IPv6 and Applications

Abstract

The deployment of IPv6 requires new strategies, practices, tools, and applications. As IPv6 moves toward full industry adoption - and as it becomes the primary foundation for next-generation networks - it introduces complexity that falls outside of the current network infrastructure focus. One area of this complexity that has had little attention is network applications. Crucial to the success of any IPv6 rollout will be the requisite modifications of applications that **both use and support IPv6**.

Introduction and Assumptions

There are many drivers and stakeholders for IPv6 adoption abound, but the most visible message has been that of imminent "IPv4 address exhaustion". Because most of the press surrounding early IPv6 adoption has focused on address exhaustion, application and network developers have incorrectly assumed that the effect of IPv6 adoption in their domains will be insignificant. On the contrary, older networks, services, and applications will be affected, as well as new applications development related to IPv6 adoption, such as:

- The 128-bit IPv6 address vs. the IPv4 32-bit address may affect applications.
- Formats and data structures for address support are different; for example, canonical form vs. noncanonical or flexible form.
- Hierarchical addressing in IPv6 introduces issues with some applications.
- IPv4 broadcast addresses will be eliminated.
- IPv6 anycast addresses will be introduced.

For context of this article, "applications" are defined as somewhat generic in nature. This article addresses:

- Network applications and services that use IPv6
- Applications that manage IPv6 or applications that IPv6 uses (nodes, routers, services, etc.)

This paper assumes a migratory and evolutionary path for compliance of applications moving from IPv4 to IPv6 that closely follows the actual strategy, development, and deployment of IPv6 networks themselves:

→ Dual-stack transition mechanism (DSTM) solutions initially give way to:

→ Translation solutions through Application-Level Gateways (ALGs) or protocol translators that lead to:

→ Native IPv6 applications, and ultimately the goal of:

→ IP-agnostic applications for network services and management

The stated assumption that DSTM will be the initial primary means of transitioning from IPv4 to IPv6 is realistic. What is not realistic is the statement that has been in print that if DSTM is used, then applications do not require any modification. This article provides ample evidence that applications should be in focus early in the IPv6 migration lifecycle. For the sake of argument, most IPv4 applications will work in any DSTM environment, though the cost is high because each node must support DSTM with an initial heavy use of tunnels.

It is assumed that the information provided in this article should be used early in the development of any IPv6 strategy and design and should not be relegated to later efforts.

Applications Used for IPv6 Management

This category of applications is specific to the **management** of IPv6. There appears to be a common and tacit assumption that the only change required in management applications is to simply accommodate the increased address space by increasing the database schema and data structures. Nothing could be further from the truth. Following are topics that address the shortcomings of this assumption and those that explain the complexities actually required to modify management applications.

NATs and ALGs

Regardless of your perception of the value of Network Address translations (NATs) and ALGs as solutions to perpetuate IPv4 infrastructures, reality dictates we assume these “solutions” will be permanent network fixtures throughout the lifecycle of any IPv6 transition process. Following are critical areas that developers need to be aware of as IPv6 is rolled out:

- ALGs perform protocol translation services and proxy services at the application layer. The applications can be required to make calls or the services can be transparent.
- NAT was created to handle IPv4 addressing constraints. NAT in its purest form simply translates one address type, or pool, to another.
- NATs and ALGs fall into the category of network applications that use embedded IP addresses or port numbers as elements of their structured packet and protocol design as well as their ultimate solution delivery. These types of applications will be the most difficult network applications to transition to IPv6 because a major rewrite of the code base, or additional ALG behavior, is added in the interim and transition solution phases.

DNS Concerns for IPv6

Domain Name System (DNS) is the largest global database upon which all network-based applications critically depend. We have already stated in the “Introduction and Assumptions” section that IPv4-to-IPv6 transition mechanisms of some sort will be in place for years in most networks. Given that DNS currently provides critical naming to IP-address services, DNS becomes even more critical in every respect during the entire transition period, and not just as part of the ultimate solution.

