

# Things They Never Taught You About Naming and Addressing

**John Day**

**Feb 2010**

**“Did I ever tell you about Mrs. McCave  
Who had 23 sons and she named them all Dave?  
Well she did and that was not a very smart thing to do  
Because now when she calls, “Yoo-hoo, come into the house, Dave!”  
All 23 of her sons come on the run  
“And now she wishes that she had named them . . .”**

**<there follows a wonderful list of Dr. Seuss names she wishes she'd  
named them and then concludes with this excellent advice.>**

**“But she didn't do it and now it is too late.”**

**- Dr. Seuss  
Too Many Daves**

# Introduction

- This is probably the single most important topic in network architecture
  - And not really covered in university courses.
- And one of the most difficult and subtle topics in all of networking.
  - Sometimes it seems that it is more philosophy than engineering.
  - Although names have figured heavily in philosophy and logic, it is naming as it relates to language. While it is informative, it is mostly not germane.
  - But there is nothing about addressing *per se*.
- The Internet has several addressing related problems today:
  - Address space exhaustion
  - Multihoming is not supported
  - Mobility is cumbersome and complex
  - Router table size is exploding

# Lets Get Back to Fundamentals

- Develop the concepts moving from more fundamental to more specific.
  - Fundamentals of Naming
  - Naming in Computing Systems
  - Naming for Communicating Processes
  - Naming for IPC

# Names and Name Spaces

- A **name space**,  $\mathbf{NS}$ , is a set  $\{\mathbf{N}\}$  of names from which all names for a given collection of objects are taken. A name from a given name space may be bound to one and only one object at a time.
- A **name** is a unique string,  $\mathbf{N}$ , in some alphabet,  $\mathbf{A}$ , that unambiguously denotes some object or denotes a statement in some language,  $\mathbf{L}$ . The statements in  $\mathbf{L}$  are constructed using the alphabet,  $\mathbf{A}$ .
- A function,  $\mathbf{M}_{\mathbf{NS}}$ , which defines the class of objects,  $\mathbf{M}$ , that may be named with elements of  $\mathbf{NS}$ . This is referred to as the **scope** of the name space (see below). This may refer to actual objects or the potential for objects to be created.
- A function,  $\mathbf{F}_{\mathbf{MNS}}$ , that defines the mapping of elements of  $\mathbf{NS}$  to elements of  $\mathbf{M}$ . This function is one-to-one and onto. The result of this function is called a **binding**.

# Operations on Names

- *Assignment*, allocates a name in a name space, essentially marks it in use. *Deassignment*, removes it from use. Assignment makes names available to be bound. This allows certain portions of a name space to be “reserved,” not available for binding.
- *Binding*, maps a name to an object. Once bound, any reference to the name accesses the object. *Unbinding* breaks the binding. Once unbound, any reference will not access any object.
  - An object ceases to exist when the last name referring to it is unbound.
- Saltzer [1977] defines “resolve” as in “resolving a name” as “to locate an object in a particular context, given its name.”
  - An object cannot be identified without locating it nor located without identifying it.

# Types of Names

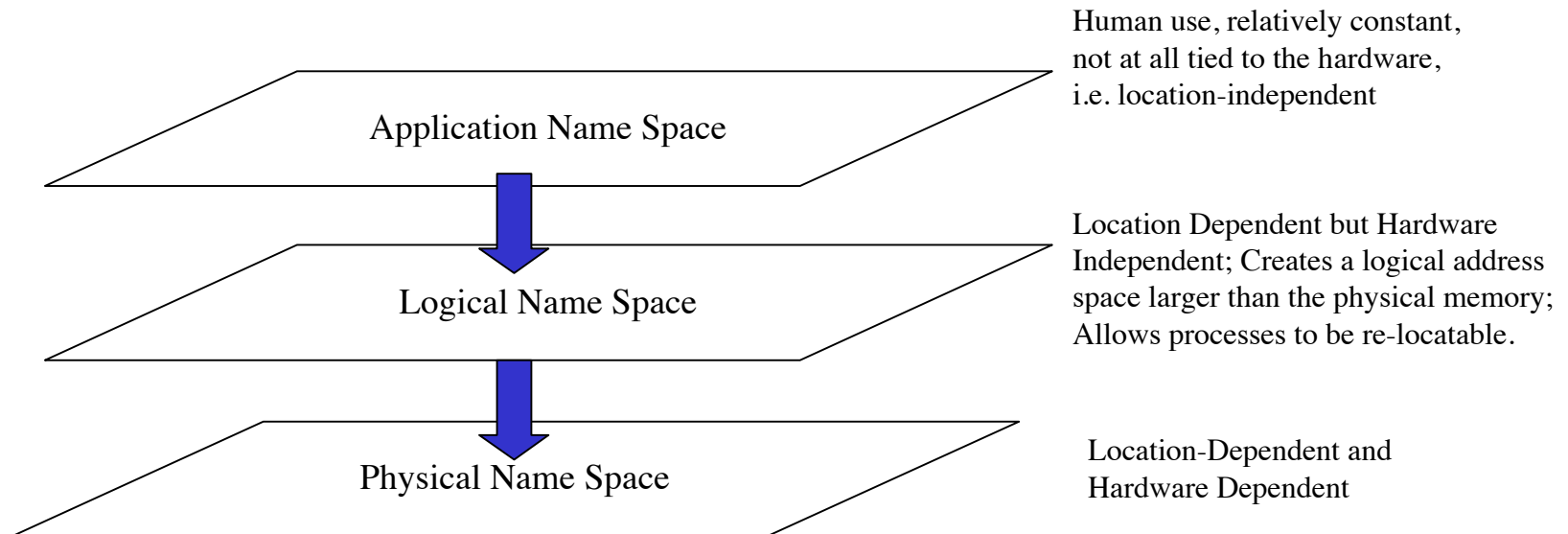
- Objects may be assigned more than one name. These are called *synonyms* or *aliases*.
  - Objects that may have more than one name, the names are unambiguous.
  - Objects that must have only one name, the names are unique.
- Names may also denote sets of names. Associated with the set is a rule that determines which names are returned when the name of the set is resolved. In networking,
  - A rule that returned all members has been called *multicast*.
  - A rule that returned one member has been called *anycast*.
  - We will refer to all forms as whatevercast!
- Synonyms or names of sets may be taken from the same or different name spaces.
  - The name of an object is the name of a set of one element.

