Systems not (just) Networks

Contents

Note 1
Executive Summary 1
Preface 1
(Re)Factoring 2
Relationships 3
Wiring ➔ Binding 5
  Name 5
  xdev 6
  DNS 6
  IP 6
  AP – Access Point 7
Writ Large 7
More Internet? 8
Opportunity, Factoring and Disentangling 9
  Leverage 9
  GUIDS 9
  Opportunity vs Solutions 10
    Factoring and new Wholes 10
  Linking Information 10
For More 11
  Wait, Wait There's More 11

Note

I plan to rewrite this as a companion piece to "From Access to Connectivity" and address the concepts at a more general level. Stay tuned. That's a joke since "stay tuned" is an idiom that dates back to the days when each conversation had a dedicated frequency – a form of a "pipe". The real challenge is that so much of our understanding of concepts is implicit and embedded in the very language we use to communicate. And of course, "communicate" is another example since the "communicate" in telecommunicate is no longer aligned with the use of "communicate" in the very day says of conveying ideas, not just waveforms.

Executive Summary

The Internet is a proof of concept in demonstrating the power of shifting from a dependency model where applications are built in layers. With traditional telephony voice is a value-added service built on a provider's network. It depends on the providers' promise of a clear path.

Voice over IP is very different. It is implemented independently of the promises of a provider using any facilities available without assurances of a clear path. Of course, if there is no path then real time voice can't work but we can still implement other forms of messaging including text.

I talk about this as factoring the physical facilities from what we do with the facilities. The key is that this is a peer relationship akin to how we factor software systems and limit the interactions between the system elements.

This shift from layered dependency to factoring is a very general concept, as I explain below. It is also at the heart of public policy. Today the layered dependency model of telecommunications limits our ability to innovate to what adds value to a provider's facilities. It is an approach that harks back to the days when we had dumb telephones that dependent upon a network operator.

Today voice is an app and not a value-added service. We don't depend on the provider's layers. But our public policy lags because a century ago we created a regulatory system to assure the economic viability of providers with high capital costs and little differentiation. Broadband is still the old model – just with fatter pipes.

We need to shift to a fundamentally different model – a Public Packet Infrastructure. This removes the layered dependency of the old model and also taps into the abundance all around us. We've already had a taste of it in the vast capacity of today's Internet, but we're still limited by a gatekeeper who demands payment for each innovation. Even with VoIP you need to pay a provider.

We know how to incrementally shift to a public packet model. What we lack is an understanding of the value of such a model. Society benefits from the innovation and devices and services providers are free to innovate without having a third party in the path.

Preface

The Internet is the face of a new paradigm. A paradigm is a pattern of thinking. The classic example is the shift from geocentrism to heliocentrism. Nothing changed except our way of thinking about the universe and our role in it. That new way of thinking helped Newton see the regularities that underpinned his physics and birthed our modern world.

The concepts that gave us the Internet are no more about telecommunications than heliocentrism was merely about astronomy. They represent a fundamental shift from a reductionist view to a more holistic understanding of systems.

Our thinking still centers on the networks with devices as clients and speech as a value-added service. This belies the reality in which we design systems without having to think about networks at all – just the relationships between the devices (endpoints).

This corresponds to a reality in which the closer we look the less meaning there is. Look at your computer screen – if you get too close you just see pixels, not words!

Yet our public policy and popular understanding is stuck in the past harking back to a time when phone calls were a service not an app. We still pay for the fiction of making a phone call when, like a Zoom call, it is just an app running on the shared infrastructure.

We pay a heavy price for this by limiting our ability to innovate and create new services by having a provider acting as a second-guesser in the middle of all transactions and relationships. I've compared it with having a sidewalk provider putting a turnstile in front of each house thus creating the concept of affordable strolls by demanding payment – as a form of asking permission. We had a real-world example when ATT offered True Voice as a way to improve voice calls, but it broke the modem connections that weren't in the use case envisioned. Fortunately, they backed off.

The Internet is central here in demonstrating another way of thinking – one in which our relationships are first class, and we are not dependent upon a provider. While I use the Internet for framing many of the issues, this is not about the Internet as such but rather than ideas it represents.

(Re)Factoring

It is remarkable that I can simply type in a URL and am at the website. It doesn't matter whatever is between me and the site nor where we are. Yet we take this for granted and, in doing so, we deny ourselves the full benefits of this powerful idea because it only works if there is a human in the path to assure an account with a "provider."

The Internet is the most familiar face of a new paradigm that turns our traditional view of layered design on its head. We're used to thinking of layered design with a connection to a website built upon the layer of technology – the wires and radios of a telecommunications provider just like in the days when there was a wire from each phone to a central office and then you could follow the path all the way to the destination.

That's not how the Internet works today. Your connection is not dependent upon any particular path or wire. Instead, you just type in a URL and the packets can use any means necessary. Instead of having a phone company assure a clear path, Voice over IP is opportunistic and adapts to the capacity available. If there is sufficient capacity, then you get video too. By converting all the capacity to packets and allowing the user, not a service provider, to decide how we can share the capacity, we found that we have enormous capacity for connecting.

