares what it names? That seems to be what a lot of people thought and they were proposing various approaches and building them. In 2006, it was noticed that router table size was growing due to multihoming and this time Moore's Law wasn't going to bail us out [RFC 2984. 2007]. After some deliberation, several proposals were pursued. Then not along ago, Dave Meyer uncovered a couple of problems: Locator-Path Liveness and Site-based State Synchronization. [Meyer and Lewis, 2008]. He found that in implementing the map/encap LISP proposed by Cisco that these two problems lead to the need for path discovery. Path discovery is known not to scale, so neither would LISP. He suspected this would be true of any approach based on loc/id split.

WHAT!? Hadn't anyone noticed? They had been saying loc/id split was the answer for 15 years! Hadn't they worked it out? Apparently not. I know I never bothered.

What!? How come? Why? I knew it wasn't the right answer. Why waste the time?

But everyone said it was the answer! There are many interesting problems in networking. Too many to waste time trying to figure out why all the wrong ways are wrong. There is too little time to figure out all the right ways.

But why is it wrong? Like most things, it is pretty obvious once you think about it: The locator locates the wrong thing. *What?!* Remember, we must be very precise about what is really going on here. To solve multihoming, we need to route packets to the ultimate destination of the packet.

The locator. No, the identifier. The identifier identifies whatever it is everything arrives at regardless of how it gets there. The locators differ depending on how it got there. So the locator is a point of attachment. It is on the *path* to the identifier. The locator doesn't name the ultimate destination of the packet. It names a point on the path to the ultimate destination. So in a sense it isn't surprising that Dave ran into issues involving path discovery. Loc/id split required routing on an identifier

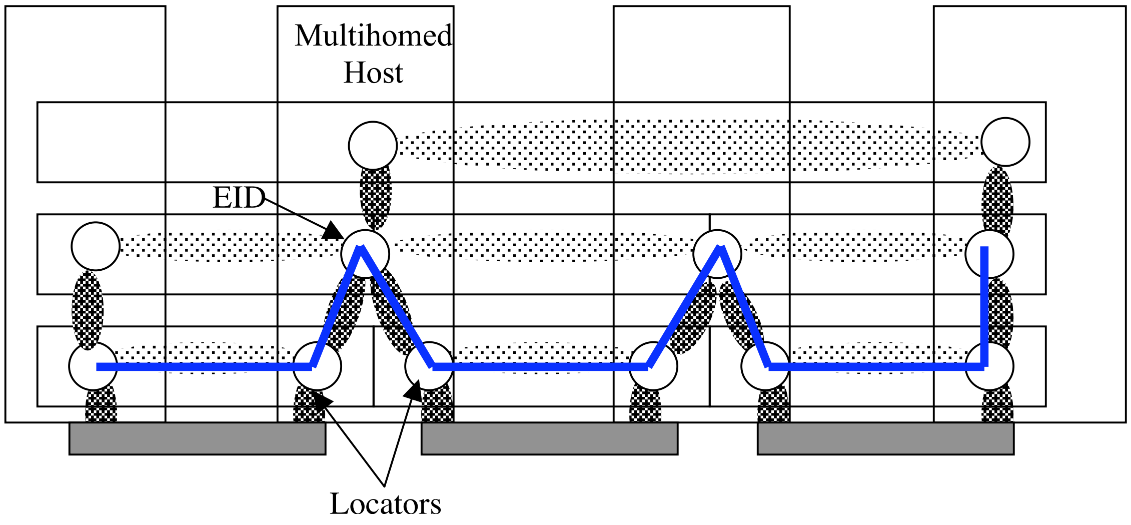

Figure 4. The Locator is locating a point on that *path*, not the destination. It is therefore reasonable that we should be encountering path discovery problems.

that was path dependent. We need to locate the identifier. We need to be routing on the identifier.

So the locator should be locating the identifier? Yes, it *is* what you are trying to find: the node. Locator is the network address and the ID is the Internet address to use Noel's terms. In PNA terms, they are simply different DIFs.