# Resolving Names

- There are 2.5 means to resolve names:
  - Exhaustive Search
  - The name provides hints to narrow the search.
- The “half” is indirection:
  - The name or part of the name points to an object that points to a name.
- Hierarchy is the most common form of embedding hints in a name.
  - Hierarchy imposes a topological structure on the name space, which constrains the names that can be assigned to an object.
  - There may be *search rules* for how to utilize the “hints.”
- Every thing up to this point is applicable to all naming in computing systems.

# Address Spaces in Operating Systems

(From my OS Course)



An name space is defined as a set of identifiers with a given scope.

An address space is a location-dependent name space.

In Operating Systems, we have found a need for 3 such *independent* spaces.

Virtually all uses of names in computing are for *locating*.

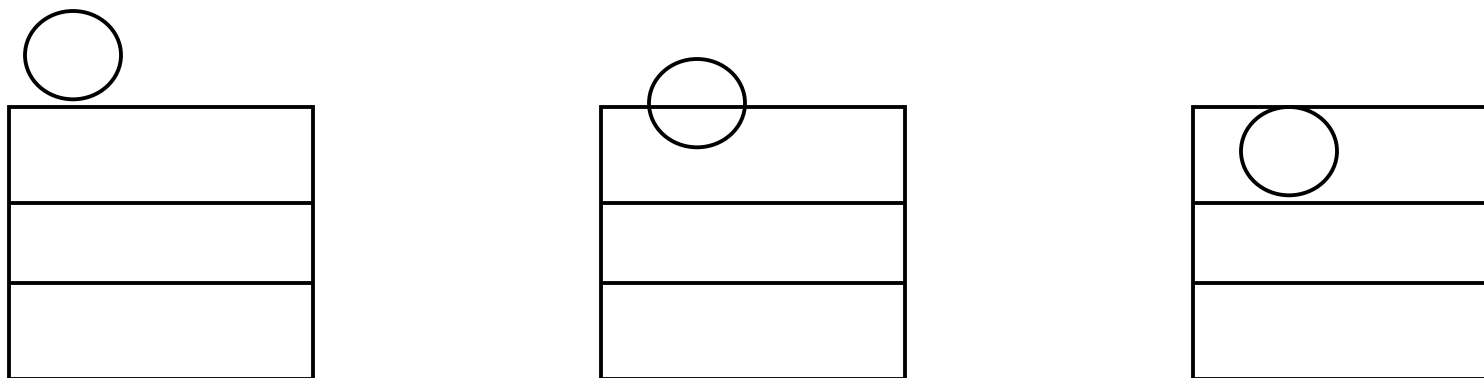
# Communicating Processes

- Application Processes exist to do work. They communicate to do that work. Communicating is not (usually) their primary *raison d'être*.
- So how do applications relate to communication?
  - Communicating applications share state on some things.
  - They can't be unaware of communicating.
  - So what is the nature of the relation between the communication mechanism and the application process?
- Believe it or not, OSI dug into this problem and somehow found the right answer, although it didn't seem so at first.

# Applications and Communication: I

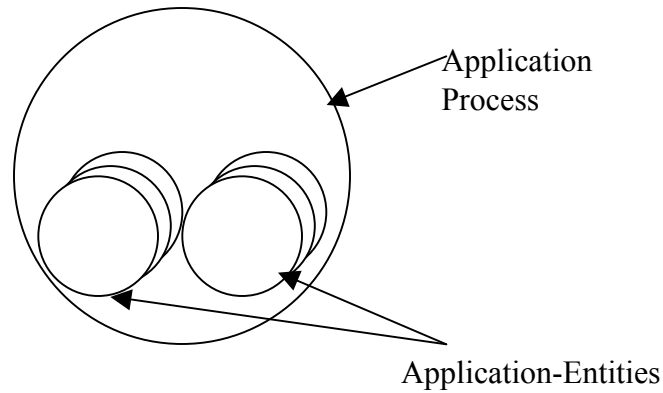
Is the Application in or out of the IPC environment?

- The early ARPANet/Internet didn't worry too much about it. They didn't need to. Only one FTP per system, only one remote login per system, etc.
- By 1985, OSI had tackled the problem, partly due to turf. Was the Application process inside or outside OSI?



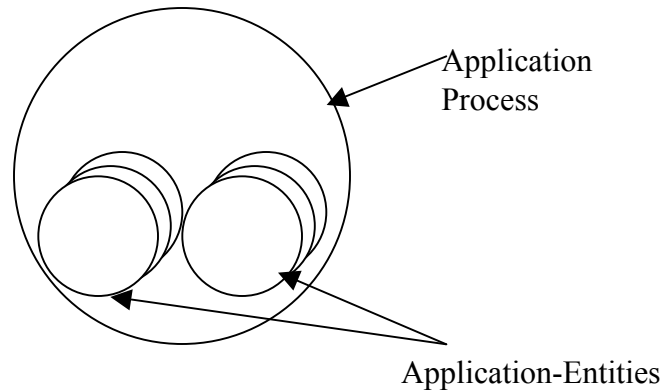
- It wasn't until the web came along that we had an example that in general an application protocol might be part of many applications and an application might have many application protocols.

# Applications and Communication: II



- The Application-Entity (AE) is that part of the application concerned with communication, i.e. shared state with its peer.
- The rest of the Application Process is concerned with the reason for the application in the first place.
- An Application Process may have multiple AEs, they assumed, for different application protocols.
- So what does this mean for naming?
  - Pretty much read it off.

