# How in the Heck Do You Lose a Layer!? (invited paper)

John Day Boston University
Computer Science, Boston University, Massachusetts, USA
Email: day@bu.edu

***Abstract*—Recently, Alex McKenzie published an anecdote in the IEEE Annals of the History of Computing on the creation of INWG 96, a proposal by IFIP WG6.1 for an international transport protocol. McKenzie concentrates on the differences between the proposals that lead to INWG 96. However, it is the similarities that are much more interesting. This has lead to some rather surprising insights into not only the subsequent course of events, but also the origins of many current problems, and where the solutions must be found. The results are more than a little surprising.**

## I. Introduction

From the beginning of networking, operating systems have been our guide. "Networking is Interprocess Communication"[1] could be our mantra. This was our model when doing the early network and early applications: Telnet, FTP, and RJE[2]. It dominated much of the thinking that surrounded the ARAPNET, CYCLADES, XNS and the landmark text, "Distributed Systems Architecture and Implementation" edited by Butler Lampson [2].

Although we were using the concept of layers from operating systems [3], it was recognized that layers in networks were not the same as in operating systems. Layers in OSs were (and are) a convenience, one possible design choice. In networks, because there is distributed shared state of different scopes, layers are a necessity. This property of distributed shared state of different scopes is the primary characteristic that requires layering in networks.[3] Shared state of different scopes is a necessary and sufficient condition for a layer. A collection of seemingly related functions organized into a layer may be a sufficient condition, but not a necessary condition. Organizing functions within a given scope into additional layers is possible but care must be taken to ensure that the functions in different layers are truly independent and that the necessary invariant properties are maintained. Otherwise, there will be problems.

Because the traditional PTT or datacomm model of "beads-on-a-string" that had been the existing model cannot represent layers of different scope, it would not

[1]Expressed in print by Bob Metcalfe [1972].

[2]Almost to prove the point and contrary to what many textbooks say, Telnet is not, nor never was, a remote login protocol. It was a terminal device driver protocol.

[3]It seems that most people this missed point. The textbook authors became fixated on what went in what layers and never mention scope. It now seems that most professors missed it as well. Admittedly, there is not much to say about layers being loci of distributed state of different scopes beyond what has just been said. But it is critically important and apparently not obvious.

be viable for networking.[4] A new theory was required. The only question was what went in which layers? There was still Dykstra's idea that functions didn't repeat in the layers, but that was continually being challenged by real requirements that said otherwise. An important step in working out that next step was INWG 96. As we have found, Dykstra was correct: Functions should not repeat in layers of the same scope. (THE was too resource constrained to observe more than one scope.)

## II. INWG 96 and its Implications

Recently, Alex McKenzie wrote an excellent Anecdote for IEEE Annals of Computing History on the creation of INWG 96 [4]. INWG 96 was a proposed Internetwork Transport Protocol developed by the International Network Working Group, IFIP WG6.1 in 1976. (INWG was initiated in 1972 by the research network developers e.g. ARPANET, NPL, CYCLADES, EIN and others, to begin a standards process for this new approach to networking.) As Alex relates, IWNG 96 [5] was a synthesis based on INWG 31 (TCP before IP was split into a separate protocol) and INWG 64 (a derivative of CYCLADES TS). At the time, the primary differences were in how fragmentation was done at Internet Gateways and whether the data would be a stream or what was called the "letter" concept.

Briefly, INWG 31 used byte sequence numbers to label fragments, while INWG 64 had a packet sequence number, more fragment bit and an offset. The letter concept concerned the structure of the data. INWG 31 provided a byte stream and required the application to delimit the units of data significant to it. INWG 64

[4]Current proposals that do not accommodate scope or speak of control and data planes are clearly still working in the old beads-on-a-string model. (The concept of "planes" originates with ISDN, a very beads-on-a-string technology.)

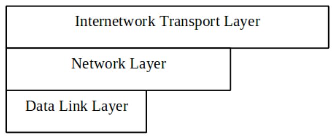

Fig. 1. Three layers of increasing scope each with their own addressing.

maintained the integrity of data passed to it. In other words, if an application passed a certain amount of data to the Transport Protocol that was fragmented or combined during transfer, the Transport Protocol would deliver that amount of data as a unit to the destination. This is what OSI would later call an SDU or Service-Data-Unit. These were not very important differences to be arguing about. INWG approved INWG 96 in 1976. NPL, EIN, and CYCLADES immediately adopted it, but DARPA didn't. INWG 96 served as the basis for OSI TP4.

However, what is more interesting about the INWG work is not the differences in the three protocols, but the similarity among the three protocols: All three transport protocols carried internetwork addresses. This means the INWG architecture was as shown in Figure 1.

If this doesn't hit you like a ton of bricks, you haven't been paying attention. This is not the architecture we currently have.