So should the loc be the id and the id, the loc? That depends on the nature of the lower layer. Remember the implication of 5) above in our list. If the lower layer does routing and its scope is large enough to utilize location-dependent addresses for aggregation, then it should be a locator would be free. If the lower layer does routing and its scope is small, it might be flat, essentially an id.

So we should be routing on the node and maybe on the point of attachment too. Correct. Remember the case that Saltzer hadn't seen yet? It is key to understanding this.

But that says, that CLNP was right all along! Given our understanding in 1992, Yes. And worse, it was deployed in the routers in 1992. Development was already done. Not only would mulithoming be much simpler but support for mobility as well. We could have spent the last 15 years on transition rather than going down a blind alley until we have a very short fuse on a major crisis. There was a joke that circulated in the Eastern Block as the Soviet Union was about to collapse that communism was the longest most torturous road from capitalism to capitalism. It would seem that IPv6 may be a similar road to nowhere.

Not sure how funny that is, given that it also means that since IPv6 and IPv4 name the interface, then it is very likely that no solution to multihoming for either one will scale. It would appear IP is fundamentally flawed. So it would appear.

How did this happen!? That is something people will study for a long time. This has all the earmarks of a textbook case of groupthink, especially given that the right answer was known at the outset. There was a small group that knew that routing on the node was the answer. But the vast majority really had no understanding of the issues of naming and addressing, especially among what became the IPv6 group. Also it seems they were never taught how to do critical thinking. Not realizing that the locator/identifier distinction was a false distinction or that fate sharing didn't apply. These are fundamental failures of education. The method was 'find a construction that seems to work,' rather than the scientific method of figuring out what the problem is telling you is the solution.

I thought scientific method was doing experiments. Yes, that is the second step. First, you need to have a theory to disprove.

Don't you mean prove? No, disprove. We can never prove a theory correct, we can only disprove theories. All theories, whether Newton, Maxwell, string theory or PNA are working hypotheses.

Basically it was an artisan approach. Like building a cathedral without a blueprint. Works okay for small ones; but for the large ones they tend to fall down. The IPng process created so much ire and stress on both sides of the argument that the best people simply went off to do other things.

So basically loc/id split is post-IPng trauma? That is a good way to put it.

But I think there is a deeper lesson here. For the past quarter century, we have not needed to obey principles. Given the effects of Moore's Law and that software is so malleable, an attitude had arisen that there was no one way to do anything. Almost anything could be made to work. In this case, it manifests itself with people thinking if they won't name the node then we can always solve the problem a different way, i.e. loc/id split. Although by 2000, the attitude had shifted to the almost universal belief that the current structure of the lower layers was finished. When in fact it never had been. (This is another common tactic in modern research: create a mess, not really get any answers, declare victory, and move on to make a mess somewhere else.) What we have uncovered here is a case where there is no other way to solve the problem. Computer *Science* is reaching the point where we must obey the principles. To make matters worse, Moore's Law has the effect that if we ignore the principles until the effects confront us (as we did in this case) the costs will be staggering and the longer Moore's Law masks the problem the worse it will be.

When does this crisis hit? Hard to say, there are projections but we have never been too accurate with them. Early estimates said 3 – 5 years. Running out of IPv4 addresses in a couple of years will make it worse. If I had to guess, about the time the economic recovery starts.

That is really frightening! It is indeed. *So we must be careful to get the principles understood early?* Yes, except for the past 25 years, we have not been training computer scientists to do scientific theory. Look at the rash of clean slate, new architecture, and GENI related papers, they are all trying to build a piece of something, not understand the fundamentals. I have even seen refereed papers that make teleological arguments. *You're kidding! In this day and age?* What is scary is not that the authors made the error, but that none of the reviewers caught it. This indicates an entire field not trained as scientists. Computer Science has been training artisans, not scientists. The principles of networking and especially the theory of naming and addressing are not taught in a single textbook. All that is taught is how to use IP and subnet masks.

So what now? Too early to tell.

Who is going to figure this out? Surely, we aren't going to expect the same people who got it wrong for years to suddenly get it right. Can we trust the infrastructure of the world economy to these people? I don't know. The utter unanimity of the groupthink makes it clear that it is going to be difficult to back out of this blind alley. It is not like there were two schools of thought and one was shown to be wrong. There was for the most part only one. We have two decades of training people the wrong way. No, we shouldn't expect the people who got it wrong to figure out what is next. There is a real need for new blood and for new thinking. But where it is to come from is not at all clear.