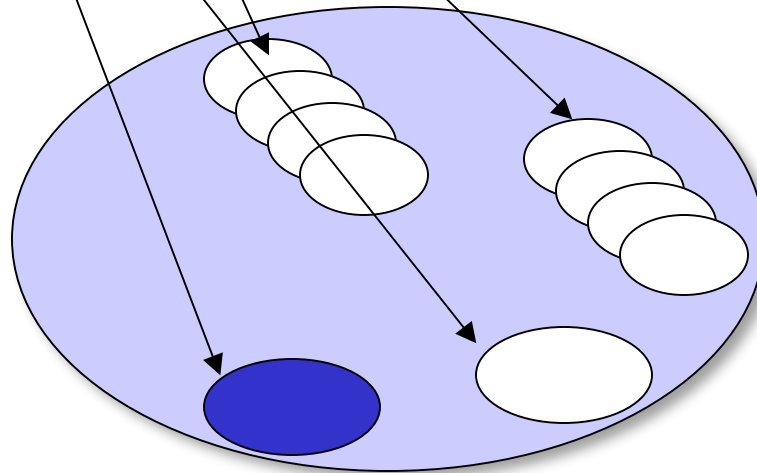
# Application Naming



- Application-process names (APN) are globally unambiguous and location-independent, but system-dependent.
  - They may have synonyms of less scope from the same or different name space.
  - There may be multiple instances of the process in the same system.
    - APN-instance-identifiers are unambiguous within the scope of the Application Process.
- Application-entity-identifiers are unambiguous within the application process.
  - There may be more than one Application-entity (AE) in a process.
    - Unambiguous within the scope of the Application Process.
  - There may be more than one instance of each type of Application-Entity.
    - AE-instance-identifiers are unambiguous within the scope of the AE.
- Distributed Application Name is the name of a set of application processes and system-independent.
- Few applications need all of these but a complete theory requires all of them.

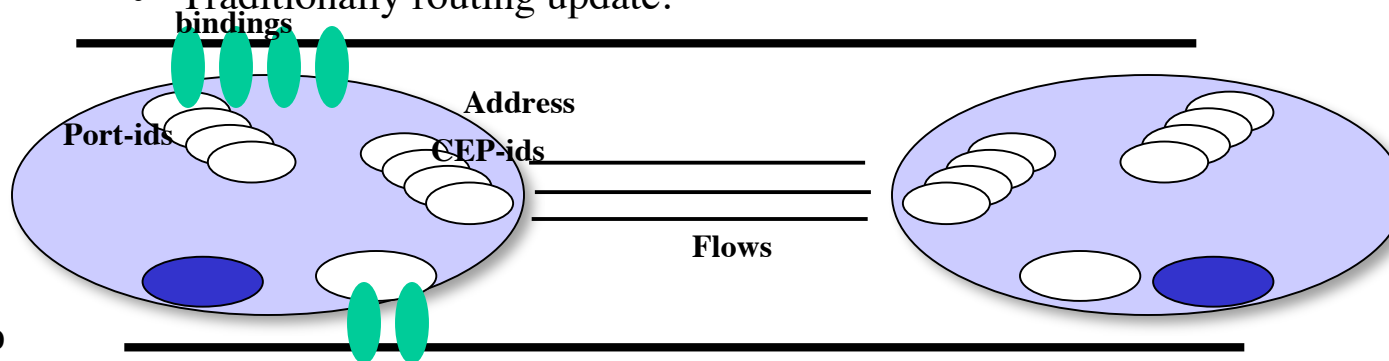
# What about IPC?

- The IPC Model says that an IPC Process is simply an application process dedicated to doing IPC. What does this tell us about that?
  - There are four kinds of Application Entities:
    - An Allocation AE to manage the creation of flows,
    - A Data Transfer AE to control data transfer, and
    - RIB Daemon AE to maintain shared state among the IPC Processes
    - Managing IPC with the (N-1)-DIF/layer

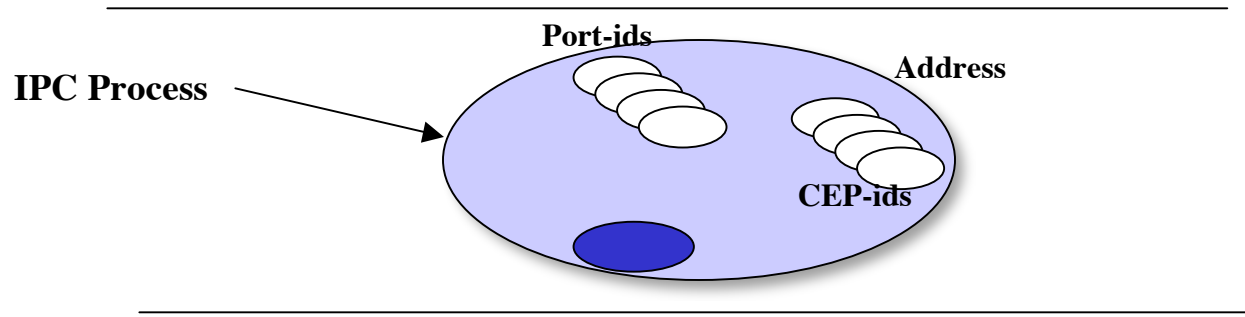


# What about IPC Naming?

- An IPC Process has an Application Process Name, but give it a synonym of scope limited to the layer.
  - Commonly called an *address*
  - In larger layers, the address may be structured to be more effective.
- There are three local identifiers:
  - An Allocation AE Instance Identifier,
    - Commonly called a *port-id*
  - A Data Transfer AE Instance Identifier, and
    - Commonly called a *connection-endpoint-identifier*
  - RIB Daemon AE to maintain shared state among the IPC Processes
    - Traditionally routing update.



# Names for IPC



- An *Address* is a synonym for the IPC Process, with scope restricted to the layer and structured to facilitate use within the layer/DIF.
  - A *port-id* is the handle returned to the calling application to refer to this instance of communication, unique within its AE.
  - A *connection-endpoint-id* (CEP-id) identifies the shared state of one end of a flow/connection, unique within its AE.
- A *connection-id* identifies flows between the same two IPC Processes, formed by concatenating CEP-ids, unique within the pair
- Distributed Application Name is globally unambiguous name for the set of all Application Processes in a Distributed Application, e.g. DIF.