What we see in the INWG model is that internetwork addresses named hosts and internetwork gateways, a term for a router that was between two networks. What we might call a border router today. These addresses were seen as global in scope. Multiple Network Layers, one for each network, comprising an internet. Each Network Layer has its own protocols and addresses for access within the local network, e.g. an interface address and for its internal routers and switches. These addresses

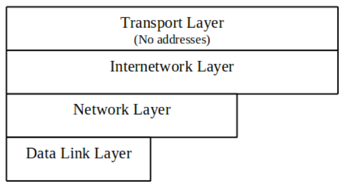

Fig. 3. When IP was Split from TCP, the result was 2 Layers of the same Scope.

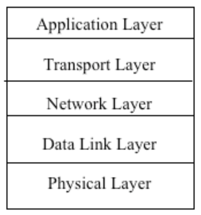

Fig. 4. The Internet Architecture as depicted in Kurose and Ross, Computer Networks, 3rd Edition.

would have scope limited to a particular network. The Data Link Layer is either a point-to-point link where no addresses are required or is a multi-access technology, such as Ethernet or wireless, that requires addresses on the media. Any Data Link Layer addresses (known these days as MAC addresses) would have scope on that segment or wireless network. To simplify, one would have had a picture like this:

Host and Internet Gateways had (at least one) network address for each network they connected to (an interface address), but one internet address. Network routing (intra-domain) occurred in the Network Layer and internet routing (inter-domain) in the Internet Layer. At the time, the ARPANET, NPLNET, CYCLADES, etc. were networks with an internetworking protocol (INWG 31, 64, or 96) over them.

## III. Things Take an Odd Turn with IP

When IP was separated from TCP, the addresses and fragmentation/reassembly were moved to IP in a different layer. This should have changed the picture only slightly to the following:

Assuming that splitting IP from TCP didn't violate any dependencies among the functions, this should not have changed anything about Figure 2, but it did. Several things happened in quick succession: 1) it turns out there are dependencies between the functions now split into two layers, 2) IP addresses name the interface not hosts and internet gateways, and 3) the architecture becomes the one now seen in most textbooks:

Somewhere they lost a layer!

The question is which one? It is not clear how or why this happened. It isn't clear whether we have network addresses in an Internet Layer or an Internet with only a Network Layer. What is clear is that something is very wrong.

By the early 80s, there was little overlap among the technical people doing TCP/IP and the original ARPANET crew.[5] When Tinker AFB joined the ARPANET in 1972 with redundant IMP connections, it had exposed the addressing problems introduced by multihoming, i.e. that routing should be to the node, not the interface. The vast majority of ARPANET/Internet hosts in the late 70s and early 80s had single connections to the 'Net. It is unclear if the Internet developers had realized the implications or even knew about the Tinker AFB issue. Clearly, the INWG group at least understood the problem and the solution even if they were not aware of that particular event. All of the other architectures (XNS, CYCLADES, EIN, NPL, DEC, OSI) took a network approach rather than a datacomm approach.

[5]It should be noted that unlike today, very little of what was being talked about was being published either in the journal literature or the informal INWG or RFC papers. There weren't that many people involved. Producing papers was much more time consuming.

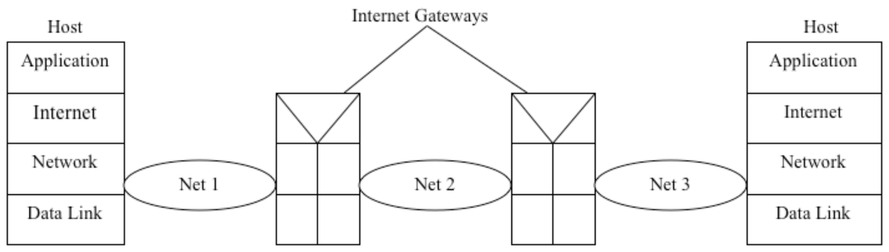

Fig. 2. The INWG view of Internet Architecture.

TCP/IP operated over the ARPANET NCP (Host-to-Host Protocol). In the INWG model, NCP/IMP Subnet was the Network Layer and IP was the Internet Layer. Since most hosts had single connections to the "net" and missing the concept of scope of a layer, it was easy to confuse that the NCP IMP address and the IP address were the same thing. But they weren't. So IP addresses were really Network Addresses (?) at best and Data Link addresses most of the time.[6] This introduces a major flaw in the architecture of the current Internet, which is compounded by the fact that early IP address assignments were not location-dependent.[7] Assignment within a network might have been location-dependent, although indications are that most network administrators followed IANA's lead and normally did not make assignments location-dependent. Strictly speaking, IP addresses before CIDR were not Internet addresses and actually not addresses at all.[8]

In contrast consider what OSI did with the same data. OSI had a somewhat different environment to deal with than the Internet. Long before the Network Layer Group (SC6/WG2) could turn its attention to what the structure might be (and far too early), OSI had adopted the seven layer model. All of this in the midst of the PTT pressure to kill off connectionless. Since most of the SC6 connectionless proponents were also working in the IETF, there was a strong inclination to follow the same approach the Internet had, i.e. that all routing related functions went in the Network Layer. There was support for this from the PTT faction as well, who felt routing was their domain and had no use for a transport layer. Separating them so they didn't have to use transport suited them just fine. OSI also had to assume that there would (and in fact already did) consist of multiple carriers using different technologies. When OSI turned to the problem, it was worked out in a hotly contested environment to figure out not only what was the structure of the Network Layer, but also what would be the role of connection/connectionless. One of the concerns at the time was the possibility of traffic traversing a network of less quality than the networks on either side. (There is a similar figure to Figures 5 and 6 in ISO 7498-1, the OSI Reference Model.)