Simply upgrading to the latest version of BIND that provides IPv6 support does not solve the problems surrounding DNS. Actual DNS server(s) design and implementation play a critical role in the success of any IPv6 rollout. Tunnel design and termination must account for DNS access and placement as well. Because this article does not focus on tactical IPv6 network design problems, you should perform additional research into the best current practices (BCPs) provided by various vendors (listed in the reference section), and you should adopt as much as possible any reasoned and articulate recommendations. Complicating matters are vendors’ implementations of DNS and naming services that contain a great amount of proprietary code and requirements. Most code bases will require research and detailed analysis to address migration to native IPv6 support.

Migration to native and transition IPv6 infrastructures and network services affects both client and server use of DNS. With regard to a generic context or perspective, the application initially queries DNS for the default IP protocol; when the application finds that protocol, it returns the specific record format. Failing that query, the application fails over to the alternate record format. For example, Microsoft's Vista assumes IPv6 is on and is used by default. All initial queries are made for, and are expecting, authentication, authorization, and accounting (AAA) records (IPv6 records), not A records (IPv4 records). If the IPv6 network is correctly designed and all necessary network services and applications are appropriately located in the IPv6/IPv4 network, then IPv6 DNS will always be the preferred solution.

Another problem that can affect application delivery focuses on the interaction between the resource records and the host name component. Also critical to the success of any DSTM design is the fact that the DSTM itself must be able to support receipt of both A and AAAA (quad A) resource records (32-bit IPv4 addresses and 128-bit IPv6 addresses, respectively) in its own use of DNS and must conform its processes to the returned record format to any requester. In the case of A and AAAA record returns for the same query (name), the DSTM application must be able to identify by policy or configuration which is the preferred IP and then resolve the query based on that policy or configuration tuple.

You should address the following recommendations and concerns early in the applications development, porting, or design to support IPv6 fully and well:

- The native application behavior must discern if it is talking to an IPv4, an IPv6, or an IPv4/IPv6 transition node. When that determination is made, the application needs to implement the appropriate policies or configuration behavior.
- In order to maintain "A" records for backward compatibility throughout the dual-stack transition, the applications developers should lead the strategy to ensure the transition application solutions are not the final ones implemented for the specific applications; eventual native IPv6 behavior should be the target.
- You must determine whether older applications, and their owners, provide enough documentation to ascertain how embedded the IP addressing is in the application and also determine the proprietary schemes used to support IP. As we all learned in the Y2K debacle, much application design and code information is either lost or it left with the developer(s) responsible for its creation.
- You must ensure that the latest version of BIND is always implemented and well-tested.
- You must determine the following: Is DNS hierarchy fully documented for the transition period? Is there a similar documented strategy for post-transition? Does the IPv6 management strategy account for the integration of location (and all other factors) into DNS, or are the strategy, architecture, and solution just a short-term fix?

As with any of these presented steps and strategies, their order of execution is critical to the ultimate success of the applications. Because IPv6 is relatively new (in terms of architects, engineers, and skill sets of support staff), very few BCPs are available for common reference.

- Upgrade DNS server software to an IPv6-compatible version.
- Ensure that all IPv6-specific OS functions are well-understood and implemented by the applications and DNS.
- Add dual-stack.
- Add IPv6 addresses to DNS records.

- Use tunnels (one method) to connect the IPv6 islands through IPv4 transport.
- When applicable, remove IPv4 support.
- Use translation to continue IPv4 support as part of the migration strategy.

Comprehensive details about the effects of IPv6 on DNS use and implementations are out of the scope of this article. Multiple “add-ons” for controlling or modifying DNS behavior have appeared in the IPv6 solutions sets over the years. Some are already obsolete, and others have been implemented in various vendors' solutions.

