Implementing IPv6 at ARIN

Matt Ryanczak

ARIN began implementing IPv6 in 2003 and finished enabling most systems and services in 2008. Today all new networks and services are designed with IPv6 in mind. What follows is a time line of events and an experience report on how we accomplished our IPv6 deployment.

The Timeline

Research on how to best implement IPv6 on our network began in 2002 with the goal of having something in production in 2003. Our production deployment began in 2003 with a single IPv6-only T1 to our Chantilly, Virginia office from Sprint. We began by hosting HTTP (www.arin.net), FTP (ftp.arin.net), and DNS (in-addr.arpa and ARIN zones), as well as a server doing secondary DNS for other regional Internet registries. All hosts ran Linux, we used Apache for HTTP, ProFTPd for FTP and Bind for DNS. The network initially used a Linux box with a Sangoma card as the router and firewall. The firewall functionality was eventually moved to an OpenBSD host because its IPv6 support was better at the time. Linux lacked stateful packet inspection in ip6tables which made maintaining a good security policy difficult. However, using a Linux box as a router proved to be great for troubleshooting because having tools such as tcpdump directly on the routing platform was very convenient. This IPv6-only network was completely separate from the primary ARIN network because we had security and stability concerns. A positive side effect of our paranoia was that the segregation allowed us to experiment without worrying about negative affects on our primary IPv4-only network. All hosts had static IPv6 addresses and the default gateway used simple static routes to forward packets.

In 2004, we built a new network at our new co-location facility, Equinix, using a circuit from Worldcom. Our goal was to add redundancy to our IPv6 services, try new technologies and continue to expand our knowledge. Much like the first IPv6 network, this network also used an IPv6-only T1 and offered the same services. This T1 was provided by Vint Cerf's office of research at Worldcom and was not a commercial offering at the time. This network used a Cisco 2800 router instead of a Linux box. We needed to make sure that IPv6 would work on our standard routers and we used this network as the opportunity to test them. We continued to use an OpenBSD firewall because its IPv6 support was very good, surpassing all commercial offerings in this area at the time. OpenBSD was also used on our IPv4 networks which meant that maintaining a consistent security policy was easier. This network was still a segregated, stand-alone network and used static IP addresses and routing. It was not connected to the Sprint IPv6 network.

In 2006, Equinix began testing an IPv6-only Internet Exchange (IX) called Equi6IX. An ARIN member and Equinix employee invited us to participate and we replaced the Worldcom circuit with the connection to Equi6IX. Transit was provided by OCCAID, a non-profit research organization. This gave us 100MB/s IPv6 transit without tunnels and our first opportunity to use BGP as our exterior routing protocol. We also began peering over IPv6 with anyone that was willing. Some of our first peers were Tiscali and NTT. We have since added more peers and have gone to a default free peering model with multiple peers. We used a Cisco 2800 router, an OpenBSD firewall, and hosted WWW, FTP, DNS and eventually added MX services using Postfix. All hosts had static IPv6 addresses and we used OSPF as our internal routing protocol.

Once this network was complete we began to experiment with dual-stacked hosts with an interest in bringing IPv6 to our entire network including the desktops of ARIN employees. In 2008 we extended the IPv6 backbone from the Equi6IX network to bring IPv6 to the ARIN offices in a dual stack configuration. The Sprint link was limited to DNS by this time, and was on its way out of production use. The Sprint circuit was retired in 2009.

In 2008, ARIN began to build two additional stub networks to host ARIN public-facing services. The general idea was to split public-facing services such as HTTP, FTP, DNS, WHOIS, etc. from the back-end provisioning systems (database, e-mail, financials, etc). From the start these networks were intended to be dual-stacked and all services would be provided over both IPv4 and IPv6. ARIN published an RFP for co-location and network services which stipulated that native IPv6 support was required for a successful bid. This was the first time ARIN had made IPv6 a hard requirement for doing business with a vendor. With these networks, ARIN had parity between IPv4 and IPv6 in terms of services provided (HTTP, SMTP, FTP, DNS, WHOIS, IRR, RWhois, etc).