Since there might be networks of different technologies, e.g. X.25, ATM, MPLS, etc., some sort of error and flow control protocol, something similar to HDLC in the data link layer or the protocols proposed for the Transport Layer to enhance intervening networks

[6]Naming the network interface is equivalent to naming the same thing a MAC address names.

[7]It was not possible to tell by inspecting two addresses if they were "near" other for some concept of "near", i.e. aggregatable for routing purposes.

[8]This is pre-CIDR. But then MAC addresses aren't addresses either.

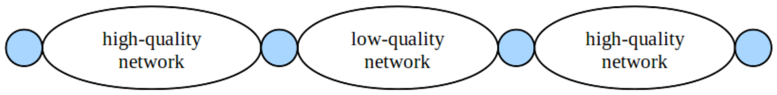

Fig. 5. A low-quality network on the path between two higher-quality networks.

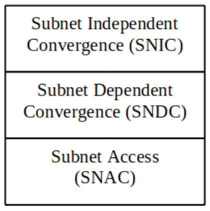
Subnet Independent Convergence (SNIC)
Subnet Dependent Convergence (SNDC)
Subnet Access (SNAC)

Fig. 7. OSI divided the Network Layer into 3 sublayers. To translate the cryptic standardese: SNIC, is the Internet Layer; SNDC is the protocol for enhancing a low quality network if needed; and SNAC is network protocol if there is one, e.g. X.25, ATM, MPLS etc. and if not, it might be a point to point wire or Ethernet.

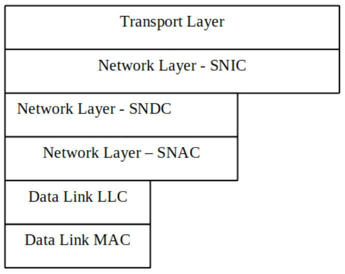

Fig. 8. The OSI view of Figure 1 and Figure 3.

that could be used to improve the quality or provide congestion control within the network.

Since there would be technology dependent subnetworks with their own protocols, there would need to be a technology independent protocol or internet protocol. At this point they had to work within the 7-layer structure, so the Network Layer ended up with 3 sublayers or roles, (some of which might not appear in all situations) [**?**]. When all was said and done they came up with a structure that looked like (Figure 7).

With a Transport Layer, Figure 7 is basically the same as Figures 2 and 3, and only differs from Figure 1 in where the layer boundaries are drawn.

So OSI was an internet architecture, and the Internet was a network architecture. The Internet is not an internet. OSI introduced the idea of intra-domain and inter-domain routing, it is not clear that everyone saw intra-domain being related to SNDC/SNAC, inter-domain related to SNIC/Transport. Mostly, SNAC was seen as X.25 and therefore belonging to the PTTs; and SNIC was CLNP and therefore for the connectionless advocates. However, there was discussion in OSI of both reversing the roles of X.25 and CLNP or even using the same protocols for the different roles, i.e. CLNP as both SNIC and SNAC.[9] This would have lead to multiple SNAC sublayers, as in Figure 2, of less scope and possibly under different auspices (with Network routing) and a SNIC sublayer of greater scope (with Internet routing). Precisely what the INWG model had considered. Probably the largest factor in obscuring the structure was the intensity of the political debate and the need to “get something out”. This precluded the luxury of stepping back to consider alternative structures in sufficient detail to see their advantages.

This does not mean that we should be doing OSI. Good grief, no. This only implies that the data OSI had to work with brought them to the same structure INWG had

[9]This was not explicitly called out in the Internal Organization of the Network Layer, but then it would not have been the appropriate place for discussing specific solutions.

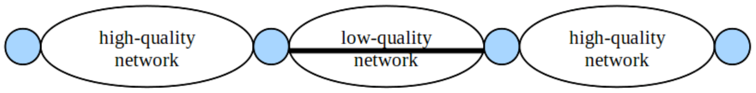

Fig. 6. An enhanced low-quality network between two higher-quality networks.

come to.[10] OSI would have brought along a different can of worms. OSI was the state of understanding in the early 80s. We have learned a lot more since.[11] There was much unnecessary complexity in OSI and recent insights allow considerable simplification over even current practice. OSI also split the addresses from the error and flow control protocol. This creates other problems.

But the Internet's course is definitely curious. Everyone else came up with an internet architecture for an internet, except them. These were the people who were continually stressing that they were building an Internet. Even more ironic is that the Internet ceased to be an Internet on the day most people would mark as the birth of the Internet, i.e. on the flag day January 1, 1983 when NCP was turned off and it became one large network. It was well understood at the time that two levels of addressing were required. This had been realized in the ARPANET when the first host with redundant network connections was deployed in 1972. The INWG structure provided the perfect solution. It is clear why the Internet kept the name, but less clear why they dropped the Network Layer. Or perhaps more precisely, why they renamed the Network Layer, the Internet Layer. Did they think, just calling it something different made it different?