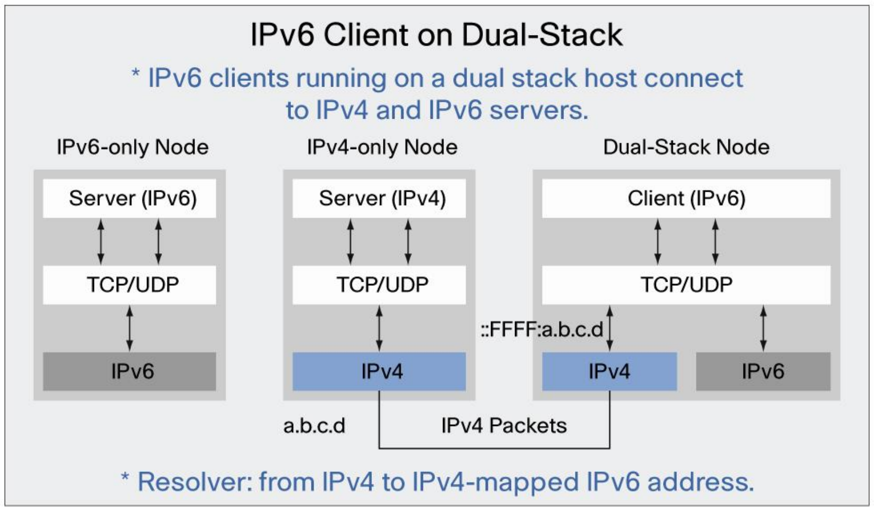
DHCPv6

Although Dynamic Host Configuration Protocol (DHCP) is critical to the success of using any IP network application, it is even more so as IPv6 is implemented in a transition lifecycle. You must consider the design from both the application and the network perspectives to ensure that the correct behavior is ultimately returned and implemented.

As in DNS, the assumption of a DSTM solution as an initial “de facto” solution is the focal point for our consideration. The actual application providing DHCP use must be modified for DSTM and IPv6. The use of DHCP is far simpler in the transition networks than that respective to DNS, however. Each IP “stack” in the DSTM (IPv4 and IPv6) mechanism will use its own version of DHCP. DHCP is used for the IPv4 stack and DHCPv6 is used for the IPv6 stack.

Each DHCP solution will return additional information with each address query result such as the preferred Network Time Protocol (NTP) and DNS preferred choice. Additional configuration information can also be included. Because both stacks are receiving this additional configuration information, it is incumbent upon the application developer to ensure that the “merged” data from each stack is prioritized and used correctly.

Figure 1. Single and Dual-stack Clients


IPv6 Client on Dual-Stack
* IPv6 clients running on a dual stack host connect to IPv4 and IPv6 servers.
IPv6-only Node
IPv4-only Node
Dual-Stack Node
Server (IPv6)
Server (IPv4)
Client (IPv6)
TCP/UDP
TCP/UDP
TCP/UDP
::FFFF:a.b.c.d
IPv6
IPv4
IPv4
IPv6
a.b.c.d
IPv4 Packets
* Resolver: from IPv4 to IPv4-mapped IPv6 address.

IPv6 Network Management Applications

DNS, DHCP, and FTP are not the only set members of IPv6 applications requiring attention. Any network management application will require modification or rewrite to fully support and manage IPv6 with parity against the IPv4 solutions that might be available today. The problem is not addressed by simply modifying existing IPv4 applications with different data structures. The following should help guide your efforts when addressing possible changes to network management applications:

- Add IPv6 stack support to the network management system (NMS): This support is paramount to implementing the DSTM mentioned earlier. This effort is neither trivial nor automated. Each implementation of an IPv6 stack along side of an IPv4 stack will require substantial effort and will have proprietary components.
- Add IPv6 stacks to managed devices: The addition of IPv6 instrumentation and IPv6 support in network elements must coincide with that in the NMS itself. Many vendors do not have long-term solutions for IPv6 instrumentation in the managed elements, using rather short-term, or point, solutions such as discrete MIBs rather than unified MIBs. The application migration must account for this reality.
- Add full Simple Network Management Protocol Version 3 (SNMPv3) support to the network over IPv4 but supporting IPv6 MIBs.
- Move NMS applications over the IPv6 stack: We assume, of course, that IPv6 is now native in the infrastructure or that tunnel mechanisms are in place to transport IPv6 through the IPv4 networks.
- Add SNMP support over IPv6 transport: SNMPv3 supports IPv6 but needs to be transported itself “over” IPv6 instead of just “managing” IPv6.

Applications That “Use” IPv6

Applications that use IPv6 are as varied in design as the mechanisms they deploy internally to accommodate the IP transport. Most, if not all, IPv4 applications that use IP for service delivery fall into three categories:

- Unicast applications: These applications are best categorized as a one-to-one service delivery model.
- Broadcast applications: These applications fall into the one-to-all service delivery model.
- Multicast applications: These applications use IP for a one-to-many service delivery model.

