The Internet is about Relationships

October 2015 Issue

Bits Versus Electrons

The Internet Is About Relationships

By Bob Frankston

My last column was about application programming interfaces (APIs). APIs represent a programmatic view of systems. Devices function in a context and in relationship to other devices, but they do not function in isolation.

The Internet is about facilitating these relationships as opposed to traditional networking, which is all about "the between." To realize the benefits of the Internet approach we need to focus on the relationships.

A "wiring diagram" approach defining relationships between devices is complementary to the API approach and gives us a way to visualize relationships. I put the term "wiring diagram" in quotes so we do not confuse it with physical circuits. With a wiring diagram, we can specify the result rather than how it is to be accomplished. A webpage is a powerful example. We link to images and other pages without having to be concerned about the mechanics of fetching and displaying the page.

This ability to focus on the end points of the relationships and not the complexities of what is between those end points is the powerful "magic" that has made today's Internet so transformative. Of course, this is not sorcery but rather defining architecture of the Internet.

RELATIONAL DATABASES

Relational databases show the power of this approach. Early databases were carefully laid out to optimize performance on disks to the point that some databases contained actual disk I/O instructions in the database. The database designer carefully specified a path through the disk drives to reach the data.

Relational databases were very different. The tables of data would be related simply by having common identifiers—index keys. Typically, these were just arbitrary numbers but they could also be names or any common identifiers. This approach not only decoupled the data from the accidental properties of the hardware, but it also flattened the database to simple tables whose relationships were described by schemas.

This allowed databases to be restructured without having to change data values and it also allowed schematic evaluation. A series of SQL queries could be combined symbolically without actually accessing the data, and the resulting query would not need to access all of the original data. One innovation in databases is the use of a globally unique identifier (GUID), which is essentially a long random number that allows disconnected databases to generate their own keys without having a central source of identifiers.

The lessons of relational databases inform our understanding of the Internet and the importance of understanding the use of relationships as an architectural mechanism. I use the term "mechanism" to avoid referring to the relationships as a layer on top of a file system or other implementation. Rather, the file system is used as a resource. Conversely today we have NOSQL databases, which are often defined as "not only SQL." They can use SQL itself as a mechanism.

DESCRIPTIONS AND DIAGRAMS

The web's success owes much to similar principles. Web pages (especially in the days before JavaScript) describe the relationships between pages—the links—without specifying how they are to be implemented. The very term "World Wide Web" builds on this idea.

The pages themselves are descriptions of how the information should be presented without specifying how that is to be accomplished. This is the way we should think about the Internet as a whole—relationships between end points be they devices, webpages or people. Of course, there are no real wires tying these end points together. The relationships exist abstractly or, to put it colloquially, in our imagination. When the time comes to actually follow the link, we may find out it is not valid.

Not only is that OK, it is essentially to the distributed nature of the web. We learn to handle such failures gracefully. Sure a 404 is annoying but far better than crashing your computer.

Too bad the domain naming system (DNS), as with the Heinz example given in the "A Circuitous Path" subsection does not allow for gracefully retiring names.

CONNECTING THINGS!

Today we are increasingly using the Internet to connect things or, perhaps,

Digital Object Identifier 10.1109/MCE.2015.2463413
Date of publication: 29 October 2015

This article is now available here on the IEEE site.. You can see other columns and writings in Further Readings.

Relationships

My last column was about APIs – Application Programming Interfaces. APIs represent a programmatic view of systems. Devices function in a context and in relationship to other devices, but they do not exist in isolation

The Internet is about facilitating these relationships as opposed to traditional networking which is all about the "between". To realize the benefits of the Internet approach we need to focus on the relationships.

A "wiring diagram" approach defining relationships between devices is complementary to the API approach and give us a way to visualize relationships. I put the term "wiring diagram" in quotes so we don't confuse it with physical circuits.

With a wiring diagram we can specify the result rather than how it is to be accomplished. A web page is a powerful example. We link to images and other pages without having to be concerned about the mechanics of fetching and displaying the page.

This ability to focus on the end points of the relationships and not the complexities of what is between those end points is the powerful "magic" that has made today's Internet so transformative.

Of course, this is not sorcery but rather defining architecture of the Internet.

Relational Databases

Relational databases show the power of this approach. Early databases were carefully laid out to optimize performance on disks to the point that some databases contained actual disk I/O instructions in the database. The database designer carefully specified a path through the disk drives to reach the data.

Relational databases were very different. The tables of data would be related simply by having common identifiers – index keys. Typically these were just arbitrary numbers but they could also be names or any common identifiers.

This approach not only decoupled the data from the accidental properties of the hardware it also flattened the database to simple tables whose relationships were described by schemas.