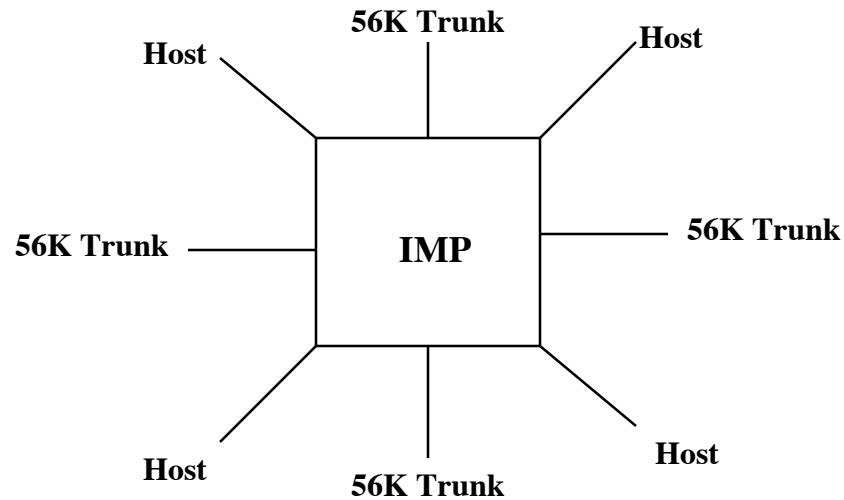
# To Summarize

<b><u>Common Term</u></b>	<b><u>Scope</u></b>	<b><u>Application Term</u></b>
Application Process Name	Global (unambiguous)	Application Process Name
Address	Layer (unambiguous)	Synonym for IPC Process' Application Process Name
Port-id	Allocation AE (unique)	Allocation AE-Instance-Identifier
Connection-endpoint-identifier	Data Transfer AE (unique)	Data Transfer AE Instance-Identifier
Connection-id	Src/Dest Data Transfer AE (unique)	Concatentation of data-transfer-AE-instance-identifiers
DIF Management Updates	IPC Process (unambiguous)	AE-identifier

- All identifiers except address and connection-id are local to the IPC Process.
  - Scope of the address is the Layer, and is not visible outside.
  - Scope of the connection-id is the participants in the connection.
- None has more scope than it needs.

# The Origins of Internet Addressing

## The Root Cause of our Problems



**Each ARPANet IMP (switch) had ports to support a maximum of 4 trunks and 4 hosts.  
Each IMP had a number. The host address (IP address) was the IMP # and the  
Host #, i.e. a port number. Maximum number of hosts was huge: 63.**

**So a host's address was its IMP Port Number.**

# Was There a Reason?

**Sure**

It was easy to build for an *experimental* network of this size.

Was there a lot of thought given to how addressing should work?

Not really.

We were doing good to do this much!

There were many much bigger problems to overcome:

Like just moving data

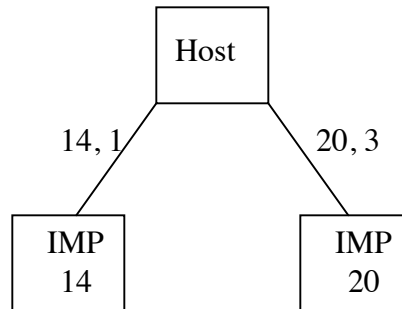
And addressing is a hard problem.

# Did it take long to realize there was a problem?

**Nope.**

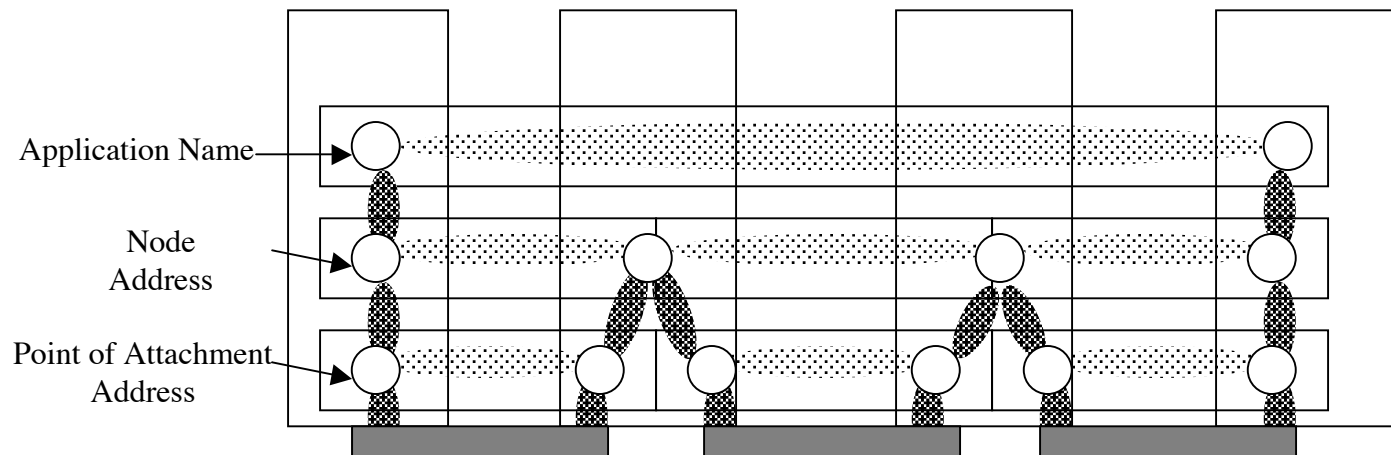
**First time (~ 1972) one of the Air Force bases took us at our word that the network was suppose to be survivable and asked for links to two different IMPs to connect its host to the Network.**

**Naming the hosts by the names of their interfaces meant that the two connections looked like two hosts to the Net.  
Still does.**