IPv6 radically changes the IPv4 service delivery models by eliminating the broadcast paradigm and introducing the anycast paradigm. Anycast allows delivery to the “closest” (relative use) node of a group. Address mechanisms for the transition period for any IPv6 network follow:

Porting Broadcast Applications to IPv6

Many IPv4 applications use broadcast either in replacement of, or in conjunction with, Unicast addressing for service delivery. IPv6 architecture has both eliminated broadcast capabilities and introduced a new paradigm, anycast.

Consider a pragmatic example from Microsoft architecture addressing “how” to port an IPv4 application that uses broadcast to its IPv6 equivalent. We will use Windows Sockets as our example, but the references here, and the concepts, hold well for any UNIX variant as well. Because, from the perspective of an application, IPv6 Multicast can emulate the older IPv4 broadcast capabilities, our example will reflect how to change an IPv4 broadcast application to an equally useful IPv6 Multicast scenario.

The IPv4 operationally equivalent approach is to set the IPv6_ADD_MEMBERSHIP socket option with an IPv6 address set to link-local scope all nodes address (FF02::1). Using this address is functionally the same as using the IPv4 SO _ BROADCAST socket option. This approach is referred to as the “all-nodes multicast group”. One critical fact that applications developers should keep in mind is that IPv6 multicasts using the all-nodes multicast group address are not the default. IPv4 broadcasts are received by default. The application developer must use this IPv6_ADD_MEMBERSHIP socket option to enable multicast reception from any sources. The application developer also must be aware that in IPv6 (Microsoft’s), interface selection is specified in the argument structure passed to the multicast socket option or IOCTL. We address the IOCTL use later in this document.

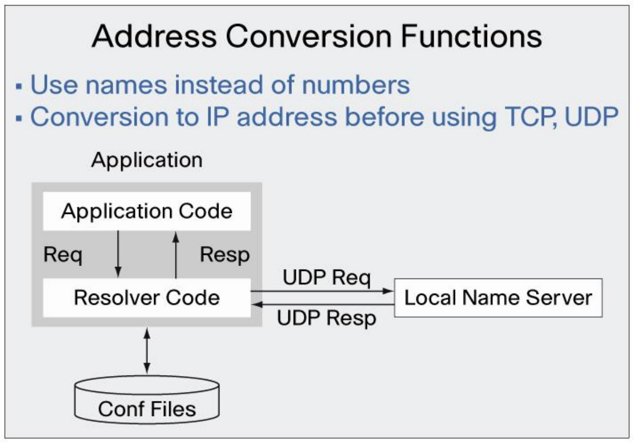
As is the case for most vendors, Microsoft has internal BCPs and strategies that it bills as the best approach. One of them is a recommendation that application developers, when porting to IPv6, learn to enjoy the tremendous opportunities available from the use of “selective multicast”. Actually this approach adds more overhead to the application and more complexity to the solution overall, but given the fact that IPv6 no longer supports broadcast mechanisms, the application developer’s choices are limited.

Using Multicast Listener Discovery (MLD) (part of IPv6) and the Internet Group Management Protocol Version 3 (IGMPv3), the application developer can migrate IPv4 broadcast applications to IPv6. This approach actually offers more features and functions than did IPv4 broadcast, such as enabling applications to specifically block (or unblock) senders within a group that could be considered a security risk.

There are two primary scenarios for application programmers using multicast: those porting from IPv4 broadcast (or multicast) applications to IPv6, and those creating new IPv6 Multicast applications. For the porting of existing applications, there are two options to move to IPv6 Multicast: using socket options and using IOCTLs.

- Using socket options is a change-based approach, which enables developers to change the socket properties as required (such as blocking or unblocking a sender, adding a new source, and so on). This approach is more intuitive and the recommended approach. For more information about the change-based approach to multicast programming, please visit MLD and IGMPU Using Windows Sockets.
- Using IOCTLs is a final-state-based approach, because it enables developers to provide a fully configured socket state, including inclusion and exclusion lists, with one call. For more information about the final-state-based approach, refer to Final-State-Based Multicast Programming. For those creating new IPv6 Multicast applications, the recommended practice is to use socket options rather than using IOCTLs.