These networks also peered with hosts using 4-byte ASNs. We heard a lot of trepidation about peering with us using 4-byte ASNs; a lot of this fear was based not on lack of router support but on lack of support from provisioning and accounting systems. Some networks, Tinet for example, used our use of 4-byte ASNs as an opportunity to upgrade their environment and peer with us.

Since 2008 we have continued to enable IPv6 for services and devices where support was lacking before or where we have found suitable work-arounds. For example in the spring of 2010 we deployed a new version of ARIN whois server with IPv6 support built in. Our monitoring and management infrastructure has seen a lot of improvements in terms of IPv6 support as well. Many of our push scripts work over IPv6 now. We also actively monitor hosts and services over IPv6 using Nagios and custom scripts. IPv6 has become part of our daily routine at ARIN. We still find software that does not support IPv6 well or does not support it at all but we have developed a toolbox that helps us cope with these problems as they come up. We have not found an IPv6 related problem we cannot solve in quite some time.

In the summer of 2010 we deployed a new IPv6 enabled network in Toronto, Ontario. Unfortunately, it took several months after the initial deployment to get IPv6 transit because there was a lack of IPv6 on our provider's network. We addressed this by joining the Toronto Internet exchange and peering with IPv6 capable networks. In the fall of 2010 we will be deploying a network somewhere in the Caribbean and plan to obtain IPv6 through peering relationships as well. We expect this network to be a test of whether we can get IPv6 everywhere we need it to be.

Lessons Learned

We've learned some valuable lessons throughout the process of enabling IPv6 on our networks. Some of the problems were expected while some of them were completely unexpected. All of the lessons learned were valuable no matter how small or seemingly insignificant.

Tunnels can cause breakage by causing sudden changes in maximum transition unit (MTU). IPv6 has built-in facilities to address these problems but if firewall rules filter ICMPv6 to hosts or devices do not implement IPv6 path MTU discovery (PMTUD) properly you will find that packets mysteriously disappear in transit. We had a lot of PMTUD problems with our early IPv6 networks. We learned that making sure firewall rules and router ACLs allow the proper types of ICMPv6 is critical to IPv6 functionality. We also learned that in the worst case, if you cannot find where packets are being dropped or you have no control over the network were loss is occurring there is something you can do. Lowering the MTU of the effected host(s) to 1280, the minimum for IPv6 will help in almost all cases.

IPv6 transit is much like IPv4 transit was 15 years ago: some providers are terrible while others are excellent. It can be difficult to tell if a provider has good IPv6 support on their network but there are some questions you can ask that will help. Asking the provider if they have a separate backbone or if they use their primary backbone for IPv6 is a good place to start. Knowing whether a provider uses tunnels can illuminate the fact that they do not have native IPv6 support on parts of their network. Ask any potential providers if they offer the same SLA for IPv6 service as for other services. If they do not you should be cautious. Another lesson learned about transit providers is that sometimes the big providers in your area may not be the best. It is worth talking to the smaller providers, you may find that they have better IPv6 support.

Routing, like transit, is not as robust on the IPv6 Internet as it is on IPv4. Things are more stable now than when ARIN started but we still see more route flapping over IPv6 than we do over IPv4. This is because the IPv6 Internet is still growing, new networks are coming on line, many people are experimenting and others are still learning how to make IPv6 work for them. This means that routing can be less stable. In the early days of IPv6 at ARIN we would see entire enterprises or even countries disappear for days at a time. Getting a full BGP feed from your IPv6 provider can help you understand why routes are disappearing, it could be that the destination you are trying to reach has gone away.