This allowed databases to be restructured without having to change data values and it also allowed schematic evaluation. A series of SQL queries could be combined symbolically without actually accessing the data and the resulting query would not need to access all of the original data.

One innovation in databases is the use of a GUID (Globally Unique Identifier) which is essentially a long random number which allows disconnected databases to generate their own keys without having a central source of identifiers.

The lessons of relational databases inform our understanding of the Internet and the importance of understanding the use of relationships as an architectural mechanism.

I use the term "mechanism" to avoid referring to the relationships as a layer on top of a file system or other implementation. Rather the file system is used as a resource. Conversely today we have NOSQL databases which are often defined as “Not Only SQL”. They can use SQL itself as a mechanism.

Descriptions and Diagrams

The web’s success owes much to similar principles. Web pages (especially in the days before JavaScript) describe the relationships between pages – the links – without specifying how they are to be implemented. The very term “World Wide Web” builds on this idea.

The pages themselves are descriptions of how the information should be presented without specifying how that is to be accomplished. **This is the way we should think about think about the Internet as a whole – relationships between end points be they devices, web pages or people.** Of course there are no real wires tying these end points together. The relationships exist abstractly or, to put it colloquially, in our imagination. When the time comes to actually follow the link we may find out it isn’t valid.

Not only is that OK, it is essentially to the distributed nature of the web. We learn to handle such failures gracefully. Sure a 404 is annoying but far better than crashing your computer.

Too bad the DNS, as with the Heinz example (below), doesn’t allow for gracefully retiring names.

Connecting Things!

Today we are increasingly using the Internet to connect things or, perhaps, an Internet *of* Things. Looking at how we use connected things to control our home can give us insight into connecting things. The simplest example may be a light switch with a wire running to a light fixture. It is very easy to understand what is happening. It also easy to understand how to change the relationships by reconnecting the wire. Easy but inconvenient, and it is difficult to extend that wire very far.

Telephones were also connected by wires and we created an elaborate switching system to extend that wire across the world with operators at switchboards making connections. Later we automated by having pulse dialers driving stepping relays.

We can see the progression from the days when the number you dialed directly stepped relays that setup the circuits to today’s phone system where numbers have become abstract identifiers that translate into circuit paths. That frees the phone numbers to act as relationship links.

In practice this means that people don’t change their (cell) phone numbers when the move around. (I was going to add “within a country” but with T-Mobile and VoIP I no longer worry about getting a local SIM).

A Circuitous Path

We need the same progression in understanding the Internet. The first step is an explicit and articulate understanding of the difference between a phone number as a network identifier and the use of phone numbers as end point identifiers totally apart from their use in laying out a circuit. In practice the phone number gets translated into a separate circuit identifier.

With the Internet the IP address is akin to the circuit identifier. It is used to map the path. This is most obvious in the split between the wide area and local portions of an address. In a 108.26.225.69 address typically the first three octets are the network number and the last octet represents the local address. IPv6 is the same though with a 64 bit prefix and a 64 bit local portion. If you move your device from one local network to another the address changes!

The problem is exacerbated by the use of NATs that enable an entire house to share a single public address. A device with a 192.168.0.1 address isn’t even visible beyond the NAT. Thus you can’t even use the address to find a path between the two end points.

While IPv6 reduces the need for such sharing of public addresses it doesn’t provide stable addresses.

The DNS (Domain Naming System) is a distributed database of the identifiers we use to maintain stable relationships between end points but it is fundamentally flawed.

For one thing the looking up of names requires a connection to the root server at least periodically. The result is the IP address which also requires being issued from a central source if it is to be publically meaningful. If I’m not actively connected to a distant server I can’t even reach my neighbor’s server!

As with the use of Network Address Translators (NATs) to provide local IP addresses there are local name resolution techniques. Of course these share the same limitations as local addresses do.

I run into these limits when I try to use my FiOS mobile app to choose what is displayed on my TV. Often I get told that my video source (set top box) is not reachable. This happens if my controller (smart phone) has connected to the “wrong” network.

There are various ways to work around these problems. IPv6 provides a way to give each device a public address without a NAT. Rather than waiting for IPv6 many programs simply create a tunnel into a cloud service as a way to make themselves available.

Inarticulate work-arounds work as long as you stay within the standard use cases. We can treat a phone number as a stable identifier as long as we take care to port it when we change service providers. That's because we don't really own the phone numbers but instead get them assigned from a provider.

The DNS is supposed to provide the stable identifiers that hold the web (and the rest of the Internet) together. But we don't even own our identifiers. The registries deigning to lease us the names can impose whatever terms they want including setting arbitrary prices. If anything goes wrong including a late payment we lose our very identity?

From 2012 to 2014 Heinz used QR codes to link to sagsmightheinz.de. It then let the name lapse and it was picked up by a porn site! (http://goo.gl/98Wlwi)

There are some extensions that allow for local names but, even then, we might not be able to find the printer next to the computers because it's on the "wrong" network.

We accept such a flawed system because it is reasonable network engineering. Today we need an approach that supports stable relationships rather than just making traditional networks function.

Social!

Let's not forget social relationships. We organize our world in terms of our relationships with other people. Yet when it comes to technology we tend to forget much of what we've learned and try to force people to conform to the technology.

Instead of appreciating the richness of trust relationships we impose a naive notion of security as if could hide behind the castle walls and be safe. We forget that networking is a social activity and view networking as a service as if we had to rely on providers to arrange dates and marriages.

An Articulate Understanding

The use of phone numbers as stable identifiers arose from practice and the DNS was designed as an improvement on the mechanism used to manage IP addresses. Now that we've seen the power of working with relationships we are ready to build on that experience and approach relationships as a mechanism on their own terms.

The relationships are not a layer upon a networking infrastructure. Instead we need to see any available facilities as a resource. This is much like the routes we use to get from point A to point B. That route may only exist in our navigation device. The road "owner" is completely unaware that the route exists and if one road is blocked we choose another path.

Relationships are also abstract – I may have a "cousin" relationship with another person which exists even if I am not aware that I have a cousin.

To keep things simple we can start by focusing on the relationships between physical things with a single connection between a controller (like a light switch) and a destination (such as a light bulb). These relationships exist independent of whatever means we use to exchange messages.

As with the database example we can use self-selected identifiers, GUIDs. A pair of GUIDs defines the relationship as [A, B]. This is purposefully abstract in the same way that an IP packet is decoupled from its meaning – that meaning exists only outside the network. It is extrinsic.

This is sharply different from protocols such as UPnP which have class hierarchies and discovery built into a single mechanism. By tying the mechanisms together we limit ourselves to solving old problems rather than building a basis for the future.

Separating mechanism from policy gives us the ability to learn by doing. For example if we have present a capability token (as with OAuth) we can then experiment with policies to determine who is authorized. One simple way is a name/password or we may choose physical proximity as being sufficient.

Beyond Wires

The wiring diagram approach gives us a concrete way to understand and manage connected things. It's a very good way to approach applications like home control.

The concrete model is a starting point for a deeper understanding.

We can also think more abstractly. URLs, for example, represent connections between documents rather than things. We can also express abstract end points such as "all people named John Smith".

We use these concepts on a regular basis. When we change the association between a DNS name and an IP address we are changing bindings. **Binding** is a technical term and part of our articulate understanding of how to work with relationships.

Another technique is the use of references rather than copying data. We don't make copies of Wikipedia, we just pass URLs. Again, this is not just about computing. Companies are increasingly locating warehouses near the customers and sending around part numbers like ISBN codes rather than shipping the product. Those warehouse are caches for physical objects.

We can also use these abstractions for electronics. Rather than run wires we run a bus and identify the end points using identifiers.

In fact a class Ethernet works just like that – a packet is placed on a common bus and the recipient recognizes its address and reads the packet.

Networks are built upon relationships and we exchange bits between related end points using the networks as a means. This is another reminder we need to think in terms of complementary techniques rather than layers.

APIs too are complementary to relationships. The identifiers we use in APIs require the context of relationships for their meaning. To turn on a light you might specify the GUID of end point representing the light and a GUID for a capability. The API doesn't necessarily know who is turning on the light nor where it is.

Public Policy

Today's telecommunications-based public policies are a poor match for working with relationships. The economics of relationships are decoupled from the costs and value of a given wire.

This is, in a way, like the routes we use in our GPS systems. The value is in the routes rather than a particular road. Fortunately we fund roads as common infrastructure and don't depend on charging people for the directions they get from their devices.

We need a similar approach for the facilities we use to speak among ourselves and for interconnecting our devices. You cannot know whether your relationships depends on navigating the twisting winding passages of telecommunications and whether that path involves a so-called "metered connection" or requires permission.

This is a fundamental difference between path-based telecommunications and a relationship-based Internet.

One of the challenges in public policy is that the very words we use have implicit meaning. I chose to say "speak among ourselves" rather than "communicate" because the sense of the word in telecommunications has a technical meaning that is very different from the day to day meaning in the sense of speech.

Teasing apart these two sense is at the heart of the shift from traditional telecommunications to the new world in which we create our own solutions. Traditional telecommunications policies treat speech as a service from a phone company. With the Internet we speak for ourselves and need an infrastructure to enable us to speak rather than one that assumes speech is provided, at a profit, by a provider.

An articulate understanding of working with relationships makes us appreciate that interconnecting devices in our homes is about more than merely replacing wires – it is about putting us in control and enables us to tap into the power of computing without getting lost in the details of programming.