From Net Neutrality to Seizing Opportunity

Bits Versus Electrons

From Net Neutrality to Seizing Opportunity

By Bob Frankston

Network neutrality is an important issue. We must not allow transport owners to limit our ability to communicate. But, net neutrality in itself positions the Internet as a telecommunications service. We need to step back and recognize that the Internet itself is part of a larger shift wrought by software.

I thought about this more when I found myself in my hospital room (after knee surgery) unable to open and close the shades by myself. Yet I could control the lights in my house! It wasn't simply that I had built a one-off special case; rather, I had carefully architected my home-lighting control implementation to minimize the interdependencies while taking advantage of existing technologies. For example, to the extent that I could, I avoided depending on the accidental properties of silos, such as Zigbee. I normalized any transport to simple packets. This is how the Internet works; it normalizes the underlying infrastructure to Internet protocol (IP), so I don't care if a particular segment is ATM or cellular. I can use the same technique as in tunneling IP through Bluetooth using the general serial protocols.

Software has given us the ability to stitch things together. In designing and redesigning applications, we have also gained an understanding of the importance of dynamic architectural boundaries that minimize coupling or entanglement.

Digital Object Identifier 10.1109/MCE.2018.2851758
Date of publication: 9 October 2018

Thus, with an IP connection, I could insert shims (also known as *work-arounds*, as long as they preserve the architectural integrity), such as NATs, or treat the entire telecom system as a simple link. I can normalize this by overlaying my own IP connection on top of what I find, including existing IP connections (as we do with VPNs). This allows us to implement and then evolve systems as we improve our understanding.

In the telecom paradigm, I would have to rely on the network to assure a path as a virtual wire from my phone to a device in my house. But in the new paradigm, we have relationships that are abstract. We can represent the relationship as [*a*, *b*], where *a* is the app element and *b* is the device or virtual device. It need not involve a physical wire. The network connection is not a layer but simply one resource I can use. It does require thinking differently and discovering what is possible rather than adhering to rigid requirements. Though I avoid depending on a provider's promises, I may be limited by policies that second-guess what I'm doing. This is why neutrality is an important principle. That includes not doing me favors by second-guessing my needs and thus working at cross-purposes as I innovate outside their design point. A better term is *net indifference*, because the intermediaries don't know my intent and thus can't play favorites.

More important is that it means that paywalls and security barriers may make it impossible for my application to work. I had to manually intervene to use the hospital's Wi-Fi connection. I can do that in simple cases, so we assume that the status quo is fine—but it is a fatal barrier for what we might call *just works connectivity*.

As with any new paradigm, it is difficult to explain, because our very language has implicit assumptions embedded. It also means that often those most expert in network architecture can get lost in their expertise. In the case of the Internet, I see this in the idea of endpoint identifiers being IP addresses assigned by network operators.

As one who takes advantage of the opportunities I find lying around, I view networks as just a means and try to program around limits. If the opportunities aren't available, I can create my own. One example is IP version 6 (v6). V6 makes it easier to produce a direct connection between two end points, but in its absence, I can cobble together a path using IPv4. This is one reason v6 adoption has been slow: we've been able to program around it, so it is nice but not necessary.

Of course, we want to create opportunity. I call one such opportunity *ambient connectivity*: the ability to assume connectivity. This doesn't mean that there is always a way to connect but rather that I separate out the problem of achieving connectivity from how we implement it. It's simple to think of providing connectivity using Wi-Fi, but it's not about Wi-Fi per se, and it's not about a mesh, because it doesn't matter how it's implemented. Those are just examples. It's architecture and not the accidental

Network neutrality is an important issue. We mustn't allow transport owners to limit our ability to communicate. But, NN in itself positions the Internet as a telecommunications service. We need to step back and recognize that the Internet itself is part of a larger shift wrought by software.

I thought about this more when I found myself in my hospital room (after knee surgery) unable to open and close the shades by myself. But yet I could control the lights in my house!

It wasn't simply that I built a one off special case but rather I carefully architected my home lighting control implementation to minimize the inter-dependencies while taking advantage of existing technologies. For example, to the extent I could, I avoided depending on the accidental properties of silos such as Zigbee. I normalized any transport to simple packets. This is how the Internet works – it normalizes the underlying infrastructure to IP, so I don't care if a particular segment is ATM or cellular. I can use the same technique as in tunneling IP through Bluetooth using the general serial protocols.