To understand this new approach, we need to shift from thinking of layers of dependency to thinking about the relationship between the endpoint such as your computer and a website or two telephones. We factor out the relationship and are no longer dependent upon smart physical infrastructure to preserve a waveform as we did when we had a special network just for phone calls.

Conversely those implementing the facilities don't have to (and can't!) know the particulars of the applications because that knowledge does not exist inside their facilities. Two people may be watching the same video but to one, it may be entertainment and to the other it may be a vital life-changing experience.

Part of the problem is in the language we use and the word network which is a very general word for any collection of endpoints. A social network is just a collection of people. The Internet, in particular, is not a physical network but rather a way to use any available facilities. But words matter and by calling it a network it makes it seem as if nothing had changed when everything has.

I compare this shift from Ptolemy to Copernicus. Both described the same system of planets but there was a fundamental shift from a universe that revolved around us to one in which are participants in a far larger universe.

Layered dependency is like geocentrism – I compare it with the idea of "Turtles all the way down" whereas a factoring model isn't layered and meaning comes from the larger context.

I write about (re)factoring because there isn't a single decomposition but rather working relationships. This is similar to the use of the term "redisplay" in spreadsheets to emphasize the dynamic and ever-changing nature of the information.

We can think of factoring as an engineering or system design principle. By factoring out our connection to a web page we can focus on the task at hand entirely apart from

the implementation of the connection. This removes the static friction and allows what is called permissionless innovation. You don't depend on a layer being provided.

I use the term Ambient Connectivity™ for the ability to just assume connectivity. The assumption may be wrong – there may not be a path, or it may be limited. This constraint is what made Voice over IP transformative. The engineering constraint was that, unlike traditional telephone, you had no relationship with the path provider and couldn't buy a clear path. VoIP only worked well once there was sufficient capacity. That capacity was not provided for VoIP but was rather a byproduct of the capacity created for the Web, VoIP was opportunistic.

This is an important point. Traditionally we try to solve problems, and, at the cost of trillions of dollars, we built a phone network. In sharp contrast, there is no cost to VoIP as an app – it just takes advantage of the available capacity. You pay for the generic connectivity but not for how the apps use it.

The phone system was built a century and a half ago for a particular use case – it was the only option then. And, indeed, spending trillions of dollars we could accomplish that goal. But the result is that we have a network that is good for that one purpose without the synergy one gets from having a shared packet-based infrastructure! To be fair, we have repurposed the existing telecom facilities as a packet infrastructure, but the value is pennies on the dollar compared with a native packet infrastructure.

The Internet is a product of its time – it was intended to be a network-of-networks in the classic sense but the problem of coordinating disparate networks with many different requirements and protocols was an impossible challenge. Even the traditional phone companies couldn't get their packet networks (ATM – Asynchronous Transfer Mode) to work across carries becomes composing the promises (Service Level Agreements) was too much. And, even if one could, only the users outside the network knew what their particular definition of "quality" was.

The solution came from the constraints of early packet networks such as ALOHAnet in Hawaii and Ethernet which was inspired by ALOHA. Those weren't traditional networks because all they carried were packets without any information about the context (or intent). The key was to use the intelligence in the newly available computers to program around those limitations and, retry if packets were lost. This mimicked the traditional reliable phone network, but the surprise was that that reliability wasn't always a good idea. After all, if you missed a temperature reading you didn't care. So, as we do with streaming, just send the current temperature.

This came together in "End-To-End Arguments In System Design which explained that we could (and should) be able to implement solutions without building special facilities into the path (the network).

We can take this idea beyond networks and see it as a general engineering principle of factoring out relationships and systems rather than building upon layered dependencies.

Important: I am using End-to-End as a term of art to focus on relationships between endpoints. But articles like this use the term in a very different sense – taking responsibility for all the details along the path. This is essentially the opposite meaning. This is a problem with trying to talk about new concepts and having to repurpose existing words.

This is especially true when the concept of relationships between two endpoints that doesn't depend on the middle is alien – it defies our physical metaphors. That's also why quantum mechanics is so difficult.

Another form of misinterpretation arose when I asked Microsoft copilot (ChatGPT): what is the meaning and origin of the term end-to-end:

The term "end-to-end" is often used in the context of communication and technology. It refers to a system where data is encrypted from one end (the sender) and only decrypted at the other end (the receiver). This means that the data remains encrypted and secure during its entire journey from sender to receiver. The term "end-to-end encryption" originally meant that the communication is never decrypted during its transport from the sender to the receiver. The concept of "end" in this context refers to the boundary or extent of something.

Here too we see an explanation that focuses on an accidental implementation detail rather than the larger concepts.

Paradigms do not change without a struggle. It requires finding people with whom the new ideas resonate because the very concept "does not compute" (to use a term from the 1950's science fiction).

Relationships

This is not about the Internet as such though I do make heavy use of Internet protocols. It's not about networks (in the sense of telecommunications) as such though it is about networking.

It's been over forty years since the publication of the End-To-End Arguments paper[i] which articulates a basic design principle of the Internet. Just as Metcalfe's law is a special

case of the network effect, the idea of relationships independent of what is between two endpoints.

For example, if you hear someone yell out "7" you can't know what it means because you don't know the context. That context might only be available to the two people having the conversation. We use the metaphor of the dog whistle (a high frequency sound only dogs can hear) for messages that are invisible to intermediaries but reach the intended audience.