The ITU is no more clueful than the IETF. The wrenching changes forced on them by the Internet have not been sufficient to fundamentally change their world-view. They have not chosen to leap-frog the IETF, but have proven to be rather plodding followers. IEEE 802 seems to be in disarray, creating projects faster than they finish them. It seems like a case of 'widgets gone wild.' The complexity is skyrocketing.

To complicate matters, there is a well-known phenomena, that when a new model is proposed, the majority have a reaction of "O, now I see it!" But they don't really have the depth of understanding and they get it wrong. We have seen this over and over in networking and elsewhere. Unless we are careful, we will see it with this as well.

Does the old OSI Reference Model provide us any guidance? Not at all. Outside of the simplest definitions like PDU, PCI (header), or subsystem everything else is flawed as well. *Really?* Yes, the important definitions like connection, address, etc. all follow the PTTs misconceptions of networking and lead to complexity not simplicity. The only useful concepts I have found (and they aren't in the Model) are the insight distinguishing application process and application entity; a simple version of ACSE combined a version of CMIP that folded in concepts from HEMS as a common application protocol is interesting (scope and filter are a constrained form of map/reduce) and maybe a couple of other things. But by and large there is little in OSI worth resurrecting.

But doesn't creating simplicity in one place merely move the complexity elsewhere? That is what the unimaginative would like you to believe. What better way to maintain the status quo? That attitude and Moore's Law is what turned us down this blind alley in the first place. There are many more ways to get something wrong than to get it right. Our reliance on committees, whether for standards or for research, creates groupthink. If we all say the same thing then it must be true. It fails to recognize that science is not democratic. We aren't looking for the view of the majority, but just the one that solves the greatest breadth with the fewest concepts, regardless of whose sacred cows get skewered. Just because most of the ways you may first to try move the complexity around is no reason to believe they all do. It only takes one to be very right.

What is it? Fast, Elegant or Cheap; Pick 2. Yea, the refuge of mediocrity dressed up as pragmatism. An excuse to not put in the extra effort required to get all 3.

So do we go back to CLNP? We have learned a lot about network architecture in the intervening years. CLNP would essentially have us take a step backward at this point. At a time like this we should be moving forward, not backward. Turns out we don't need a protocol like IP or CLNP. Things get much simpler and the structure much more powerful if we rearrange things a bit.

WHAT!? It seems that 30 years ago we split TCP in the wrong direction. Things get much simpler if you split it the other way. But it is getting late. Lets leave that discussion for another time, shall we?

References

[Baker, 2008] Baker, Fred. Implementing GSE using IPv6/Ipv6 Network Address Translation. Internet Draft, draft-baker-nat66-gse-00.txt, Nov. 2008.

[Chiappa, 1999] Chiappa, Noel. Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture. Internet Draft, 1999.

[Day, 2008] Day, John. **Patterns in Network Architecture**, Prentice Hall, 2008.

[Meyer and Lewis, 2008] Meyer, D and Lewis, D. Architectural Implications of Locator/ID Separation, Internet-Draft, draft-meyer-loc-id-implications-00.txt, Dec 2008

[RFC4984] Meyer, D., Zhang, L., and K. Fall, "Report from the IAB Workshop on Routing and Addressing", RFC 4984, September 2007.

[O'Dell, 1997] O'Dell, Mike. "GSE: an alternative addressing architecture for IPv6. Draft-ietf-ipngwg-gseaddr-00.txt, 1997.

[Saltzer, 1977] Saltzer, Jerry. "Name Binding in Computer Systems" in Operating Systems: An Advanced Course. Eds.Bayer, R. et al. Springer Verlag, 1977.

[Saltzer, 1982] Saltzer, Jerry. "On the Naming and Binding of Network Destinations" in **Local Computer Networks**, edited by P. Ravasio et al. P North-Holland Publishing Company, pp 311-317, 1982, republished as RFC 1498.