It is likely that, because the early Internet developers did not recognize shared state of different scope as the major characteristic of layers, that networks were not exactly like operating systems, for them, it was purely a case of modularity and not repeating functions (Dijkstra's characterization).[12] With such a view, using the same protocol in different layers would repeat functions and be viewed as inefficient. This led them to a variation of the traditional phone company model that might be termed, "striped" beads-on-a-string, rather than a true layered model. The depth of their misunderstanding is indicated by the belief that a flag day was necessary. With an internet architecture, decommissioning NCP could have been done by just making another network available and letting the ARPANET atrophy. There may have been economic or political reasons for a flag day, but there would be no technical reasons. However with a network architecture, there are technical reasons for a flag day.

Since the property of different scope at different layers is so striking, it was not apparent to others that they had missed it.[13] To further confuse matters, there was a group in OSI, who saw the task as creating in OSI what was in the Internet. They had not understood the

[10]There was little or no overlap between SC6/WG2 and INWG.

[11]And if the politics had not been so intense and research had continued to develop better understanding, we would have learned it a bit sooner.

[12]While Dykstra's characterization may apply to general applications, it certainly does not apply to either networks or operating systems, at least not strictly in the form he stated it. In 1968, computing hardware were outrageously constrained. Dijkstra's characterization of layers was an artifact of these constraints. There is a pattern, but the hardware of the 60s (and 70s) tended to mask it.

[13]It was only recently that I suspected it, but couldn't believe that it could be the case. It is a bit rude to ask long time experts if they see something so fundamental. So the question was never asked.

importance of the address naming the "network-entity" rather than an interface, i.e the data-link entity. They even advocated something as naive as embedding MAC addresses in the NSAP address[14] which would have defeated the entire purpose of the structure. But they supported a connectionless solution and that the time connectionless needed all the support it could get. This difference seemed a minor point. OSI had the elements of the right structure.

N.B. With INWG 96, we had minor differences that didn't matter but were the primary focus of divisions. Here we have minor differences that have major implications, but were not the source of major divisions, partly because of more immediate pressures and partly because the technical implications were not an immediate requirement.

Another reason might be that since the Internet was entirely under DoD auspices, there really were no independently administered networks[15] except at the edges, which makes it easy to treat it as beads-on-a-string.[16] It does appear that this is a case of neglect, rather than intent. One certainly hopes it was not because they thought this was the right answer. However, we are looking at two layers of different scope. This should have been an indication that something else was going on.

This constitutes another rather major flaw in the Internet architecture. It is this confusion about Internet addresses naming interfaces rather than nodes that makes multihoming and mobility so cumbersome and complex and causes scaling problems with routing. In addition, an opportunity was lost when the host name file was automated. Rather than create an application directory, we got a step backward and macros for jump points in low memory. It is interesting that all other network architectures did not make this mistake.

[14]The field is called System-id in the address but does not say it could be the MAC address and for the clueful it wasn't. Designers were given the option to shoot themselves in the foot.

[15]These did not appear until at least the NSFNet, which was a considerably later and the mold was set.

[16]There was still a tendency to fall into the beads-on-a-string model. (It is easy to do. It is a very natural and intuitive model.) For example, many interpreted ARP as translating between IP and MAC addresses (as if IP ended at the edge of the Ethernet) when in fact it was a mapping between IP and MAC addresses.

## IV. Why Name the Interface?

There is still the problem of why name the interface? Why expose addresses at the layer boundary? The ARPANET followed the datacomm example for addressing simple terminals in a small network. Their task was focused on proving that a packet switch network was feasible. The subtleties of addressing were distinctly a second- or even third-order consideration. All of the subsequent architectures, e.g. XNS, CYCLADES, DECNET, OSI, etc. did not do this.

Clearly in an operating system, one would not expose addresses. Applications would be referred to by their name. The ARPANET had used the expediency of well-known sockets as a means to designate application protocols. Had the USING effort continued it probably would have fixed this and defined APIs that hid addresses.

OSI made the same mistake, but here we know why. It was an early compromise with the telephone companies, addresses were names of service-access-points that sat on the layer boundary. In the early discussions it is far from clear that the PTTs understood that this use of "interface" in OSI was software and not hardware. The computer companies involved in OSI were averse to defining APIs at the layer boundaries for fear that customers would want to expose them at every layer. However, some sort of interface definition was necessary to specify the protocols. Hence, service definitions were invented as essentially abstractions of APIs limited to only those primitives what would generate protocol.

In OSI, defining addresses as the names of service-access-points created one problem after another. Each requiring another complicating circumlocution to work around the problem. This culminated in the Network Layer group creating a typographical fiction to define "Network-Entity-Title" as the name of a Network-Entity, i.e. a network layer protocol machine.

Exposing addresses at the layer boundary was clearly a mistake made both OSI and the Internet and is indicative of a traditional, beads-on-a-string model and not compatible with networking.