This is in sharp contrast to the idea of a network carrying meaningful messages. In VoIP Inflection I use Voice/Video over IP to contrast the traditional idea of layered dependence (voice being dependent upon a network provider) and an approach I call factoring in which the facilities we use to exchange packets are independent of the meaning of the packets.

This approach of factoring is central to software engineering where we limit the interactions between components of each element of the system. This is of course a general principle of systems but with software we can more easily refactor systems as a way of iterating our designs.

A closely related concept is binding in which we define the relationships between endpoints and can operate on them. This ability to manage relationships is key to the success of relational databases which allow us to manage the bindings using abstract linkages to connect tables. This is why we use (in the US) social security numbers rather than names. You can change your name without losing your benefits.

Enough abstractions. Let's use a real example. The table below has the following fields:

- **Name**. The name I use for the device function or location.
- **IP**. The IP address of the device. Note that can be reachable by multiple paths.
- **xdev.** The database identifier for the device. Note that a device might be a sub-device when a single IP address has multiple elements.

Name	IP	xdev	DNS	AP	Last	13432	13419	13397	13374	1
20:04:40						20:04	20:04	20:04	20:03	2
3805	172.20.2.146	3805	LIFX-DLCOL-76592A	be3 Office UP5-6 (#1 @47)	0207 2004	113	132	114	143	
3806	172.20.3.243	3806	LIFX-DLCOL-766268	be3 Office UP5-6 (#1 @66)	0207 2004	94	2	84	105	
bellfront	172.20.2.251	3171	shellyplus1-a8032abc4fa8	748 UP7-6 Basement (#11 @59)	0207 2003	31	136	19	31	
bellrear	172.20.2.211	3170	shellyplus1-b8d61a8a7074	748 UP7-6 Basement (#11 @66)	0207 2003	247	1001	70	179	
bellrearbuttonsensor	172.20.0.85	3099	shellyplus1-a8032aba7cd4	5E3 U7-Pro (#1 @38)	0207 2004	20	5	31	84	
br2mStripBack	172.20.0.54	2918	wiz_77cd58	be3 Office UP5-6 (#1 @40)	0207 2004	71	186	55	59	
br2mStripDway	172.20.1.216	2921	wiz_7751ec	5E3 U7-Pro (#1 @45)	0207 2004	215	129	113	141	
brmf1	172.20.0.32	3068	tasmota-10515B-4443	be3 Office UP5-6 (#1 @29)	0207 2004	2	4	2	3	
brmf3	172.20.0.129	3062	tasmota-103D2F-7471	be3 Office UP5-6 (#1 @32)	0207 2004	2	7	13	4	
bsb1	172.20.2.214	3231	LIFX-DLCOL-757D25	bfc kitchen HD (#11 @39)	0207 2004	68	74	98	137	
bsb2	172.20.0.67	3227	LIFX-DLCOL-76E096	bfc kitchen HD (#11 @37)	0207 2004	65	652	204	148	
bsb3	172.20.1.250	3226	LIFX-DLCOL-7696A8	5E3 U7-Pro (#1 @49)	0207 2004	63	590	175	50	
BsmbA1	172.20.1.119	2912	wiz_bed6cf	748 UP7-6 Basement (#11 @33)	0207 2003	24	232	32	36	
BsmbA2	172.20.3.27	2904	wiz_becbc5	bfc kitchen HD (#11 @27)	0207 2004	24	29	68	11	
BsmbB1	172.20.2.182	2911	wiz_bd2b34	748 UP7-6 Basement (#11 @33) 5E3 U7-Pro (#1 @39) be3 Office UP5-6 (#1 @28) 5E3 U7-Pro (#1 @38) be3 Office UP5-6 (#1 @29)	0207 2004	24	12	7		
BsmbB2	172.20.1.3	2900	wiz_bec0ed	be3 Office UP5-6 (#1 @28)	0207 2004	32	1	24	9	
bsmbb3	172.20.1.173	3077	tasmota_3F5A54-6740	748 UP7-6 Basement (#11 @24)	0207 2003	91	178	3	59	
BsmbC1	172.20.1.96	2362	Lifx-25F234	748 UP7-6 Basement (#11 @29) 5E3 U7-Pro (#1 @32) 748 UP7-6 Basement (#11 @29)	0207 2003	1	159	179	50	
BsmbC2	172.20.3.122	2879	wiz_800a73	748 UP7-6 Basement (#11 @37)	0207 2003	66	137	175	50	
BsmbD1	172.20.1.194	3800	LIFX-DLCOL-76BFFA	748 UP7-6 Basement (#11 @42)	0207 2003	186	2484	2030	545	
BsmbD2	172.20.1.44	2848	wiz_bec187	748 UP7-6 Basement (#11 @48)	0207 2003	55	144	245	16	
BsmbE2	172.20.3.217	2878	wiz_bd3435	748 UP7-6 Basement (#11 @38)	0207 2003	53	180	255	197	
Dining1	172.20.2.128	2882	wiz_6594c6	bfc kitchen HD (#11 @31)	0207 2004	25	20	94	17	
Dining2	172.20.2.174	2883	wiz_6a3810	5E3 U7-Pro (#1 @40)	0207 2004	39	187	134	68	

- **DNS.** The DNS name of the device. It typically corresponds to the MAC address, so I don't list the MAC separately.
- **AP.** For Wi-Fi devices, the Access Point the WI-FI device connects to.