Figure 2. Address Conversion


Address Conversion Functions
▪ Use names instead of numbers
▪ Conversion to IP address before using TCP, UDP
Application
Application Code
Req
Resp
Resolver Code
UDP Req
UDP Resp
Local Name Server
Conf Files

General Guidelines and Recommendations

The following are provided as guidelines for developers to consider when creating the IPv6 strategy. These guidelines should map to specific tactical steps in the IPv6 application development and deployment processes:

- Validate and ensure that the applications are using correct data structures. Because IPv6 uses the 128-bit address format (and other changes if dealing with IP-agnostic data structures), research OS-specific changes identifying socket call and storage data structure recommendations at the OS level. Implement these changes (and proposed changes) in the lifecycle of the applications.
- Investigate and determine if there are any "IP-agnostic" sockets, data structures, and functions for immediate applications use or determine what roadmap and timeline is being stated for such. Map these results to specific vendors of choice and those that are in consideration for use and glean their IPv6 support strategy. Keep the applications development strategy synchronized with the evolution of the IPv6 application calls.
- Develop policies and BCPs that address using new IPv6-oriented functions as soon as they are available instead of IPv4 ones.
- Address the topic of IPv4 constants. Protocol- and address-specific constants are intended here.
- Develop an application development strategy that moves name calls to the newer IPv6 "gethostb ynamev6" calls (for example).
- Because most applications use a character string format for an IPv6 address somewhere in the application lifecycle, research and replace the older IPv4 functions with those that are IP-agnostic.
- Research and implement any IPv6-specific macros and libraries for advanced IPv6 features and functions that can be under the control of the applications.
- Because most older applications store IPv4 addresses as 32-bit unsigned integers, and because most of the corresponding databases do not accommodate a native 128-bit data type that can be used for IPv6 addresses as easily as the applications use IPv4 addresses, ensure that all major application code changes are identified and implemented in sync with the database changes.
- Nontrivial changes will undoubtedly be required in the applications supporting the necessary parsing functions for IPv6. Parsing IPv4 was far simpler with years of common libraries and I/O metaphors used for parsing. The applications responsible for parsing IPv6 addresses will require a major rewrite and include additional code base(s) and algorithms. This situation provides the applications developer an opportunity to abstract this behavior, leading the way for a high degree of use and reuse.
- Most applications today can use native URLs. The traditional standards for URLs and Uniform Resource Identifiers (URIs) do not accommodate IPv6. RFC 2732 does come into play here, but it does not "automatically" proxy or translate for traditional application support.
- Sockets and end-user applications:
  - IPv4 use assumes a single address per interface, and legacy code is very loose in this regard. Binding server sockets follows this paradigm. IPv6 use assumes multiple IPv6 addresses per interface and introduces more specific socket binding rules in new code.Legacy code cannot be updated because library and application problems are also involved.
- In IPv4 fragmentation was done by routers. In IPv6 fragmentation responsibility is moved to the applications and endpoints. This situation requires considerable skill set increases in developers as well as network architects and administrators to ensure packets are not dropped by older applications as they enter an IPv6 network.

  - One solution proposed is limiting maximum-transmission-unit (MTU) size to a common denominator, but there is no way to accurately predict all path constraints in a dynamic network. MTU discovery should not be an iterative and constant process under application control. Application testing and analysis tools will need to be refined, and these tools must support IPv6 natively as well as any and all transition solutions to be of use to developers and engineers.
- Skill sets and knowledge bases:
  - Though first introduced as IPng in 1993, IPv6 is still in its infancy in terms of deployment. Applications are lagging infrastructure because of the problems mentioned previously. Experience for staff and automated analysis and management tools must be built from pragmatic experience. Costs are high during this period of build-out and transition.

Summary

We have tried to represent the migration and compatibility complexity that IPv6 poses on applications in as brief a format as possible. We have addressed those applications that are used by IPv6 and those that use IPv6. We have taken the discussion to the socket level and briefly presented vendors' approaches to the solutions. Critical to the successful design and deployment of any IPv6 rollout will be the initial and tight integration of the applications migration strategy.

References

- http://www.deepspace6.net/docs/ipv6_status_page_apps.html
- "IPv4 -to-IPv6 Transition Strategies", Tim Rooney
- "Enabling IPv4 Applications for IPv6", Microsoft Press/TechNet
- "Porting Applications to IPv6", Mark Meiss
- "IPv6 and Applications", Jun Murai
- "IPv6 Porting Applications", Eva M. Castro