But why would the Internet take a beads-on-string approach!? It should have been the anathema of what they stood for. We can only hazard a guess. For the people doing the ARPANET, most were operating systems experts and looked at the problem as one of distributed computing. Most probably had never had a data comm course. (Heck, most had probably never had an operating system course! They were only just coming into the curricula in 1970!) By the late 70s, new people coming into the 'Net were more likely to have taken an operating systems course and a data communications course. Where they were taught as very different subjects. At the time textbooks were very much data comm beads on a string, e.g. Gallagher and Dertouzos, The first networking text does not appear until 1981 (Tanenbaum) and it had many errors and misconceptions. These texts were very much oriented toward the beads-on-a-string model of terminal communication to a mainframe, where addresses had named interfaces. Another instance where the initial case was not the one to generalize from.

While operating systems did I/O, it was very machine dependent and seldom could general statements be made, so it was ignored. Most operating systems didn't even consider interprocess communication (there were a few), even though it was fundamental.[17] The networking approach is more represented by the Lampson text cited at the beginning. So as the shift to the Internet begins and with the demise of USING to continue the push toward a distributed computing model, the Internet at the critical juncture of getting the fundamentals right had a greater inheritance from the old guard beads on a string model than the new model the early developers had been pursuing. Combined with missing the key property of layers, the die was cast. A layer was lost, the address architecture was incomplete, and the protocol design was botched.

## V. Problems with IP

Splitting IP from TCP has not been without its problems. With IP in a different layer that was supposed to be "just the network" to do routing, fragmentation was seen as an IP problem. The byte sequencing put in TCP to make fragmentation at Internet Gateways easy obviously was now unavailable. And since IP packets are not sequenced, a new mechanism was required to do fragmentation. So a packet identifier, a more fragment bit, and an offset are included in IP. The trouble is that it does not really work.

Consider what happens when a packet is fragmented at a gateway, and the fragments are forwarded, there is a finite probability one (or more) of the fragments is lost. The destination has to reassemble the packet and wait for the lost packet to arrive. But it is never going to arrive. IP must hold the partially reassembled packet

[17]One of the more telling indications of this was in a report by Jon Postel, called A Survey of ARPANET NCPs. While Postel did not call it out, it was clear from the report that they fell into two classes: big ones and little ones. The big ones were on OSs with poor IPC, the small ones on OSs with good IPC.

for at least 1 Maximum Packet Lifetime (MPL), which is currently defined for IP as 5 seconds.[18]

In a little more than an RTT (¡¡ MPL), TCP will time out and retransmit, hand it to IP, which will assign a different packet-id and send it. This packet is fragmented, the probability not all fragments will arrive is the same as the last one. So with some probability, IP now has two copies of the same packet partially reassembled tying up buffer space. IP can't know that it has two copies of the same data TCP has sent! (Even if the packet gets there whole, the old copies will be retained until MPL expires.) As maximum packet lifetime (MPL) expires on these partially reassembled packets, they can be discarded. But there can be a lot of copies (megabytes) sitting around in 5 seconds.[19]

And if that weren't enough the 16 bit packet identifier field means that only 216 packets can be sent on all flows between the same two addresses within 1 MPL. Most of the time this is acceptable, but with more high-bandwidth applications, it has become more of a problem.

These are not big problems. There are workarounds for them. But it is the nature of the problem that is interesting. That the problems arise because IP needs to know what TCP is doing is a strong indicator it was not a good idea to separate TCP and IP in the first place. Splitting IP from TCP breaks two rules: 1) the functions should be independent, and 2) that functions don't repeat in the same scope.

Remember if the devil is in the details, there is something wrong with the design. Invariances in the problem have been violated.

[18]Ignoring, as current implementations do, the inconsistency between 5 seconds and any hop count including 255.

[19]This is not news. Reassembly interference had been recognized since the late 1970's soon after IP was created.

## VI. What happens if there is no split?

The problems simply don't arise. But more interesting, splitting an error and flow control protocol vertically instead of horizontally has many more benefits. In other words, split control from data, i.e. the feedback aspects of the protocol (retransmission and flow control) from the purely data transfer aspects. The implementation is simpler, lends itself better to hardware assistance, it decouples the feedback mechanisms from data transfer for greater efficiency and simplicity, yields both the unreliable and reliable forms without requiring two protocols, and it avoids heavy weight solutions like IPsec. There is no need for a protocol like IP. We must conclude that TCP was split in the wrong direction!

The reader may be wondering what happened to fragmentation. Don't we still have to worry about intervening networks with smaller MTU sizes? Precisely, networks. This is a network problem, not an Internetwork problem. The IP layer is above these network layers. The underlying network layer should do fragmentation. The IP layer has no business knowing the MTU of the underlying layer, any more than the Application should know the MTU of the underlying Transport Layer. Clearly, the idea of fragmenting in IP is more consistent with the beads-on-a-string concept of protocol translation between networks than a layered internet concept of relaying between networks. And of course, it has the benefit that reassembly can be deferred to the destination, so that not all fragments have to go through the same intermediate reassembly node.