I wrote the tool because I was having problems with delays and many devices were not responding. For some reason, after I wrote the tool, the system started performing far better and I don't know why. I still have problems with long delays for pings and need to understand that better. But those performance issues are a separate topic.

What is relevant here is how the tool evolved and the constant refactoring. When I initially started using home control using **X10**, things were relatively simple with 256 addresses available. I would assign an address to a light socket (smart bulbs weren't a thing then). Even then I would use a database to associate a name like **ksink** (for the lights on the sink side of the kitchen and associate it with **J15**.

Things have evolved since then with many families of devices and protocols. There are also abstract devices created by groups and rules.

I wrote this ping tool after I upgraded my Ubiquiti system and used the Unifi application to get a view of the network devices. But I needed a tool to give me views of the other devices I have and their relationships. One big limitation is that, unlike Ubiquiti's devices, there isn't a standard interface for providing rich information about my devices.

For now, I've set a limited goal of determining which devices are reachable and how long the delays are. That in itself is very useful. In the future I hope to extend the approach to other overviews.

This is in sharp control to applications like HomeAssistant which are more focused on "home automation". I see that as an application that uses devices are resources and not fundamental. Again, this is a factoring of the problem. Home automation is a use case that drives adoption but, over time, we see that it is – just that, one use case. Other observers might use the devices differently.

This is especially true for the traditional telecommunications industry which was created to support specific applications such as telegraphy and the talking telegraph, AKA, the telephone. Today the Internet has shown that the facilities can be better used as a generic packet medium by factoring out applications such as phone calls (VoIP). The problem is that public policy has lagged thus frustrating our effort to take full advantage of the power of a Public Packet Infrastructure.

One reason is that we are used to layering with voice as a layer on top of a telecommunications infrastructure rather than seeing voice being factored out which the existing telecommunications facilities being one means of exchanging packets This is why the use of the term "network" is so pernicious in that it hides this fundamental change by allowing us to continue to use the old terminology as if nothing had change – even though the very concept of telecommunications as a transport service no longer makes sense.

With software we have many tools for refactoring but it's far more difficult when public policy embeds assumptions such as the idea that telecommunication's business model of selling services such as voice is no longer viable now that voice is an application.

Wiring ➔ Binding

When we have dumb endpoints, we rely on the network to add the smarts be it to maintain the telephone waveform or connecting a light switch to a fixture so one can turn a bulb on and off.

And just as in telephony, we now have smart endpoints and are no longer constrained by the path. The smart light switch (an app smartphone) can send a message from anywhere in the world to the bulb. I can take advantage of this capability to write an app to control the bulb in response to an event.

We can use this capability to automate a particular function, but the important point is the particular application but the very ability to implement such applications. The powerful idea is that we can have a relationship between the endpoints independent of the particular path or, in this case, wiring plan. I don't need to call an electrician to rewire my house just to move a light switch.

Once I'm no longer constrained by the wire, I can take advantage of the opportunities to send rich messages. I can also define systems (or meta-objects) such as a group of bulbs or even something more abstract such as "the light I need to allow me to choose what clothes to wear". It may be implemented by opening a window shade. Perhaps we would want more control and choose the particular kind of light (color temperature) for a particular venue.

A technical term for defining the relationship between to endpoints is "binding". The simple example is to bind a function to a bulb just as we use a URL to bind a browser to a website. It's also possible to be more abstract, as we see in the example of binding to the particular lighting scenario but for the purpose of understanding it is sufficient to thin in terms of connecting a switch (function) to a bulb (light source). We can change relationships by simply changing the bindings. I can "move" frankston.com from one computer to another by simply changing the IP address it is associated with. I can bind a SKU (part number) to a particular a particular instance of a product that is closest to me rather than shipping one from across the country (logistics).

The key point is that these bindings are first class relationships independent of how we use them. In the case of the SKU, I can use it to inform the shipper of my intent rather than sending a message to the product itself.

I learn by doing and by having concrete examples to build on so let's look at the bindings in this particular example. Binding is a very general concept, so each use case has its own nuance:

Name

The name is the first column and already we have nuance. The name is not an intrinsic property but rather a handle I use for the device just as someone might be called "lefty" by friends who might not even know his parents call him "Melvin".

This is the name I use for my own control purposes and scripting. The name is likely to embed semantic such as

"the first bulb in the row on the sink side of the kitchen" which I write as "ksink1". I do have a description field.

The nuance is that is not actually the name of the bulb. It is the name of the location of the bulb. If I want to change the bulb itself, I use a "swapbulb" command I wrote which rebinds the location to the particular bulb (device).

I can then create a group called "kitchen" which is bound to the bulbs in those locations. If I swap a physical bulb, there is no need to update "kitchen" which is bound to ksink1 and not the bulb itself.