Dual stack, that is assigning IPv6 and IPv4 addresses to hosts simultaneously, makes deploying IPv6 much easier on a pre-existing network. One reason this is the case is that you can implement IPv6 in a more piecemeal fashion, enabling it on a limited number of hosts and network segments at a time which helps make the work load more manageable. There are other reasons as well though. For instance if you have networks which run DHCPv4 you need not necessarily worry about DHCPv6 support and instead implement only IPv6 router advertisements. This gives you IPv6 address auto-configuration without having to worry about DHCPv6 support in clients which is limited at this time. In fact only Windows 7 and Solaris 10 support DHCPv6 out of the box. Other operating systems such as Linux, OSX and the BSDs can support DHCPv6 but they require installing and configuring additional software. A related benefit of using router advertisements on an existing DHCPv4 enabled network is that you can continue to use hosts and equipment that do not support IPv6 completely or at all. Windows XP is a good example of this. It has limited support of IPv6, it supports address auto-configuration but can not do DNS lookups over IPv6. If you dual stack Windows XP hosts you will find that they can do DNS lookups over IPv4 and still connect to IPv6 hosts on the network. Legacy devices and hosts that do not support IPv6 at all can still be accessed by dual-stack hosts without requiring proxy servers or other transition mechanisms. There are many advantages to using the dual-stack approach to enabling IPv6.

There are also some down sides to dual-stacking. Maintaining a good security policy is more difficult. You have to worry about firewall rules and access control lists for two protocols rather than one. You can also introduce breakage when dual-stacking hosts. We had to manually run many automated push scripts that relied on SSH for instance. SSH, like many applications, prefer IPv6 if it is available. This caused many scripts to hang on the host authentication portion of the SSH login because they had never connected to the target hosts using IPv6 before. Overall security is still a concern, a dual-stacked host has a larger footprint in which to attack, but we have found that maintaining good access controls and understanding how IPv6 differs from IPv4 really helps to maintain a good security posture.

There are many tools that can help you transition to IPv6. We have had to add IPv6 support for old applications that did not have IPv6 support at all. The legacy Whois service is an example of this. We wanted IPv6 only hosts to have the ability to use our Whois service but the software did not support listening on an IPv6 address. We solved this problem by using 6tunnel to proxy TCP port 43 on IPv6 to TCP port 43 on IPv4. This solved our problem and while it did add some overhead it allowed IPv6 hosts to use Whois until replacement was ready early in 2010. The Apache HTTPd server is another example of how the application of a proxy can help IPv6 enable legacy applications. Using Apache's mod_proxy you can add IPv6 support to just about any HTTP based appliance or application. If you have embedded appliances that use a web based interface and you need them to be available over IPv6 Apache can help you do this.

Many open source tools support IPv6 very well. Wireshark, tcpdump, mtr and many other network diagnostic tools support IPv6. It is easy to monitor IPv6 hosts using Nagios along with Perl, Python or even BASH. Generally, Unix based tools have good IPv6 support and if you have a library of scripts in Perl, Python or other scripting languages there is a good chance IPv6 support is there. You may need to tweak REGEX and command line options but IPv6 support does exist. The down side is that if you have a large library of scripts like we do it can be labor intensive to update everything. We took advantage of the dual-stack approach to IPv6 enable scripts only when we really needed to. As scripts were replaced IPv6 was the default option. Today we still have a mixture of scripts that use one protocol or the other.

We also found that your vendors will listen to you. If you need IPv6 support from your ISP, keep harassing them about it. You may have to talk to six or seven people but eventually you will find someone that knows what they are doing with IPv6 and can help you get service. The same is true for hardware vendors as we had several vendors provide us with updated firmware based on our IPv6 requirements. We continue to work with some of our vendors on improving the IPv6 support in their products. While it can be difficult and time consuming to work with vendors in this way we have been rewarded with equipment that is constantly improving in regards to IPv6 support. In 2008 we started requiring IPv6 support on all requests for proposals and it is not apparent that this has hurt our ability to find vendors. The IPv6 requirement has not had any negative effect on our ability to find vendors willing to work with us.

With the available free pool of IPv4 address space decreasing, it is clear that Internet growth can only be sustained by adopting IPv6. Making the appropriate changes inside a network and coordinating those changes with your Internet service provider does require work, but odds are the equipment and operating systems you have already offer the support you need. Enabling IPv6 on your network is not as hard as you may think. Hopefully this report has given you some idea of how to enable your network. Interest in IPv6 is exploding, there is a wealth of information available on the Internet to help guide you and many people are taking their first steps into IPv6 which makes now a great time to get started.

ARIN is committed to help smooth IPv6 adoption, taking into account the contributions of all the organizations that have sustained the Internet through its successful history and maintaining the innovative spirit of the Internet into its future.