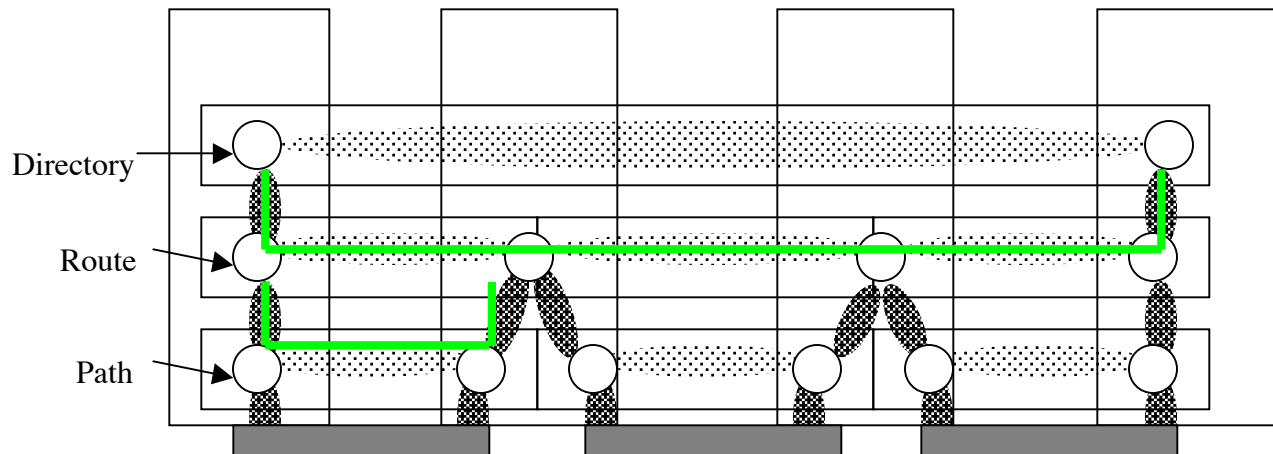
# Saltzer's Model

- When do we need addresses?
  - When the “wire” is not point-to-point, when the other end is ambiguous.
- Application names map to node addresses.
- Node addresses map to points of attachment addresses.
  - Concepts of node and point of attachment are relative.
- Routes are sequences of points of attachments.
  - Just as in an operating system.
  - But networks are always more general than operating systems.



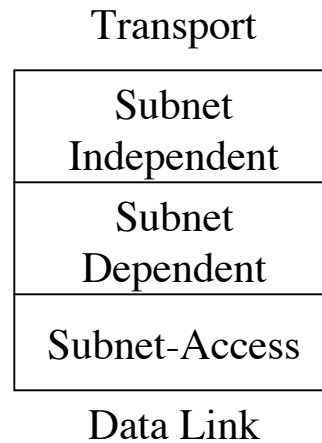
# Generalizing Saltzer to Networks

- Directory maintains the mapping between Application-Names and the node addresses of all Applications reachable without an application relay.
- Routes are sequences of node addresses used to compute the next hop.
- Node to point of attachment mapping for all nearest neighbors to choose path to next hop. (Saltzer missed this because they hadn't occurred yet.)
- This last mapping and the Directory are the same:
  - Mapping of a name in the layer above to a name in the layer below of all nearest neighbors.



# OSI Was Working with Greater Diversity

(for better or worse)

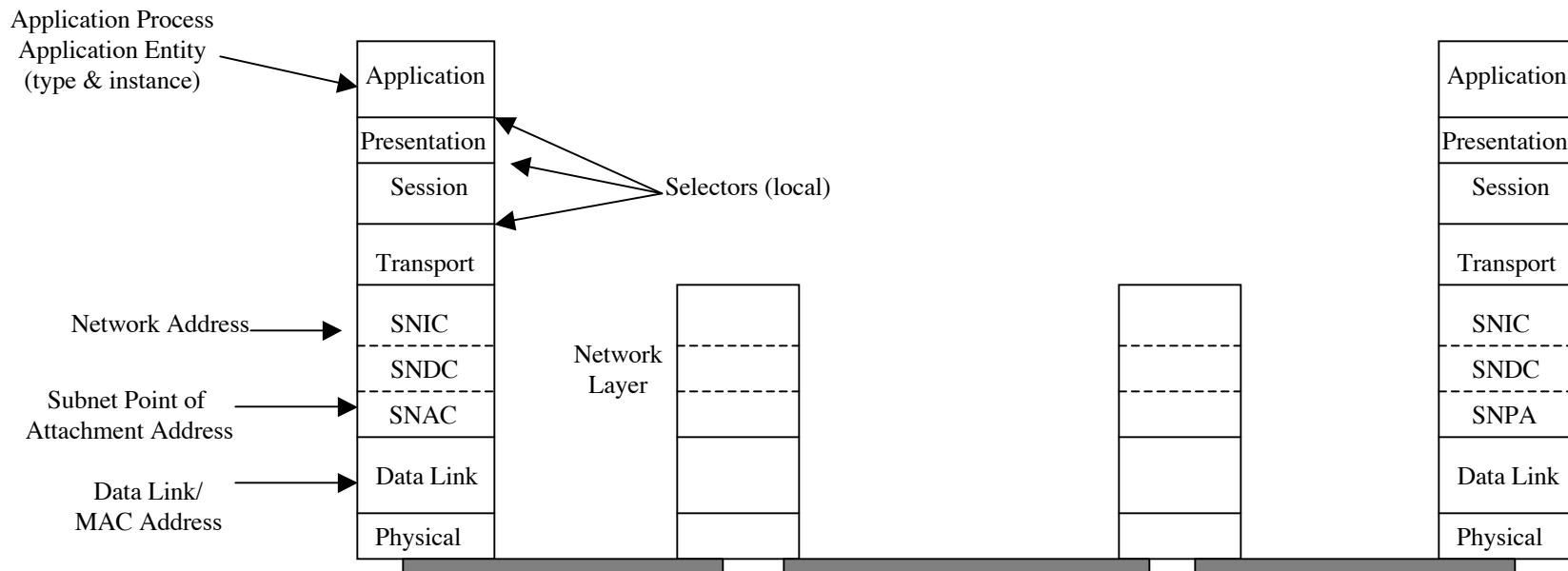


- OSI Determined that the Network Layer had 3 parts.
- Point of Attachment (interface) addresses were either Data-Link addresses (Ethernet) or Subnet addresses (X.25, IP).
- Subnet Independent (node) addresses at the top of the Network Layer (CLNP)

# Applying Results to Real Architectures: OSI

(This *was* an Internet Architecture)