The lack of a separate binding is why I can't use Alexa. It does not scale beyond the demo or trophy home level. Perhaps is not a problem if you don't swap out bulbs too much and, with LED bulbs, there isn't much need to. I had to because I kept having connectivity problems. Ironically one byproduct of this monitoring effort is to reduce the need to swap bulbs. The failures are indeed a learning experience, and they forced me to understand scaling and management issues.

Note that the name field is also used for DNS names that aren't devices. Those are just names with a wider scope.

xdev

xdev is the local database key for the devices. I use an autoincrementing integer as the database key for convenience since I often need to type it in. The name is bound to the xdev which I can then use to point to the device itself.

My swapbulb apps simply changes the binding of the name to xdev.

Note that there are many devices without DNS entries such as Zigbee and Z-Wave devices (which I've pretty much eliminated).

I can also use the xdev in lieu of the name (as in the 3805 and 3806 in the example). I happen to take advantage of the fact that xdevs are numbers and names are not. This is a convenient overloading of mechanisms which may offend purists, but my goal is simplicity not purity.

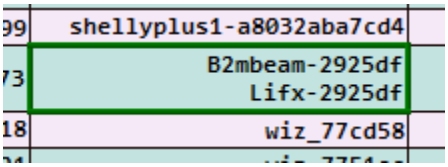
DNS

The DNS column is basically the intrinsic name of the bulb. I use the term DNS since that is the binding mechanism. Again, more nuance, since I am using DHCP and relying on it to act like a local DNS. This does not seem to be an official protocol, but it is vital. It is a good example what I wrote about in how I repurpose existing Internet protocols for my needs.

I used to name the bulbs with their descriptive name. In this example you see the old name (**`B2mbeam-2925df`**) and the name version which is basically the mac address name assigned by the manufacturer.

In the earlier approach it was convenient to be able to just look at the device and its place and purpose. But it also made it difficult to change. Once I had improved my tooling including my "swapbulb" command, having semantic information in the name was a problem.

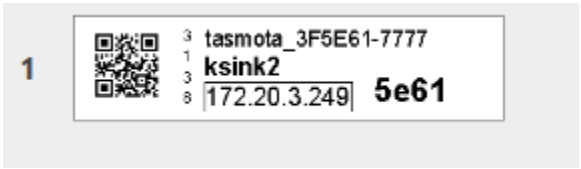
Another form of built-in semantics is the group name. This approach is used by Lifx and others. But, as we've seen, a device can be part of multiple groups, so the mechanism works only in the simplest cases. It does provide some degree of organization within their app.



The key assumption is that this name is unique and stable. The mac address is similar but is not usually directly visible.

IP

I need to know the IP address because that's what is used to send a message to the device. I could just use the DNS and let the other routines map to it.



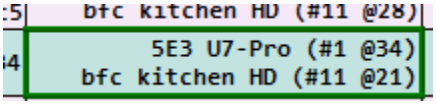
This is a label I generate for the devices. The QR code corresponds to the DHCP name (the local one, rather than the fully qualified name so it can be moved to another site). The IP address is here for convenience but may change. When I moved from the older Ubiquiti USG (Security Gateway) to the UDM (Dream Machine) pro I lost all my old IP addresses. Turns out is it not a big deal since I don't rely on them for long term stability.

Note that I am not using IPv6. That is partially a tooling and support problem but, more to the point, the V6 address is provided from an external provider whereas the local IP address is locally generated. IPv6 link-local would be more useful – more akin to using the stable mac address without the need for extra tooling but the support isn't there.

AP – Access Point

This is not a binding in the same sense. The relationships between the sender and receiver of a message didn't depend on which AP was being used.

The reason I expose the AP for diagnosing problems. I can see the channel and signal strength for connections. I can also observe when the AP changes.



I can manage the path of the packets without impacting the application. So, if there is congestion, I can fix it without worrying about entangling with the application.

This is vital diagnostic information, but we have to be vigilant about depending on the accidental properties of the path. This is one of the major problems in repurposing the existing telecommunications facilities – the tendency to view the legacy smarts as still being necessary rather than seeing that those smarts are working at cross purposes with the relationships between the endpoints.

Writ Large

There's a lot more that I'm learning from managing the many devices in my house but the goal here is to focus on the concept of binding and relationships between endpoints in contrast with traditional networking which is depending upon smarts in the path.

The very concept of end-to-end relationships independent of the path represents a stark contrast in how we think about systems and public policy.

In this context, today's Internet is an engineering proof-of-concept. And, with no small degree of hubris, I claim that the experimenting I am doing in my house is another proof-of-concept.

I first came across the concept in class when we studied ALOHAnet – a packet radio network in Hawaii. There is no network operator. Instead, it relied on the smarts in the computers at the endpoints to make use of the opportunity. One option was to retry sending packets until they get through but other approaches such as just sending the latest information (such as the weather) periodically also worked. While I didn't fully appreciate it at first, this approach completely decoupled the intent from the packets of bits. You can see this clearly with Ethernets (Bob Metcalfe's project in that class). If you look at the packets on the wire you know nothing about their context or intent.

This complete decoupling of the relationship from the packets in the path removes the static friction associated with traditional telecommunications.

