Published on *NetworkWorld.com Community*
(http://www.networkworld.com/community)

Troubleshooting IPv6 Networks and Systems

By *Scott Hogg*
Created *May 20 2011 - 11:37am*

Whether your organization has deployed IPv6 or not, you may end up troubleshooting IPv6-related issues as other nodes on the Internet move to dual-protocol connectivity. We need to consider how the introduction of IPv6 will change the way we troubleshoot networks, now that we are operating in a dual-protocol world. This article will focus on troubleshooting dual-protocol applications running on dual-protocol servers over a dual-protocol network.

One of the first places to start your troubleshooting process is at the endpoints of the communication. We need to first validate that the end-nodes have the correct IP addresses and are operational on their local networks. We will need to verify that each system has IPv4 and/or IPv6 addresses and correct DNS resolvers. IP addresses could be statically configured (common practice for servers in a datacenter) or dynamically configured (common practice for end-users on access networks). In IPv4 networks we would be troubleshooting DHCP. However, in an IPv6-enabled network we need to be able to troubleshoot Stateless Address Autoconfiguration (SLAAC) and investigate the ICMPv6 Router Advertisement (RA) messages coming from the local first-hop router. Based on the information contained in the RA message, an end-node could use SLAAC, stateless DHCPv6 (router provides the DNS prefix and resolver information) or stateful DHCPv6. We must also be aware that WindowsXP and Mac OS X do not use DHCPv6 but they can use SLAAC and then locally configure their DNS server. Another option would be to use Dibbler [18] (an open-source DHCPv6 client/server/relay).

Now that we have validated that the node has its IP addresses we need to validate that the host can ping its default router and can ping beyond that first hop. Often times the default router could have both a Aggregatable Global Unicast address and a Link-Local address. Our host may be configured with the Link-Local address that comes from the RA message. We can ping using Link-Local addresses as follows, depending on your operating system. When we ping a Link-Local address we must specify the interface that we would like to use to send this ICMPv6 echo request packet.

```
ping6 -I eth0 fe80::1
ping fe80::1%12
ping fe80::1%GigabitEthernet0/0
```

The next layer we want to troubleshoot involve the application mapping of human-recognizable fully-qualified domain names into IP addresses. We will need to validate IP connectivity to the DNS resolver and troubleshooting DNS lookups. We can use nslookup, dig, and the host command to validate the DNS queries for A and AAAA records as well as PTR records. We need to be cognizant of DNS servers that may communicate with IPv4-only or are dual-protocol. We also need to remember that WindowsXP, Windows Server2003 and Mac OS X only perform DNS lookups over IPv4 transport. It may also be useful to use Wireshark [19] or tcpdump [20] to view the DNS lookup packets. We will want to see how the client sends separate A and AAAA queries and follows RFC 4074 [21].

Most dual-protocol operating systems will perform DNS queries for IPv4 and IPv6 records and will prefer to make a connection using IPv6 if at all possible. However, older versions of Mac OS X use the first returned DNS response to make the connection. If the A record response came back first then the connection would take place over IPv4, but if the AAAA record response came back first then the connection would take place over IPv6. Furthermore, various web browsers [22] and other applications may not make connections over IPv6 even though the node has dual-protocol capability and is on an active dual-protocol network.

The next step in our troubleshooting methodology is to ensure bi-directional end-to-end connectivity with IPv4 and IPv6. This means that we will want to perform a ping and traceroute in both directions. We need to do these tests in both directions to see if there is any asymmetry in the communications path. We must also be aware of any IPv6-in-IPv4 tunnels that could exist along the path. There could be manually-configured tunnels, dynamically-configured tunnels (ISATAP, 6to4, Teredo) or translation (NAT-PT, NAT64/DNS64) occurring along the traffic path that could affect end-to-end connectivity. Tunnels could add to the latency and performance of the communications. We could also use pathping [23] (e.g. pathping -6 2001:DB8:0DD:BA11::1) or JPerf [24] to verify end-to-end performance.

The next phase of our troubleshooting will focus on IPv6-specific issues that we have not yet tested. IPv6-capable nodes follow a process of default address selection (RFC 3484 [25]). If there is something wrong with the prefix policy within the operating system it could cause mysterious behavior. This could affect either source address selection or destination address selection. On a Microsoft system we can use the "netsh interface ipv6 show prefixpolicies" command to view the policy table. On a BSD system we can use the ip6addrctl command and on a Solaris system we can use the ipaddrsel command to view the policy table.

Another thing we will need to test is the Neighbor Discovery Protocol (NDP). This is the IPv6 equivalent to IPv4's ARP. Because IPv6 doesn't use broadcast, the NDP ICMPv6 messages use multicast to map Layer-2 addresses (MAC Addresses) to IPv6 addresses. We can use ping to verify IPv6 connectivity to the other nodes on a LAN and then check the neighbor cache (like the IPv4 ARP cache). On a Windows host we can use the command "netsh interface ipv6 show neighbors". On

a Linux system we can use the "ip neighbor show" command. On a BSD system the command is "ndp -a" and on a Solaris system the command "netstat -p -f inet6" will show you its neighbor cache. On both a Cisco router and a Juniper router, the command is "show ipv6 neighbors".