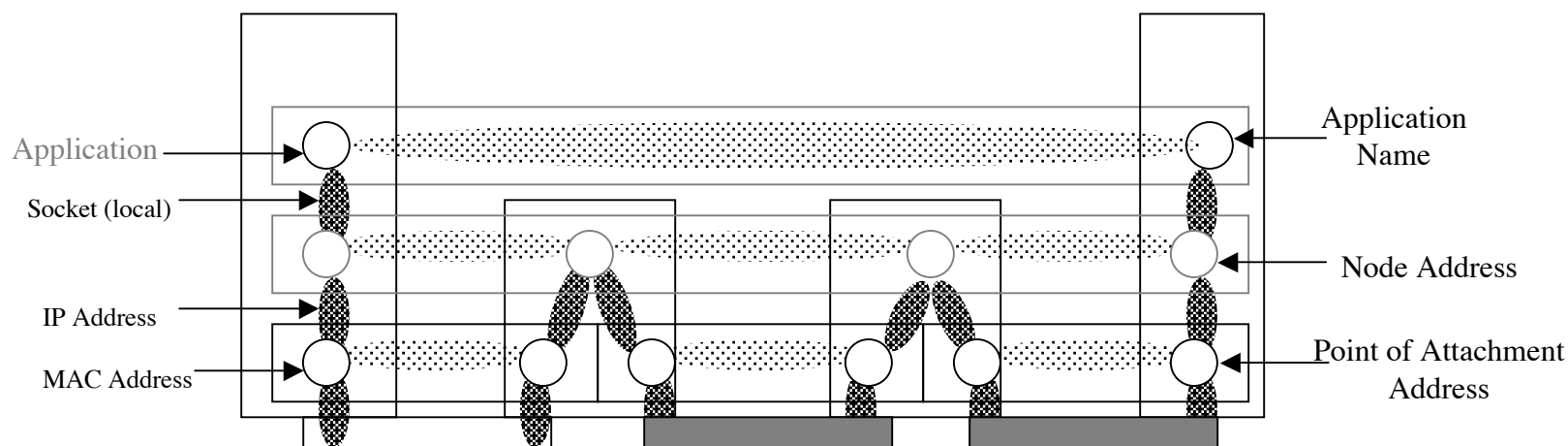
- For better or worse, it is a complete addressing architecture.
  - If anything, too many addresses.
  - But its Internet Protocol *is* an internet protocol.
- The good news: Session and Presentation addresses should be null.
- Why Subnetwork attachment addresses *and* Data Link Addresses?
  - Politics. Equating X.25 and Ethernet in the minds of some was an anathema.



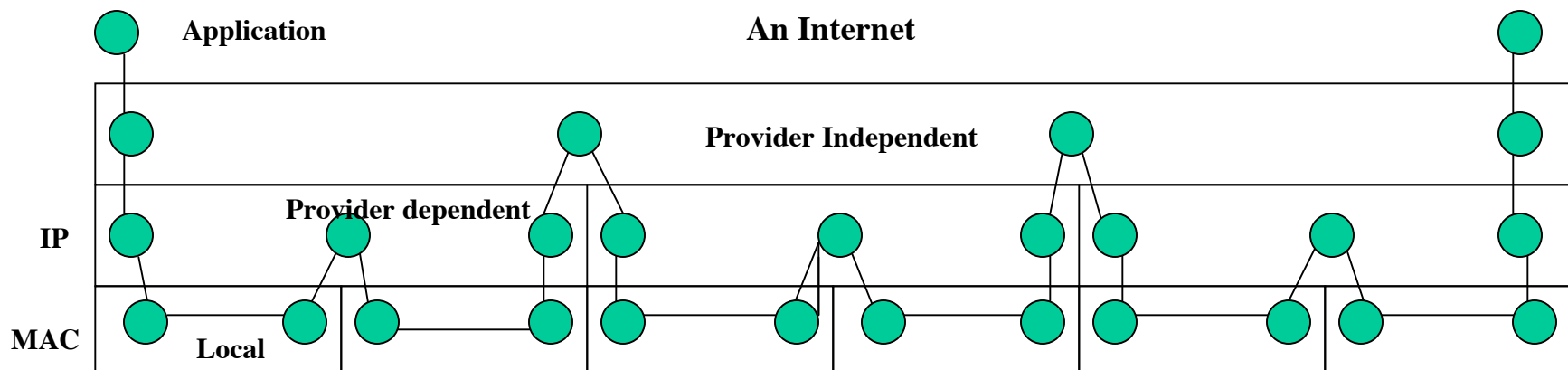
SNIC - Subnet Independent Convergence  
 SNDC - Subnet Dependent Convergence 23  
 SNAC - Subnet Access © John Day, 2010  
 All Rights Reserved

# Applying Results to Real Architectures: The Internet

- The most striking feature is that half of the addressing architecture is missing.
  - No wonder there are addressing problems.
  - The only identifier we have for anything is the IP address.
- There are no node addresses and no application names.
  - And the point of attachment is named twice!
  - *Internet* Protocol names a subnet point of attachment.
    - Why does it route on the layer below's address?
  - Domain Names are synonyms for IP addresses. URLs are pathnames through the stack and location dependent.



# So What Should it Look Like?



- At the very least, a provider-independent Internet Layer over a provider dependent Network Layer, over a Data Link Layer.
  - like OSI.
    - Not Just a Network Layer over a Data Link Layer.
      - like the Internet
- Routing should be done with addresses in the layer. . . Well, duh!
  - A layer's addresses are always node addresses.
  - Points of attachment are addresses in the layer below.

# The First Great Internet Addressing Crisis

- In 1992, we have the first addressing crisis.
  - IPv4 addresses are getting scarce
    - Router Table Size is increasing exponentially.
- The IAB convenes the ROAD process
  - Recommends CLNP as IPv7
    - Basically IP with variable length aggregateable addresses.
    - CLNP names the node. Hence, fixes the multihoming problem.
- The IETF goes berserk!
  - No OSI, no way, no how!
    - A model? We don't need no stinking model. *We've got*
    - Rough Consensus and Running Code!

# The IPng Process

- The Rules were:
  - Fixed Length Address
  - Continue to Name the Interface.
  - At least 20 octets of address
  - Open Standard
- Violá! IPv6!
  - or anything but CLNP.
- Still no solution to multihoming
  - Problem is now 20 years old

# All is Not Well

- Several Months later, light dawns slightly!
  - Name the Interface, but route aggregation is a must
  - Implies provider based assignment.
  - Means change providers, must renumber. WHAT!?
    - Kind of shot yourself in the foot, eh?
- Transition is dual stack with a NAT
  - Once you have a NAT, you don't need v6 . . . oops.
    - How many feet do you have?
- Finally around 2000, need to deal with multihoming, but
  - given negative reaction to naming the node need a workaround
  - The problem is that we have been overloading the semantics of the IP address with location and identifier information.
  - We need to split them. Loc/id split is the answer.
    - A locator we can route on and a flat endpoint id (EID)
      - Psssst! Can't identify something without locating it and vice versa
      - Got another foot?
- Okay, IPv6 is the future!