This is the key to what made the Internet so transformational but it only hints at the larger concept because we still view it through the lens of traditional communications and second-guessers are all along the path. We accept this because we are looking through the lens of traditional telecommunications and don't know what we can't see.

When some of the ideas resurface, they are seen as new ideas. The Internet is inherently peer to peer and is built on things (with smarts) and interconnect. But people only see the surface use cases. The Internet acts just like the old bulletin board systems they dialed into and the web and social applications they use. No wonder it seemed asymmetric with larger services and users at their mercy.

Even when I write about giving people ownership of their community infrastructure I get pushback because worry about the mayor trying to operate a phone company. That misses the point, it's just the opposite. We are asking a phone (or cable company) to operate facilities that are fundamentally different from the way they currently operate their facilities. The very idea of putting smarts in the network creates static friction.

Network-neutrality is an attempt to address this, but the problem is just bad actors but also misguided efforts to do us good. The current policies are sustained by confirmation bias. Instead of being excited by how well Voice (and Video!) over IP (VoIP) works, they attribute the success to smarts in the network.

We see this in today's broadband policies. When I was at Microsoft, I took advantage of the broadband that was being rolled out in the 90s. It was intended to be a way for the carriers to offer new services without the cost of new facilities. But it was very much modeled on the days of using a modem to dial into AOL with a monthly fee for each device. By putting the router (a Network Address Translator) between the house and the broadband pipe I was able to repurpose it to allow peer connectivity with the Internet. A home user could have any number of devices without paying anything extra. The idea worked so well that it achieved a 100% adoption rate! And that's the problem – there was no sense of discontinuity. It seemed just like the old approach but faster, AKA, High Speed Internet.

The Internet isn't the wires (or "pipes"), it is the way we use them.

But it is important to recognize that the Internet itself is more of a use case – one that is also a resource for others to build on. It has allowed us to implement traditional net-

working services such as phone calls and accessing remote sites but without relying on a traditional network?

More Internet?

The reason for the Internet's success is that it enabled permissionless innovation in sense that I didn't need to negotiate with a path provider to implement a new service. This was possible due because of the complete separation of the application (the intent) from the bits in the packet.

The reason I cite VoIP as the inflection point is that some applications still seemed to need special treatment in order to meet performance requirements with voice being one of the most demanding. It turns out the answer was not in smarter network but in adding more generic capacity. The capacity created for the web enabled VoIP.

The irony is that, to a large extent, messaging has displaced voice, and we could also do a voice version of messaging – recording snippets and sending them off. VoIP is also free because it is an application built on the opportunity created by fungible connectivity. The facilities might cost but there is no additional cost for voice as an application.

The Internet didn't have a requirement of supporting voice. This is in sharp contrast with a phone networking purposely built for voice and was thus unable to evolve beyond that design point.

What enabled VoIP was the Internet as a technique for taking advantage of any opportunity by having fungible packets convert any facilities into fungible packets without predetermining their purpose.

We see the same fallacy in the concept of High-Speed-Internet. Sure, having more bits per second is really nice but not fundamental. The web was initially built using dialup modems - just like the earlier online services. That drove a virtuous cycle of opportunity and innovation.

Today's broadband has turned toxic. I know it is popular and rightfully so. It is wonderful compared with the alternative, but it is a local optimum. It's as if we were stuck with the 1970s phone system because we couldn't imagine anything better and accepted that $1/minute was the real cost of long distances. The idea that VoIP not having a cost would've been unimaginable and, if anything, a threat because it didn't make guarantees. Today I'm all-too aware of how much more is possible beyond today's broadband.

Taking advantage of the existing telecommunications infrastructure enabled the rapid growth of the Internet. It is a technique that could take a wire dedicated to one phone call and enable it to be shared by thousands. This is despite many of the links being too smart and overengineered for the purpose. Or, worse, lying to the devices about the actual capacity.

The largest barrier to permission list innovation are the myriad second-guessers along the path such as firewalls. How can I write an application when I don't know what firewalls are going to decide my bits are not worthy? Of course, we can't just turn off firewalls, but we need to start taking approaches that aren't dependent upon a "village walls" approach.

I understand that these efforts are well-intended and, indeed, I was complicit in using the home router as a firewall because early Windows 95 applications were written on the assumption that they were only accessible from trusted local users. Thus, we had to block access by default.

The problem comes when we confuse necessary kludges with architecture. The intent was that when we could assume encrypted connections and aware applications, we'd dispense with such artifice.

If you think of a classic Ethernet which is just a coaxial cable as a contained radio medium, then it is meaningless to ask a piece of copper wire to implement such a policy.

We accept the limits both because we're inured to them because traditional telecommunications has been a service that came with assurances and because the idea of a secure network seems like a good idea. But as we see with the that networking is not inherently a service – telecommunications is just another use case.

That said, it would be nice not to have to worry about all possible problems when writing each application. The key is to recognize that such protection can be an optional service. You can choose to have your messages relayed by a firewall or other vouching service while having the option for unfettered connectivity. And you can choose the kind of firewalls or protections that meet your particular needs.

One size does not meet all and a facilities operator cannot know the particular needs of each application. By making such vouching an explicit service then we allow for innovation.

The reason that this is not the norm is that the current approach seems to work very well but that's only because we don't know what we don't have. This is a good example of permission-based innovation since we need to petition to have the barrier removed.