Another problem that could be encountered on dual-protocol networks is links with reduced Maximum Transmission Unit (MTU) size. This can happen if the IPv6 packets have encountered a tunnel and the tunnel overhead has reduced the MTU size. If the IPv6 packets are placed inside a 6in4 tunnel within IPv4 Protocol 41 packets then the MTU size will be reduced by 20 bytes (the IPv4 header size). Because IPv6 routers do not perform fragmentation it is required that the router drop the IPv6 packet and send back an ICMPv6 Packet-Too-Big message indicating the preferred MTU size. The IPv6-capable source must then perform Path MTU Discovery (PMTUD) and then fragment the packet into the proper size. Using ping with various packet sizes can reveal if there is an MTU size reduction along the traffic path. You can perform a "ping -l 1500 2001:DB8:DEAD:C0DE::1" and then verify the ICMPv6 packet too big response with the embedded ideal packet size.

Once we have verified solid end-to-end connectivity with both protocols, then one of the final things to test is end-to-end application protocol communication. It is conceivable that there is a stateful firewall between the nodes that is blocking some type of traffic. In order to test this we may want to generate some synthetic traffic and validate that it makes it between the two end-nodes. We could use a utility like netcat6 [26] to create simulated traffic between the nodes using a specific port number. We could also use telnet or SSH (BTW, my favorite SSH client is SecureCRT [27]). We could perform an NMAP [28] scan of the destination host from the source. We could also use an IPv6-capable web browser and browse by IPv6 or IPv4 address [29].

As we begin to encounter more IPv6-enabled systems we will need to refine our troubleshooting skills to compensate for this added complexity. Even though dual-stack is the preferred transition technique [30], it is not a panacea. No one claimed that living in a dual-protocol world would be easy. During the lengthy period of time where systems will need to speak both the IPv4 and IPv6 language we will need to become bilingual and become fluent troubleshooting either protocol.

Scott

Stateless DHCPv6

By Cricket Liu (not verified) on Fri, 05/20/2011 - 4:25pm.

If you're using SLAAC and stateless DHCPv6, it's usually the DHCP server, not the router, that provides DNS-related parameters such as the IP addresses of recursive name servers and the search list.

Stateless DHCPv6

By Scott Hogg on Wed, 05/25/2011 - 9:36am.

Actually, it is common that the router provides the DNS information. These links provide information on how to configure a Cisco router with this feature.
http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6553/whitepaper_C11-472610.html
http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-dhcp_ps6441_TSD_Products_Configuration_Guide_Chapter.html#wp1080090

Scott

Great IPv6 Tools

By Anon (not verified) on Sat, 05/21/2011 - 2:51pm.

Thanks for the article, lots of good information. Have you checked out the free IPv6 tools at:

https://www.ultratools.com/ipv6Tools

Same as IPv4 only different

By Chris (not verified) on Tue, 05/24/2011 - 4:47am.

Great article. A lot of the steps are the same as IPv4 troubleshooting (ping, traceroute, check DNS, check DHCP/address configuration, etc.). For those who have been through it, it makes the IPv6 specific pieces much more understandable and less intimidating.

One point you brushed over and which took me a lot of time to understand when I first started looking into them are the prefix policies. I know from the RFCs how they should work and what the default policy should look like, but troubleshooting a problem with them would be tough. Especially when you start adding in some custom rules, like prefer an internal ULA, or have multiple interfaces, like on a dual-homed server. I would be interested in knowing if there was a tool which would evaluate the policy and tell you which interface a packet would use for a specific source/destination address pair.

- IPv6
- General discussions
- dual-protocol
- IPv6
- netcat6
- pathping

- ping
- traceroute
- Troubleshooting
- Wireshark

Source URL: http://www.networkworld.com/community/blog/troubleshooting-ipv6-networks-and-systems

Links:
[1] http://www.networkworld.com/community/node/45776
[2] http://www.networkworld.com/community/blog/testing-nat64-and-dns64
[3] http://www.ietf.org/rfc/rfc2464.txt
[4] http://www.ietf.org/rfc/rfc2472.txt
[5] http://www.ietf.org/rfc/rfc3572.txt
[6] http://www.ietf.org/rfc/rfc2590.txt
[7] http://www.ietf.org/rfc/rfc4338.txt
[8] http://tools.ietf.org/rfc/rfc4944.txt
[9] http://tools.ietf.org/rfc/rfc3146.txt
[10] http://www.ietf.org/rfc/rfc2470.txt
[11] http://tools.ietf.org/rfc/rfc2467.txt
[12] http://www.ietf.org/rfc/rfc2497.txt
[13] http://www.ietf.org/rfc/rfc793.txt
[14] http://www.ietf.org/rfc/rfc0768.txt
[15] http://www.ietf.org/rfc/rfc2960.txt
[16] http://www.ietf.org/rfc/rfc4340.txt
[17] http://www.hoggnet.com/NWWPics/Troubleshooting-Flowchart.png
[18] http://klub.com.pl/dhcpv6/
[19] http://www.wireshark.org
[20] http://www.tcpdump.org/
[21] http://tools.ietf.org/rfc/rfc4074.txt
[22] http://www.networkworld.com/community/blog/ipv6-enabled-web-browsers
[23] http://technet.microsoft.com/en-us/library/ff963096(WS.10).aspx
[24] http://code.google.com/p/xjperf/
[25] http://www.ietf.org/rfc/rfc3484.txt
[26] http://www.deepspace6.net/projects/netcat6.html
[27] http://www.vandyke.com/products/securecrt/
[28] http://insecure.org/
[29] http://www.ietf.org/rfc/rfc2732.txt
[30] http://www.networkworld.com/news/tech/2007/090507-tech-uodate.html