# Trouble is Not Much Interest in v6


- It offers no benefit to those who pay for adopting it
  - They just don't know they want it.
- For the next decade, there is the hype:
  - Better security, better QoS, better mobility
    - “A desert topping and a floor wax!” - Mike O'Dell
      - In fact, all of these are the same as IPv4 . . . no different
- “IPv6 has all the benefit of a minor technology change with all the disruption of a major fork-lift upgrade.” - Geoff Huston, 2008.
  - Just switch to v6 and all will be well.
- By 2000, aware the architecture is running out of steam
  - NEWARCH, Clean Slate, FIND and GENI are hot!
  - Starts to be a lot of talk about loc/id split.
  - This is clearly the answer . . . . (really?).

# Houston, We Have a Problem

- October 2006, major presentation at IEPG.
  - Router table size is on the increase, due to multihoming.
  - Moore's Law won't bail us out this time.
    - This is a big time crisis. We are in big trouble.
- If not fixed, it is the end of the Internet as we know it.
  - Net will fragment. Costs in the core will skyrocket.
  - NetworkWorld sits on the story for a year.
  - Tons of papers written on loc/id split!
- Finally, Cisco and others start proposing solutions.
  - Mostly requiring patches involving NATs and a bevy of new protocols.
    - This makes everyone nervous, but what else to do?

# But We Do Multihoming!

(not really)

- We kludge it.
- Because we route on the interface, this forces route aggregation to be provider-based.
  - Addresses with a common prefix go to the same provider then we can store a single route (to the provider) rather than a route for every address on that provider.
- To do multihoming, must assign provider-independent addresses or new AS numbers (same thing).
  - Can't be aggregated  Router table size increases
  - Remember what our problem is? Router table size is increasing

# So Why Wasn't It Fixed?

- Odd that a DoD network touted to survive nuclear attack didn't support redundant links. Lots of “good reasons:
  - Not that many hosts need to be multi-homed.
    - Not then, but the ones that did were the ones everyone wanted to get to.
  - Not everyone should have to bear the cost for a few.
    - Classic committee politics: Put a condition on the solution that guarantees any proposal will be rejected (asymmetry in this case)
    - Also assumes there is a cost.
  - Multihoming will be to different providers, so no point.
    - Assumption is wrong and even if right assumes a static network.
  - Remember, we tried to fix it but it was rejected by the IETF.

# But By This Time

- Cisco and others start proposing solutions to the multihoming problem.
  - Mostly requiring patches involving NATs and a bevy of new protocols.
  - In particular, LISP or Loc/ID Split Protocol
    - This makes everyone nervous, but what else to do?
- This is a crisis!
  - Best way known to stampede people to your view.

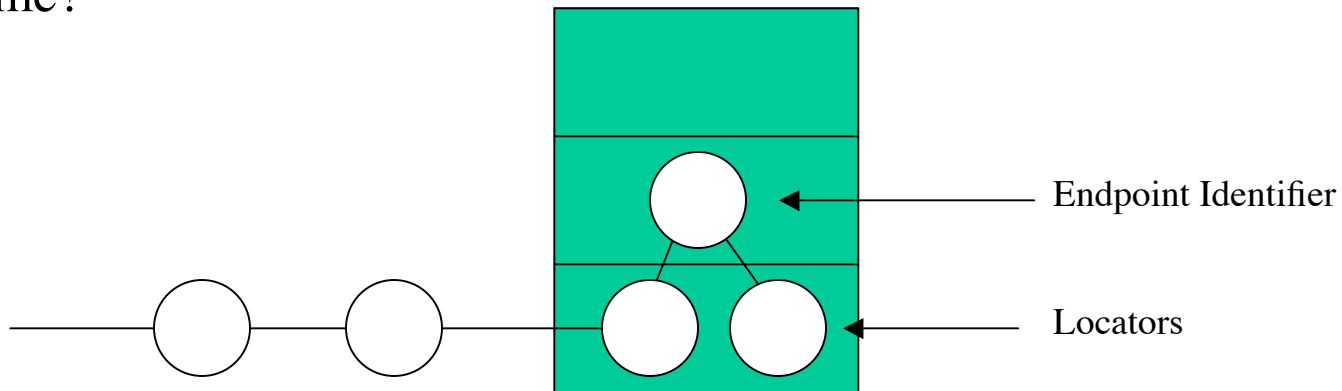
# November 2008

## Houston, We have a Bigger Problem

- Dave Meyer calls, ‘I have an “architectural issue” to discuss.’
  - He has come across two problems in implementing LISP.
  - Both require doing path discovery.
  - Path discovery doesn’t scale. LISP won’t scale. QED
  - He suspects that any loc/id approach will have the same problem.
    - draft-meyer-loc-id-implications-01.txt
    - In case you didn’t notice, we just went to DefCon1
- Dave: Why hasn’t anyone noticed this in the last 15 years?

# Dave is Right

- All proposals based on loc/id split have the same flaw.
- Once put in the context of the PNA theory, it is obvious.
- Let us look at it very carefully. What do the locator and the identifier name?



**The Locator Locates the Wrong Thing!**  
The locator is part of the path, not the final destination.  
No wonder Dave ran into *path* discovery issues.

Apply the e2e principle!

## Solve the Problem in the Hosts

- There are poor misguided souls claiming this.
  - Mostly done by changing the definition of multihoming.
  - Ignore that it might take seconds if not minutes to do the failover.
- Remember the fundamental problem is that the network doesn't know that two paths go to the same place.
  - This is a problem of *delivering*, not sending.
  - There is no host-based solution.
- There is no solution as long as one routes only on the interface address. Which means . . .

# If Loc/ID Doesn't Scale

- Then there is no solution involving routing on the interface that scales.
- In other words,

## IP is Fundamentally Flawed

(v4 or v6)

- But we saw that coming a long time ago.

# So What Is the IETF Plan B?

- Plan A
  - What are the major Universities working on?
  - Plan A
  - What are the vendors working on?
  - Plan A . . . . Still?
- 
- Talk about scary, there was a summit in the Spring of 2009 to discuss the addressing crisis
  - The agenda: to determine what form of loc/id split to use!
    - It is important to get those deck chairs lined up very neatly.

# What Do the Vendors Say?

- Yes, there *may be* some scaling issues with loc/id split.
- But we recommend using it, they are very unlikely to ever be a problem.