I often use the example of the connected pill bottle which is a very simple idea and can be done for under $20 using

existing technologies. The reason we don't have them is that the difficulty of negotiating for passage and onboarding the devices is too much of a hurdle. Not only are we missing out on a simple opportunity to improve people's lives, but those providing services (AKA cloud providers) are missing out on all the services they can offer for connected devices.

The biggest and, perhaps, the most egregious problem is the persistence of a telecommunications business model and architecture that no longer makes sense. Traditionally telecommunications companies made money selling services. Now that those services are applications that business model no longer makes sense but instead of seeing the opportunity to create a new infrastructure, we have institutions like the FCC that are firmly planted in the 19th century.

The consequence is to put an impenetrable barrier along the path – the paywall necessitated by a subscription model. It is a failure first approach because allowing free riders would subvert the legacy business model so, instead, we have a corporate welfare system to keep the model alive.

After all, we don't know what permissionless innovation is and are scared by open connectivity, so we accept scarcity. A major source of scarcity is to shard the abundant capacity into thin pipes. With wired connectivity, we are largely packet-based but still have the artifact of multiple providers of the identical connectivity. This alone should be evidence of a fundamentally flawed market.

The situation is even worse for wireless in which we have worked diligently to create the idea that we need to assign a serviced path for each conversation and then sell that path off to providers [sic] who can monetize it. That harks back to the days when we'd run a dedicated wire to each phone whether or not it is in use!

The solution is the Public Packet Infrastructure which lets us tap into the abundance all around us and removes the fatal barrier of the paywall. And, in the US, it honors the US Constitution's ban on unnecessary impediments to free speech.

While we're still looking at the Internet, we need to ask why the whole DNS (and the Web) unravels every year. This is inexcusable and 100% unnecessary. The fatal error is linking the endpoints using semantic-based identifiers – something that would get a student flunked in a database class. The alternative is to remove the semantics from the links. I explain more in my Forever URLs proposal. The approach would be a step towards a more distributed Internet than today's which is focused on the dialup idea of "access".

Because the Internet is a way to use the existing facilities and not the facilities themselves, the proposals can be implemented by repurposing the existing facilities rather than building new ones.

Because they make far more efficient use of the facilities and create real competition (as opposed to the fantasy competition mandated by the FCC), the costs would go down tapping into the abundance all around us.

Opportunity, Factoring and Disentangling

Leverage

As we've seen, End-to-end relationships independent of the path between remove the static friction that comes from coupling systems.

There is another form of this in the rapid innovation enabled by shifting from teams to loosely coupled individuals. This is my approach to programming – providing tools that give me leverage as well as using shared tools and libraries.

These libraries have well-defined interfaces which limit the interdependencies. These dependencies can limited by having versioning systems that allow one protection from breaking changes.

A related idea is using GUIDs to define new interfaces. Instead of changing an interface, one can define an additional interface with a new GUID. The programs using the old interface would continue to work while having the option of embracing the new capabilities. Microsoft COM3 is built on this idea.

GUIDS

The Internet is a useful laboratory for understanding concepts and experimenting with them. But the ideas go far beyond the Internet. It is also important to acknowledge that the concepts long predate the Internet. We long known that we need to use abstract identifiers for linkage. In the US we have Social Security Numbers so that you don't lose your benefits if you change your name.

There is tendency to by putting semantics in the identifiers. Social security numbers used to encode the place they were issued, but perhaps, that was because we didn't have a good way of managing a distributed systems in those days.

Phone numbers are another example. They started out in as paths through a switching system with each digit you dialed operating the next rotary switch. But today they are just abstract identifiers but, alas, with legacy semantics. There was an effort to create a DNS branch for phone

numbers (ENUM), but the legacy carriers fought it and insisted that the semantics be part of the number. This is reminiscent of the firewall issue.

There is no more reason to cling to the artifice of dialing numbers than to require people to type in IP addresses to reach websites.

The key idea is the GUID (Globally Unique Identifier AKA UUID or Universally Unique Identifier). These are used in databases and are basically random numbers that are highly unlikely to be duplicated so that two databases can assign records confident that they won't be in conflict.

They are not numbers in the sense of being able to do arithmetic on them. They are just abstract identifiers without any inherent semantics. This is what makes them so useful – the semantics is imposed by the context or usage.

(As an aside, such abstract tokens are also the basis for the LLMs used in AI).

This idea is fundamental and reflects are reality in which meaning comes from context and usage and a real world that is very distributed and constrained by the speed of effect (AKA, the speed of light). I get into some of the philosophically in writing about The Age of Software.

Opportunity vs Solutions

We tend to try to solve problems but that is a very inefficient and problematic approach. It is far more fruitful to create opportunity and then discover what we can do with the opportunity (without having to ask permission).

The virtuous cycle is part of this when the solutions we create become resources for further iteration. The web is a wonderful example because it started out a very simple protocol. It was open without any attempt to prevent innovation thus leading to today's web, which is very different than the earlier version.

This is the power of an infrastructure approach in which creates resources for a community without being limited to a single purpose. Each infrastructure has its own origin story because it's hard for people to see beyond their familiar use case. After all, why would anyone need a talking telegraph when you never spoke to people far away?