Software has given us the ability to stitch things together. In designing (and redesigning) applications we have also gained an understanding of the importance of (dynamic) architectural boundaries that minimize coupling (or entanglement). Thus, with an IP connection I could insert shims (AKA work-arounds as long as they preserve the architectural integrity) such as NATs or, indeed, treat the entire telecom system as a simple link. I can normalize this by overlaying my own IP connection on top of what I find, including existing IP connections (as we do with VPNs). This allows us to implement and then evolve systems as we improve our understanding.

In the telecom paradigm I'd have to rely on the network to assure a path from my phone to a device in my house as a virtual wire. But in the new paradigm we have relationships that are abstract. We can represent the relationship "[a, b]" where a is the app element and b is the device (or virtual device). It needn't involve a physical wire. The network connection is not a layer but simply one resource I can use. It does require thinking differently and discovering what is possible rather than having rigid requirements. Though I avoid depending on a provider's promises I may be limited by policies that second-guess what I'm doing. This is why neutrality is an important principle. That includes not doing me favors by second-guessing my needs and thus working at cross purposes as I innovate outside their design point. A better term is "indifference" because the intermediaries don't know my intent and thus can't play favorites.

More important is that it means that paywalls and security barriers may make it impossible for my application to work. I had to manually intervene to use the hospital's WiFi connection. I can do that in simple cases, so we assume that status quo is fine, but it is a fatal barrier for "just works' connectivity.

As with any new paradigm it is difficult to explain because our very language embeds implicit assumptions. It also means that often those most expert in network architecture can get lost in their expertise. In the case of the Internet I see this in the idea of end points identifiers being IP addresses assigned by network operators.

As one who takes advantage of the opportunities I find lying around, I view networks as just a means and try to

program around limits. If the opportunities aren't available, I can create my own. One example is IPv6. V6 would make it easier to make a direct connection between two end points but in its absence, I can cobble together a path using IPv4. This is one reason V6 adoption has been slow – we've been able to program around it, so it is nice but not necessary.

Of course, we do want to create opportunity. One such opportunity I call Ambient Connectivity – the ability to assume connectivity. This doesn't mean there is always a way to connect but rather I separate out the problem of achieving connectivity from how we implement it. It's simple to think of providing connectivity using Wi-Fi but it's not about Wi-Fi per se and it's not about a mesh because it doesn't matter how it's implemented. Those are just examples. It's about architecture and not the accidental properties of radios or wires. It's also about economics; that is, the architectural separation means we can't pay for the infrastructure by setting a price based on the value of the service, because we just see raw bits out of context.

And it's not just about networks. It's about devices that have open interfaces. It's about thinking about devices that can exist for a purpose but also have open interfaces that allow me - or you! - to use them as components. It is about devices and protocols that are smart but not so smart that they build in assumptions. This is why Bluetooth is a problem – it is very tuned to use cases and protocols and limits me to a proximate relationship rather than factoring out distance.

It would be great to have a discussion of this new world that centers around relationships (binding) and software and creating reusable objects. We understand that meaning is not intrinsic. When we sit on a box it becomes a chair. What is new is that we can use software to define (or redefine) what something is. A portable computing device is a telephone in the sense that it can run a telephony application. This is a sharp departure from the notion that a device is a telephone because it was built for that purpose.

Today's Internet is one byproduct in multiple senses. One is that we don't need to build a physical network for one purpose. Another is that we don't depend on networking as a service but simply ask for disparate facilities providers to help packets move ahead, as a highway facilitates driving but doesn't provide the rides themselves —or as we don't apply common carriage to roads because they are inherently neutral in not knowing the drivers' intent. It's why network neutrality is a fine principle but is hard to define once we aren't depending on network providers' services.

It would be great to have a conversation about these big ideas. And part of it is also rethinking the Internet. The particular protocols such as TCP are valuable, but we need to see them in context as means and not as rigid requirements.

The Internet is just one example of what we can create when we have opportunity. Imagine what else we can do given opportunities.

Network neutrality is about networks. We need to move on to just assuming connectivity as mundane infrastructure (https://rmf.vc/BBToInfrastructure). We can then shift our attention how to create and to what we do with the new opportunities