In modern networks MTU increases as one goes toward the backbone. Traffic is aggregated for bulk transfer between intermediate distribution points. One wants to be routing more stuff less often, not less stuff more often. Then we should expect small MTU sizes at the edge and the problem is moot. The problem was temporary for early networks.[20] More and more, it has been recognized that MTU is a property of a layer. There was serious discussion of removing fragmentation from IP altogether during the IPv6 discussions and it was relegated to an option, probably ensuring very limited use.

It appears that the Internet experts saw delta-t as just a competing protocol and not as the vehicle for demonstrating Watson's important and fundamental results. They do refer to this period of the early 80s as the "protocol wars".[21] But as is the norm, the winner generally goes to the one with the most money, which in this case was DARPA. If the implications of Watson's results had been taken into account even further simplification would have been possible that would have also improved the security properties of the protocol.

## VII. Light begins to Dawn . . . Dimly.

Recently, there have been inklings among Internet experts that something was not right. Around 2000, efforts were begun to find a new architecture. After a decade of work that effort has come up dry. In fact, the organization of the efforts ensures such a result. Their success in the market has convinced them that much of what they believe is correct and has prevented them from getting outside the box to see that it isn't. With millions spent and no new insights, there has been a drift toward redefining the problem to something they can do. Even though, the fundamental flaws remain intact.

But more critically in 2006, it was noticed that router table size was on the increase again (either exponentially or combinatorially it was difficult to tell which as if it matters). It was being driven by demand for multi-homing, a problem we have known about since 1972. However, their devotion to routing on the interface was so complete that they attributed the problem to the absurd idea that they were overloading the semantics of the IP address and needed to separate locator uses of the address from identifier use[22], still missing that there are layers of different scope involved.

It is quite obvious that this is a false distinction: one can't locate something without identifying it and vice versa. In fact, all identifiers in computing are used to locate something in some context. Occurrences of "flat names or flat addresses" are either assignment by enumeration or using the identifier outside its context.[23] They should have noticed this in the mid-80s when the "multi-path routing" issue arose. That issue showed that multihoming wasn't supported for routers let alone hosts. It is definitely odd they didn't notice. The problem isn't overloading the address. The problem is distinguishing logical (internet) location from physical (network) location (see Figure 1). The Internet is lacking Internet Addresses and Application Names.

If the answer is so obvious, why didn't they just do it? The Internet is a craft tradition, not a scientifically based engineering tradition. First they had always done it that way, so the vast majority believed it must be right. Second, there had been a traumatic fight over changing it in 1992 when the IPng issue arose and no one wanted to revisit that. The attitude was that "It was software; there must be a workaround".Trouble is there

[20]Yes, it is possible to do it wrong and shoot yourself in the foot.

[21]It is amusing that they see this as a war between TCP and OSI TP4, when that decision had been decided with the INWG 96 vote several years earlier. In other words, the "war" if it existed had been over before the Internet experts knew it had begun.

[22]Someone will ask, What about IPv6? It does nothing for these problems but make them worse and the problem it does solve is not a problem.

[23]Example: the context for "MAC addresses" is manufacturer of network equipment.

isn't. So loc/id split is post IPng-trauma. (A more in-depth treatment of the loc/id split pseudo-problem can be found at http://www.pouzinsociety.org, Why Loc/Id Split Isn't the Answer.) Third, unlike other fields that teach engineering based on solid theory with some current practice, networking teaches only current practice, regardless of whether it is right or wrong. Consequently, the majority are not even aware of what the right way is, or why things are the way they are. All they have are (bad) examples to follow.

Embarrassing as it may be, solving the addressing problem is, in fact, obvious. It does require carefully laying out the elements in the various layers and carefully considering what is going on. In other words, it requires a careful scientific or engineering discipline be applied to create an architectural model, something that has been studiously avoided. When that is done, it is clear that the address names what does the relaying. With that, all else follows.

## VIII. Conclusions

It is clear that INWG had a clear idea of what needed to be pursued and was progressing along normal lines of scientific investigation. The elements of a complete architecture were there. The understanding of naming and addressing was there. The problems of congestion control were being discussed in papers as far back as 1972, a conference was held on the subject in 1979, and there had been on-going research. They weren't far from seeing that all of the layers were doing the same thing over different scopes and ranges of allocation.