Basic research and education are other examples of resources to build on. This is why it is so tragic that education is seen only in terms of monetization and researchers see their efforts measured in terms of how much someone will pay now. No wonder radio researchers have to create fictional stories about the value of 5G in order to get funding no matter the damage.

And no wonder we see so much science hype. Who creates the buzz gets the bucks.

Factoring and new Wholes

Engineers are taught about layered design but that is a very limiting paradigm. This is why I emphasize factoring problems which disentangle the system elements. This is another face of end-to-end.

We need to think in terms of systems but in order to manage the systems we need to be able to find the simplicities. There isn't just one simplicity because simplicity is in the observer. People see patterns in the stars in the sky and create constellations in order to organize their understanding. There isn't a single inherent pattern. The solar system is just a way to organize our view of the planets. Do is the Ptolemaic system but it is hard to work with and we don't get the same insights.

In physical systems the factoring may be more visible but not necessarily obvious. Communities of people may be geographic, but they may also arise out of shared interest among people living in disparate communities.

Perhaps we're used to layered engineering because it's a simple metaphor and, as we've seen with telecommunications, it made sense to treat applications as a layer on the common facilities. But now that we understand software we have an explicit way of talking about systems and factoring and needn't limit ourselves to simplistic layering.

Perhaps this is also why I keep using software examples. It is confusing because when we have a meta system such as a collection of connected pill bottles we tend to think about the mechanics of the "between" – how they send messages to each other. They form a network, so we slip into thinking of the physical network even when they don't even have to send messages to each other.

By having an explicit language for factoring and systems we are ready to move on and understand why we need to create opportunities and who to take advantage of them.

Linking Information

One byproduct of the same forces that gave us the Internet are JSON objects and QR codes. They have their own semantics independent of how they are exchanged – the meaning is in how they are interpreted by cooperating endpoints. In that sense they are very much like we use language.

We can encode an address book entry using standard fields such as first name, last name etc. The problem is how do we know two entries are the same as that we can see that

an address book has been updated in one entry and update it in the other?

Lotus Notes solved this in the 1980s by using GUIDs (yes, GUIDs again) to link entries. If an entry changed it was assigned a new GUID. The GUID histories would then be compared to determine "newer" and to detect conflicts. This is a very simple and fundamental idea, so the question is why isn't this the norm? **Why?**

Note that a word like "name" is contextual so I'm over-simplifying things here. One approach is to use a GUID tag for the bundle so that we don't have semantic-based naming. This is similar to the risk we face using a word like "network" without recognizing that the meaning is contextual.

But we need to somewhere with "every knows". approaches just like we do in the world of paper messages. We can then build more sophisticated systems to handle more edge cases. The key is understanding the concepts.

Just as we need to liberate ourselves from network-first thinking, we need to liberate ourselves from database-first thinking. Today's databases tend to be procrustean whereas such JSON bundles are flexible.

As an aside, this related to why VisiCalc was so successful. The key is it had no inherent knowledge of finance but reflected the knowledge for the users. It was also a departure from the rigidity of databases.

JSON bundles have become the norm for interfaces (APIs or Applications Program Interfaces) because they can sent using HTTP, MQTT, Email or any means rather than depending on a particular layer. Adding the concept Idempotence gives us flexibility and resilience.

One nice thing about JSON bundles is that they are readable by people. While that is not the norm, it gives us transparency which is vital for understanding system behavior.

QR code are closely related and can be passed as JSON bundles or references to bundles. The idea of passing references is, again, something we do every day when using names or SKUs. What is new is having an articular language to build on such concepts.

QR Codes allow for fate sharing by using the same kind of signage we use for people but which a readable by computers. They can contain the text of the message or, more likely, a reference (like a URL).

This is a very deep topic so I can only skim the surface. The important point is that the endpoints can be abstract such as VoIP conversation that move across devices.

Physical metaphors are very useful for understanding the end-to-end concepts, but we need to be wary about limiting ourselves to the accidental properties of such physical metaphors.

For More

I only touched upon some of the ideas here. They are part of my larger effort to explain the concepts. I have other related writings:

Purpose vs. Discovery. which explains why we get effectively unlimited capacity by not locking ourselves into arbitrary requirements.

Refactoring Consumer Electronics. This is really about deriving the Internet from first principles but since I was writing for Consumer Electronics Magazine, I emphasized the CE aspect. This was a fortunate constraint in that refactoring is the key idea. Alas, after writing the piece I came to better appreciate that today's Internet has not yet transcended the framing of classic networking.

Internet Concepts. This is a work-in-progress. So much of the way we talk about the Internet embeds legacy assumptions. No wonder it is so hard to escape the past.

Beyond Limits. This is a chapter I wrote in 1996 to answer the question of what the limits of Moore's Law are. I argued that it is really about market structure and argued the Internet worked because it factored out the relationships for the path (my current wording). It's similar in other markets such as the hardware/software split. And in history railroads provided such opportunities.

Wait, Wait There's More

I plan to have an epilog. There is also so much that didn't fit into the narrative – SystemsNotes.docx has some working notes for future related writings including how to manage families of devices

---

[i] https://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf