The ISP Column
A monthly column on things Internet

Geoff Huston
April 2016

IPv6 and the Internet of Things

It has often been claimed that IPv6 and the Internet of Things are strongly aligned, to the extent that claims are made they are mutually reliant. An Internet of Things needs the massively expanded protocol address space that only IPv6 can provide, while IPv6 needs to identify a compelling use case to provide a substantive foundation to justify the additional expenditures associated with a widespread deployment of this new protocol that only the Internet of Things can provide. A good example of this circular line of argument of mutual dependence is found in in this ZDNet report of IPv6's new "Killer App" (http://www.zdnet.com/finally-ipv6s-killer-app-the-internet-of-things-7000027644/)

However, the evidence we have so far with small self-managed device deployments does not provide a compelling justification of this case. The existing deployments of sensor networks, automaton devices, and various other forms of microware are not exactly piling into the IPv6 Internet in their billions. Some large scale device deployments use the mobile cellular data networks, and commonly find IPv4 as a convenient generally available overlay protocol. Others make use of WiFi or a near field technology such as Bluetooth, and its often the case once more that IPv4 is a generally used upper layer communications protocol simply due to its ubiquity. These devices are often built and deployed within a constrained final unit cost budget, so using IPv4 is often a simple pragmatic choice of using what's readily available at very low unit cost. The question here is: Does the Internet of Things require IPv6 as an essential precondition, or are we going to continue to deploy an ever expanding population of micro devices within today's framework of ever increasing address sharing on IPv4?

What's today's picture? Estimates vary wildly, but there is some level of consensus behind a figure of 10 to 15 billion Internet-connected devices in 2016 sitting on the IPv4 Internet. At this time the Internet routes some 2.8 billion unique IPv4 addresses, so it's a logical conclusion that the majority of these connected devices are located behind conventional Network Address Translation (NAT) units, that allow one address to be shared across multiple devices simultaneously. It's evident that the current Things in today's Internet are not reliant on an IPv6 network. But that may not always be the case. What other factors are relevant to the choice of protocol for Things?

Pull vs Push

Part of the debate over this question relates to the nature of the embedded Thing and the way in which it communicates within its external environment.

One model is a "polled model", where the device collects data and retains it in some local memory, and passes the data back to a controller when polled (*pull*). This polled model of data collection requires that the device is the target of connection requests from the broader external environment. This model of communication would conventionally require each polled thing to use its own uniquely assigned public IP address, as the thing is the subject of a rendezvous process, rather than the initiator. Given the sizeable volumes of devices being contemplated in an Internet of Things, this polled model of operation appears to require the larger address family provided by IPv6, as the addressing requirements simply cannot be sustained on IPv4.

When we consider continuous sensor models, such as the video streams of web cams or continuous data flow environmental sensors, and also consider forms of "just in time" opportunistic data collection, then the ability to poll the sensors as and when needed becomes a significant asset for the sensor function. However, exposing an unattended micro device to polling from the Internet has its attendant issues relating to security and abuse. Our experience of such devices, particularly when using open access models, has highlighted the risk of such addressable devices being co-opted into participating in various forms of distributed DOS attacks. While each device may only be coopted into delivering a small data flow, the possibility of orchestrating millions of such vulnerabilities into an attack army can transform such small scale flows into attacks that can reach hundreds of gigabits. The question of whether the larger address space of IPv6 effectively prevents the opportunistic discovery of sensor devices, or whether operational prudence requires that such exposed sensors are equipped with robust security and continual monitoring and maintenance is at present an open issue for the sensor industry. But maybe its not really a fair dichotomy in any case. Prudent advice would indicate that you can run, but you just can't hide, even if you are trying to hide within the vast space of the 64-bit interface identifier. Robust security in the device is required in any case.

So if a thing is a polled thing, and if the polling is opportunistic then the device will need its own public address, and in this case IPv4 is clearly unable to cater to such a model. If this is the paradigm for the Internet of Things than its clear that we can only sustain such an environment within a massive address pool and its here that IPv6 has an undoubted advantage over IPv4.

An alternate sensor reporting model is a "report to base" model, where the device collects data, and periodically initiates a connection to its controller and passes the collected data back to a controller (*push*). This second model functions adequately in both IPv6 and an environment of IPv4 and NATs, as the device initiates connection requests. In the case of IPv4 and NATs the thing is assigned the use of a public address only for the duration of each push connection. At the same time, this second model essentially "hides" the sensor device from the external Internet, as it has no requirement to respond to unsolicited connection requests. NATs are well aligned to this model, as the NAT function prevents external agents from initiating any form of communication with the device.

Much of the work to date in sensor networks and similar application environments for embedded automated devices uses this "report to base" model of connection, which permits the devices to be located behind NATs and use the existing IPv4 network, and, as such, these devices do not add to the impetus for a broad IPv6 deployment.

How Many Addresses?

To answer this question we first need to answer the push vs pull question. In a pull model where things are probed in an essentially open model (think of a device in the same mode as a web server, where anyone can query the server) then there is a strong case that we will need a unique IP address for each thing. Obviously, that won't work in IPv4 for the 10 billion things that we reckon are already out there, and it certainly won't work for the next hundred billion devices either. So if it's a general use of an open pull model of device access then it's IPv6, clearly.

However, the vast majority of things in today's network don't use pull. It's push. And it's push through NATs. One way of looking at NATs is they generate unique address tokens by borrowing additional packet header fields to extend the effective address size. By using a 16-bit TCP and UDP port field then up to 65535 unique address+port combinations can be generated for each IP address. In other words, this approach can expand the unique space from 32 to a theoretical maximum of 48 bits. Currently we use these 48 bits to allow 10 billion devices to communicate. That's 48 bits to encompass 34 bits of devices.

Its not possible to use all 48 bits of the space, as NATs are not associated with every IPv4 address, but clearly we've been able to retrieve 2 additional bits of address space. The NATs can be made significantly more efficient by using the entire 4-tuple of addresses and ports as the lookup key. This creates a 96-bit lookup space, and it may be feasible to contemplate a network of 100 billion connected devices (37 bits) in such an environment.

So as long as the environment is willing to continue to tolerate NATs, and as long as devices tend to use the push model to communicate across the external Internet then the scale of numbers we see in the next few years for the Internet of Things is not any particular cause for alarm. If the industry choses to squash them into 5-tuple NATs over IPv4 then its probably going to be feasible to do so. Of course if would probably be an easier fit to use IPv6, but this is not the only consideration when thinking about IP protocols in this context.

What other factors influence the decision on protocol selection for such devices?

Plug and Play – Auto Configuration

Another part of the debate lies in the division of roles between manufacture and deployment. In the same way MAC addresses are loaded into an Ethernet device at the point of fabrication of the device, it’s tempting to find a mechanism that would load a functional configuration into a device. The device is shipped from the factory in a fully functional state, and in the field it's a simple case of just providing power.

IP does not easily support such a mode of pre-configured operation. A device’s IP address is in effect a location address that allows the Internet’s routing system to understand the location of the address device relative to itself. All that can be placed into the device at the point of manufacture is a unique identity component, and the device needs to “position” itself into its environment and gather an IP address from its local environment once it is powered up. This is the case irrespective of the IP protocol. Both IPv4 and IPv6 require that step of local configuration for the IP address.

While the requirement is the same, the two protocols have a quite different approach to how to achieve the required outcome.

In the IPv4 environment the default bootstrap mechanism is the Dynamic Host Configuration Protocol (DHCP), and outcome of the earlier BOOTP protocol. It is a simple protocol, and readily described. A newly connected device sends a broadcast query on the locally connected network for a DHCP service. DHCP servers on the network respond to the client with a DHCP offer, which includes the details of an offered IP address and, typically, the address of the subnet’s router, the local domain name suffix and the domain name servers, and its own identifier. The client will respond with a broadcast a DHCP request, indicating which server’s offer has been accepted. Any DHCP server that has also made an offer to the device is now able to withdraw its offer and return the IP address to its available pool. The configuration procedure is completed with the server responding with a DHCP acknowledgement. This DHCP simple 4-way exchange is used very widely in the Internet as the default means of power-up device configuration. The advantage of this approach is that a client is provided with its address, subnet mask, default router IP address, subnet domain name suffix and domain name servers in a single DHCP transaction. The disadvantage of this approach is that it is an unauthenticated packet exchange, and its susceptible to rogue DHCP servers intruding on the network. The broadcast nature of the DHCP exchange does expose any attempts to intrude upon the network, so it is challenging to perform such subversion in a stealth mode.

The IPv6 auto configuration environment is different, and there are a number of bootstrap mechanisms. IPv6 Stateless Address Auto Configuration (SLAAC) looks like an obvious choice, as it involves the local routers on a network periodically advertising their address on a dedicated IPv6 multicast address as a Routing Advertisement (RA), part of the IPv4 Neighbour Discovery (ND) framework. A SLAAC node listens on this multicast address, and when it sees an RA it takes its unique local interface identifier and combines that with the 64 bit prefix of the Router’s Advertised IPv6 address in the RA to form a local IPv6 address for this network interface.

Experience has identified a few weaknesses here. Firstly, there is the problem of rogue RAs, where a large open or semi-open network, such as WiFi systems, may include a number of rouge routers that provide bogus Route Advertisements. Of course one could use Secure Neighbour Discovery (SEND) to try and improve this situation, but while SEND can prevent tampering, the original question of authenticity of the RA still remains. Secondly, while the SLAAC system allows a router to generate its local network addresses and install a default route semi-autonomously simply by listening to router advertisements, the system does not necessarily include domain name configuration data, and that still may need a separate configuration step. So this introduces the third problem: choice. There are a variety of ways to ways to perform the full configuration in IPv6. The first is to complete the stateless auto configuration with a two additional options in IPv6 Neighbor Discovery (ND). This would allow a client to discover both a router and the DNS configuration in the periodic Router Advertisements. Authenticity is still an issue here and the additional options really don't help the basic question of whether an observed RA is authentic or not. Another approach is to complement SLAAC with a separate configuration step using Stateless DHCP for IPv6, using the DHCPv6 step to load the DNS parameters (default local domain name suffix and local recursive resolvers). The change in DHCP here is to shift from the IPv4 broadcast model to an IPv6 multicast model to solicit a DHCPv6 server to respond with the requested information. Another choice is to eschew SLAAC and use DHCPv6 for the entirety of the auto configuration process, using DHCPv6 in a manner that is more directly analogous to DHCP in IPv4 providing both an IPv6 address lease and DNS data.

Another question is whether this thing that is attempting to auto configure is a unitary device or is itself a small network of things. For example, a modern car evidently has a few hundred processor systems, and it's probably better to think of the car as a wannabe mobile LAN or even a collection of discrete LANs[1] rather than a unitary self propelling mobile device in networking terms. If that's the case then it might make sense to use DHCPv6 in Prefix Delegation mode and assign an entire IPv6 prefix to the requesting device and then allow that device to control its own local subnets. And then of course there are the possibilities of using a dual stack environment and complement the IPv6 prefix and routing configuration of SLAAC with a DNS context provided by DHCP in IPv4.

So there is a choice of configuration options in IPv6, and there is no single assuredly available mechanism. The problem here is just that: choice. In IPv4 it's DHCP. In IPv6 it's… complicated (to borrow a Facebook term). The combination of SLAAC and Stateless DHCPv6 for DNS makes some sense, but you cannot be assured that is what you will always get in the field. A IPv6 device needs to be flexible in its use of auto-configuration modes and be capable of interoperating with SLAAC and ND, SEND, Stateless DHCPv6, or DHCPv6 PD. It's complicated.

Security

The security models for the two protocols are probably no different. Sure, an IPv4 device is probably located behind a NAT and cannot be readily polled and probed by the onslaught of IPv4 address scanners. But that being behind a NAT does not mean that it should place any trust in the benign intentions of its neighbours on the local private network. A compromised host is a potential source of attack, and this holds whether the device is behind a NAT or not. A far safer assumption is that all external interfaces are a source of attack, irrespective of whether the transport used in the attack is IPv4 or IPv6. So NATs does really make life any easier in this respect, or any harder for that matter.

For IPv6, hiding behind the law of very large numbers in no defense for anything other than cryptography (and even then its not clear that we can hide there forever). Its clear that the efficient all-of-the address-space scanners can't work in IPv6 in the same way that they operate across the IPv4 address space, and opportunistic discovery of a device through its public IPv6 address is statistically

[1] The story of discrete LANs is relevant to the story of vulnerability analysis of such complex devices. It was reported that the entertainment system was successfully used to penetrate into a vehicle's driving control systems. The logical response is to consider a vehicle as a collection of distinct security domains, which may in turn lead to the view of the vehicle as a set of distinct networks. (https://www.schneier.com/blog/archives/2015/07/remotely_hackin.html)

unlikely, but addresses leak in a myriad of ways, and every device should assume that it will receive hostile traffic from its network interfaces.

There are a few basic tenets of secure operation for the things that populate the Internet of Things are somewhat protocol agnostic.

All of these unmanaged devices need a way to phone home, and do so in a reliable and trustable manner. "Home" needs to be able to upgrade the device in a fully unattended manner, or in the final case, needs to be able to shut the device down, or otherwise close it down into a safe state.

All devices need to be paranoid. Trust is the outcome of negotiation, and obscurity is a lousy substitute for an effective security framework. Whether the device is connected on IPv4 or IPv6, or both, it should assume that it will receive hostile or malicious traffic, and it needs to be able to do distinguish between friend and foe. They should only respond to transactions that they can authenticate, and respond in ways that cannot be perverted or otherwise warped into an attack vector on others.

The sad fact is that today's network is badly compromised, and at the heart of many of today's most vicious attacks are millions of "things". Populating the Internet with more of them will not make this problem go away. Quite the opposite. There is a distinct risk that we will create a network that is unsustainably toxic.

What do Things Need? IPv4? Or IPv6?

Millions of years of biological evolution teaches us a major lesson here - they key to sustainability of complex systems appears to include flexibility and adaptability. Things need to be able to sense and adapt to their environment and fulfill their intended function using the means are available to them.

In many ways this has already been the path of computing over the past few decades. The original models of computing were rigid and oriented towards an interface that matched the simple systems in the computer. If anyone still remembers IBM's JCL, its pretty clear that JCL forced the user to adapt to the limitations of the system. Subsequent evolution of the environment has produced systems that have a natural and elastic interface for humans. These days it is possible to direct a system through a raised eyebrow, if that's what you really want! This evolution is not only increasing elasticity and adaptability in the user interface, but also adding elasticity and adaptability into the computer and network interfaces that we should make use of.

What does this mean for the Internet of Things?

There is no point waiting for a ubiquitous IPv6 substrate to support your thing.

Equally, there is no point deploying a thing that is only equipped with IPv4 these days.

For the moment, the most pragmatic answer to the question of what IP protocol to load into a thing is: both!

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990's. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001 and chaired a number of IETF Working Groups. He has worked as an Internet researcher, as an ISP systems architect and a network operator at various times.

www.potaroo.net

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.