In the late 70s, everyone saw that this was going to be a huge business opportunity and standards would be necessary. OSI was quickly bogged down in jockeying for position. Computer companies in Europe and Japan knew the Americans, IBM foremost among them, were the dominant market force. This created three problems: First and foremost, they (including the other American computer companies) did not want to give IBM the upper hand. This new effort coming out of research was the perfect foil to SNA. But at the same time, given the experience with the ARPANET, they did not want to give US companies a lead ("level playing field" was a big watch phrase at the time), even though this was still years from having commercial importance. In addition, they had to prevent the PTTs from declaring everything attached to the network belonged to the PTTs. The INWG model did both of these. The connectionless/connection war was already in full swing. Once OSI was a joint project with ITU in 1979, OSI became the focal point of that war and the differences between the two camps so great that OSI was largely doomed. As we saw, OSI was able to keep the INWG model in place, although overloaded with options[24] and other distractions. Europe[25] wanted everything based on X.25. It is hard to see how this could have sustained the applications at the rate of technology change that was taking place. As we saw, X.25 was a SNAC protocol and CLNP would have operated over it. There was a path out of that problem.[26] Standards are no place to do research, but OSI was able to contribute new insights with inter- vs intra-domain routing, working out that the upper three layers were actually one layer that repeated, and with network management (another place where the Internet took a step backward). But as indicated earlier, it would have taken major re-writing (equivalent to starting over) to clean it up and simplify

[24]Standards never make choices. When there are significant factions supporting something they create options and lets the market decide.

[25]Primarily the PTTs but they got a lot of support from European computer companies.

[26]X.75 was ever proposed as an OSI standard, so the only internet protocol in OSI was connectionless.

it[27] and probably could never have been accomplished in a committee environment as large as SC21 and SC6. Too many people had too much invested in the status quo.

The Internet took a wrong turn somewhere after the INWG vote. While there was considerable ARPANET participation in INWG, there was much less overlap between INWG and the Internet. Most of the people who were technically involved in the ARPANET moved up[28], and/or to working on OSI. The Internet was confined to DoD contractors. If this technology was going to be commercial, it would be done elsewhere. The Internet in the 1980s was primarily a network operations effort with new developments limited to accommodating growth. Telnet, mail and FTP were sufficient. There was never real network research effort in the US, i.e. on-going research to determine the properties of networks. The ARPANET was built as a proof-of-concept. But BBN did too good a job. It worked quite well for researchers and immediately became a production network. BBN was limited to one night a week for deploying new IMP systems, making their ability to conduct experiments quite limited. CYCLADES, on the other hand, was built as a network for research on networks but shut down for political reasons in the late 70s. Soon, engineers new to the Internet far out numbered the early ones and finding a working system naturally saw it as the right way to do things. More than that, they saw it as the only way to do things!

The vision of a resource sharing network was lost in 1974 when USING (User's Interest Group) ceased to exist. The Internet had clearly stagnated. No new applications were deployed between 1973 and the advent of the web in the early 1990s, which did not come from the Internet. It had become more about maintaining a tradition than continuing to develop. By 1980, one already heard the argument that, it was "too big to change". (!) But the understanding of the ideas was flawed. This too is not unusual. It is often the case that the adopters of a new idea do not have the deep understanding that the originators had. Many times, I have seen, a new idea laid out and the reaction is "O, now I see". But they don't. They see what was explained but miss the more subtle points that are not so intuitive (and often quite counter-intuitive) but crucial to getting it right. There is strong evidence that this is what happened here.

What is particularly astonishing is that among "best and brightest" no one noticed any of this or if they did, hid behind "well it is good enough"[29], or as is most likely, just went along. This is going to be great fodder for sociologists studying group think for years to come. After all, there was no point in complaining it only made getting published harder. Not a single network textbook over the last 30 years teaches the principles of networking or the fundamentals of naming and addressing that would have revealed the problems. Now we have a couple of generations of network engineers grilled in how to do it wrong.

If there is a lesson here, it is that new ideas were rushed into production too early and continuing work on understanding was frozen too early. Research needs to stay focused on understanding, theory testing, not return on investment.

[27]The task of collapsing the upper three layers into one repeating structure was a maze of documents to accomplish a relatively straight-forward task.

[28]Deservedly rewarded for their success.

[29]This phrase is a codeword for, "yea, I know it is wrong, but I want to do what I want to do".

## IX. Epilogue

The Internet is an especially important case given its role in the modern world, and the fact that so few people saw the problems, and the impact they might cause. We need to understand what happened.

IP gives testimony to the ability of software and Moore's Law, i.e. given enough thrust, to make even pigs fly. An immense screw-up upon which the world economy now relies.

As long as Moore's stayed ahead of Internet growth, all was good and the flaws never had to be confronted. In fact, quite the opposite. The success in the market proved the effort was brilliant, ignoring that this is the same argument that DOS is the greatest achievement in operating systems.

What is very disturbing is that these guys are 0 for 7 on major design decisions[30]: not only did they botch the architecture, the protocol design, and the addressing, miss an opportunity with DNS, but also botched the congestion control. We haven't discussed the latter so briefly: As previously noted, the problem of congestion control in datagram networks had been a known issue essentially from the beginning. However, the Internet was caught flat footed in 1986 with daily congestion collapse. They never saw it coming. The response is equally curious. They essentially adopted the scheme from Ethernet with a few tweaks. But for some odd reason they put the congestion avoidance in TCP.[31]

The effectiveness of a congestion control scheme is primarily determined by time-to-notify, i.e. reaction time. The Ethernet scheme works reasonably well because the maximum length of an Ethernet segment is 1 km or less, gjven the speed of light in coax, ensures a short tome to notify. With TCP, on the other hand, it is more like 104 km. Compounding the problem, the variation in the length (delay) of TCP connections means the scheme induces chaotic behavior. Worse yet, congestion detection is implicit which means that it will react to anything that appears to be congestion within the Transport Layer or below. Any attempt to introduce additional congestion control at a lower layer (with less scope and) with shorter reaction time will be thwarted by the longer reaction time at TCP. Explicit notification allows reaction to be limited to a single layer. The TCP scheme is predatory. The best one can say about TCP congestion control is that it might be acceptable in the limited environment of a network, but not in an internetwork! In the INWG model, congestion control goes in the networks, where time to notify can be bounded.