## A Black Swan?

- You mean unlikely as in “it is very unlikely a lot of people will default on home mortgages”?
  - (we know where that went)
- Don't worry! It won't happen very often, only in a crisis.

# But Everything Works Fine!

- Yes and it will for a few more years.
- When you notice it, it means we are already over the cliff.  
Vince Fuller, Cisco.

“Has he vertigo?  
No, only about  
ten stories more!”  
- Ogden Nash

# Wanna Hear the Real P\*sser?

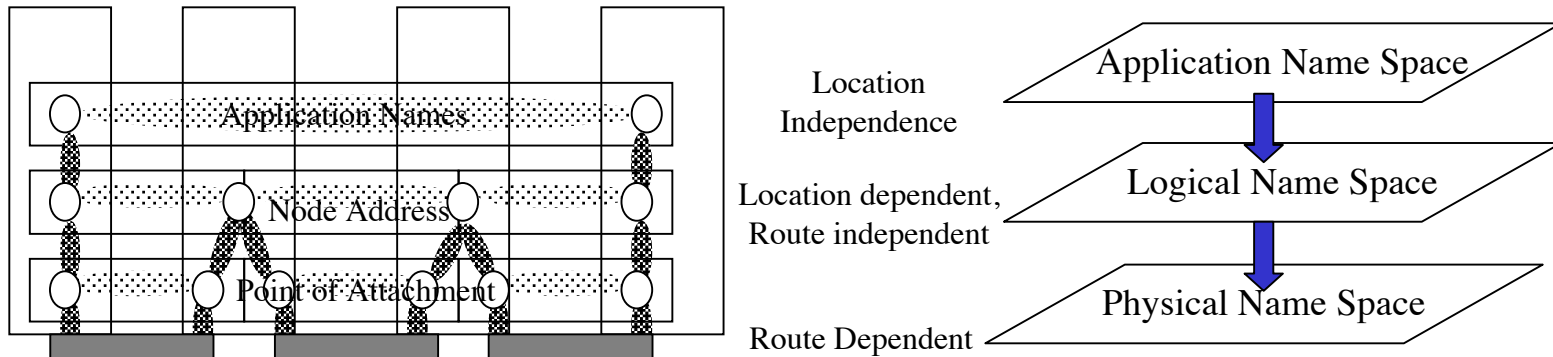
- We had the right answer in 1992.
  - Actually known the answer since 1972.
    - It was rejected by the IETF.
    - Why? Hubris and the Failure of University Education
- And to add insult to injury, it was **DEPLOYED** in the routers.
  - We could have spent the last 15 years working on transition
  - Rather than 100s of millions on a small incremental step that provides no benefit to your bottom line and is fundamentally flawed.
  - You just can't make stuff like this up!

The Problem was Never Separating Locator from Identifier.

It was and is at least

## Separating Logical Location from Physical Location

- It is impossible to locate something without also identifying it.
- This pseudo-problem arises from not having a complete address architecture.
  - And creates enough epicycles upon epicycles to make Clavius proud
- And from not being precise in your terms, Or as Wittgenstein said,
  - “7 That of which we can not speak, we must pass over in silence.”
- But we will give O’Dell the last word:
  - When all you have is a hammer, everything looks like your thumb



# Principles of Addressing: I

- The Scope of Layers tends to increase the higher in the Architecture
- An address is an synonym for the IPC process with less scope and structured to be useful within the layer.
  - An address is simply a short identifier used only within the layer.
    - For large layers, structuring an address facilitates its use within the layer.
      - Allows a structure more effective than the application name structure.
- An Address should only be unambiguous within the scope of a layer.
  - And no more, otherwise this ends up overloading the semantics of the address and creates problems
    - MAC addresses are 3 times longer than they need to be.
- Routes are sequences of (node) addresses.
  - Source routing is a “male thing,” not willing to stop and ask directions.
- The relation of node and point of attachment is relative and hence irrelevant.
  - A layer routes on its address. A node address is this layer’s address, the point of attachment address is lower layer’s node address and not visible to this layer anyway.

# Principles of Addressing: II

- Since the address is structured to facilitate its use within the layer, it must be assigned by the layer.
  - Only the layer knows *how* to make the synonym useful.
    - Any addresses assigned by elsewhere are not addresses.
- An address names the locus of processing that removes the header containing the address.
  - Naming the host is irrelevant.
- An address should not be constructed by concatenating an identifier in this layer with one from the lower layer, i.e. don't put embed MAC addresses in IP addresses.
  - Why? Because it makes a *pathname*, route-dependent!
  - Addresses in adjacent layers should be completely independent.
  - Layers with hierarchical addresses are used to organize addresses in this layer, not the layer below.
  - Each layer is a level of indirection.
- Ignoring these will lead to complexity and cumbersome solutions, but then you already knew that from experience, right!?

# Implications of the Principles

- Multihoming - is accommodated by having distinct logical and physical address spaces.
  - However, there must be a *topological* relation between the two address spaces.
- Mobility - is merely dynamic multi-homing with expected “failures”
  - New attachment points appear to a system; addresses are assigned to the new attachment point and it is bound to an appropriate node address. In some cases, old attachment points disappear. Nothing special required.
- NATs - are either everywhere or nowhere.
  - NATs are only an issue in incomplete architectures.
    - Or to paraphrase Bucky Fuller, NATs only break broken architectures.
- Changing addresses is simply adding another synonym and letting the old one die.
  - On-going connections are never disturbed.
- A Global Address is Unnecessary, probably never need more than 32 or 48 bits.
- Even stranger, while useful there is no requirement for a global application name space either.
  - This one has some scary implications.

# Conclusions

- Most of the problems in the Internet today: scaling, multihoming, mobility, distributed applications, etc. are rooted in not having a complete addressing architecture.
  - The Internet is more like DOS, than UNIX.
- The Internet has a quarter of an addressing architecture.
  - Unlike an OS, there is no indirection in its addressing architecture.
    - Imagine trying to do re-locatable code without virtual addresses!
  - More identifiers without decoupling the name spaces solves nothing.
  - Loc/id split was just post IPng trauma.
    - Such intense reaction to naming the node, that there had to be a workaround.
    - There isn't. (Just like there isn't in operating systems)
- Now Mrs McCave wishes she had fixed that problem in the 70s.
- But she didn't do it and now it is too late! . . . for IP.