This track record defies the odds. Do these guys have some sort of anti-midas touch?! At least, one thread that seems to link all of these is that while they said it was an internet, they acted like it was a network.

Since before Aristotle, science has not been flattering to our intuition about the world often showing our intuitions wrong. This is why there is a scientific discipline, as a check on human intuitions. "Rough consensus and running code" is not sufficient especially when the majority has been taught faulty information. This is why science never has been, and never can be democratic. It is a tyranny of the data. This is a case of what happens when there is no design and when investigation of what the design should be is abandoned and becomes how to fix what exists. It is always the case that there are many more wrong ways to solve a problem than right ways.

If it weren't for Moore's Law it is likely that the normal self-correcting process of scientific investigation

[30]0 for 7, if we include the SNMP debacle and the "buffer bloat" fiasco.

[31]All previous research had focused on congestion control at lower layers where the scope was shorter and reaction time could be kept short.

would have taken hold. However in communications, there is almost a necessity to settle on a single solution. At least in OSs, while DOS was dominate in the market place, there was always UNIX and others indicating a different direction. Networking has had no alternative to consider.

What we see here in every case is making an immediate fix to an immediate problem with no consideration of how this leaves the structure positioned for the next step or moves toward some ultimate (even if never achieved) design goal. In each case, a construct (or discipline) needed to be maintained that while perhaps not needed immediately would be necessary later and was precluded by the immediate fix. In other words, it is a bad design. More specifically, each case treated the Internet as a network. The tacit assumption made was that every immediate fix was correct and did not preclude the next correct fix. Neither of which was true, of course. Their behavior seems akin to the novice programmer whose first reaction when given a problem is to start coding, rather than start thinking.

This pattern speaks to the recent fad of the Internet "evolving": that this is somehow a natural (and therefore correct) process. (This is clearly an attempt to justify, give luster and credibility, to what they are doing rather than a carefully considered scientific position. Scientifically it is clearly preposterous.)

Indeed individual mutations are an element of natural selection, but that is far from all that is required. Clearly, for evolution to apply, there would need to be billions of Internets all with different mutations competing and undergoing natural selection. What we have is one Internet where we assume that every mutation is perfect. The probability of that is vanishingly small. We already have a string of at least 7, which were all wrong in a major way. This is not origin of species by natural selection, but origin of a specie without natural selection. Unless the mutation is self-destructive (equally improbable)[32], it will stumble along and with no competition to compare with and group think being what it is, everyone will see it as a paragon of success. What we have is not evolution, but Rube Goldberg by accretion.

It is time to admit that the current Internet architecture is an evolutionary dead-end. No amount of patching is going to solve its fundamental flaws. The flaws were introduced early and are impossible to eradicate, if for no other reason than too many people have staked their careers on it. It really is DOS. And we know what happened when Microsoft tried to transition from DOS to Windows by incremental changes.

The good news? The road to a complete architecture can be smooth. The transition can be seamless and based on economic self-interest rather than fear and edict. Even better news? Down that road, there is a renaissance in networking and we will one day look back on the last 30 years as the Dark Ages.

## REFERENCES

[1] R. M. Metcalf, Strategies for operating systems in computer networks. In Proceedings of the ACM annual conference - Volume 1 (ACM '72), Vol. 1. ACM, New York, NY, USA, 278-281, 1972.

[2] B. W. Lampson, M. Paul and H. J. Siegert (Eds.), Distributed Systems - Architecture and Implementation, an Advanced Course. Springer-Verlag, London, UK, 1981.

[3] E. W. Dijkstra, The structure of the THE multiprogramming system. Commun. ACM 11, 5 (May 1968), 341-346, 1968.

[4] A. McKenzie, INWG and the Conception of the Internet: An Eyewitness Account, IEEE Annals of the History of Computing, Vol. 33. No. 1, 2011.

[5] V. Cerf, A. McKenzie, R. Scantlebury and H. Zimmermann, Proposal for an Internetwork End-to-End Transport Protocol Rev. 1 (revision edited by M. Gien, G. Grossman, C. Sunshine). IFIP WG 6.1 INWG General Note 96, 1978.

[32]And if the politics had not been so intense and research had continued to develop better understanding, we would have learned it a bit sooner.

[6] ISO/IEC 8648 Information Processing Systems - Data Communications - Internal Organization of the Network Layer, 1987.

[7] Discussion of INWG, OSI, ARPANET, etc is largely based on having been a member of the ARPANET NWG, INWG, and Rapporteur of the OSI Reference Model. 1970-1995.