Applied Research Associates, Inc.
IPv6 Testbed Final Report

Abstract:

The Applied Research Associates Embedded Web Technology (EWT) Internet Protocol version 6 (IPv6) Testbed was developed to assist the Department of the Navy with the deployment of IPv6 enabled networks to the Global Information Grid (GIG). This focus was achieved via the research and documentation of key areas of interest including IPv6 migration scenarios, test cases, assessment metrics and procedures, the development of facilities capable of implementing and testing MANET and other mobile IPv6 technologies, and the development of a web-based portal to publish research findings and information as well as aggregating IPv6 information in a centralized repository. This final report is a comprehensive overview of the research activities executed in support of the ARA EWT IPv6 Testbed.

Authors:
Kenneth Lloyd Ayers III, Jonathan Faranda, David Hickman

Table of Contents

1.) Introduction

At the request of the Department of the Navy, Applied Research Associates, Inc (ARA) created the Embedded Web Technology (EWT), Internet Protocol version 6 (IPv6) Testbed. The objectives of this testbed included assisting the Department of the Navy with IPv6 network planning and implemenation of new networks utilizing IPv6, as well as transitioning legacy networks to IPv6. The Testbed was established in September 2006 and testing and evaluation ended in March 2009.

The Testbed was developed to make extensive use of various technologies and techniques during data collection and compilation. This included packet capture and analysis, network routing protocol evaluation, and the use of dynamic network topologies.

In order to simulate as many network conditions as possible, the IPv6 Testbed utilizes a plethora of networking and computing environments such as completely vitrualized networks, enterprise level networking hardware, and common, off-the-shelf devices.

2.) Background

Internet Protocol version 4 (IPv4) is the current version of the internet protocol used in most packet switching based networks. The addressing schema is based on a 32 bit address separated into 4 sections of 8 bits, each commonly referred to as a "dotted quad" notation. As implied, a 32 bit address has 2^{32} total addresses for use, which roughly equates to 4.3 billion addresses. Given the proliferation of the internet, and the desire for the data consumer to always be connected to information, it has become apparent that the current addressing schema is inadequate. Other shortcomings of IPv4 include areas such as security, mobility of nodes and networks, and quality of service (QoS).

Internet Protocol version 6 (IPv6) is the next evolution of the internet protocol standard. It was designed to take many of the successes of the current generation protocol, IPv4, and expand upon them while addressing shortcomings. The IPv6 addressing schema retains a familiar format with a few modifications that allow for a much larger addressing space. The base address is comprised of 128 bits which allows for 2^{128} total addresses. In base 10 scientific notation, this is approximately 3.4×10^{38} total addresses. Again, the total size of the IPv4 addressable space is 4.3×10^{9} total addresses, which means that the entire IPv4 address space would fit into the IPv6 space about 7.9×10^{28} times.

As the IPv4 space draws closer to exhaustion, it is becoming apparent to governments and private organizations that the need to migrate is soon approaching. The United States currently owns the majority of IPv4 addresses, forcing the rest of the world to adopt IPv6 at an accelerated pace. In an effort to maintain global military dominance, the Department of Defense (DoD) circulated a memorandum articulating a proposed transition plan for military and federal government networks.

In compliance and accordance with DoD mandate for migration, the Department of the Navy is working with ARA to develop and assist with testing and migration strategies for the transition to IPv6. The ARA EWT IPv6 Test Bed was established as a research and development operation to serve the needs of the United States Navy with researching and validating migration to the next generation of the internet protocol.

3.) Test Bed

The ARA IPv6 Testbed is a robust, highly configurable, and highly scalable virtual network. This network enclave consists of two Dell Blade servers mounted in a Dell 1955 Blade chassis with expansion slots for up to ten total servers. Running VMware's ESX server, these Blade servers are attached to an EMC CX300 Storage Area Network (SAN) via a 2 Gbps fiber switch. The SAN is currently configured with 11 hard drives, each having a rotational speed of 15,000 rpm and a capacity of 146 GB. The hard drives are installed in two RAID containers: one RAID 5 and one RAID 10 configuration. The Dell 1955 Blade chassis, Blade servers, fiber switch, and CX300 SAN are mounted in a half-rack that is housed in a secured room with dedicated power and cooling.

VMware ESX server enables installation of numerous virtual operating systems (called virtual machines) running concurrently on the same physical machine. The VMware ESX server supports a large breadth of operating systems including Microsoft Windows, Sun Solaris, Linux, and FreeBSD. While running, virtual machines can be interconnected via virtual switches from within the ESX server, which allows for the creation of virtualized networking environments. Each instance of ESX server (one per Blade) supports a maximum of 248 virtual switches. An advantage of using a virtualized environment is the cloning of a virtual machine. Once a virtual machine is installed and configured it can easily be cloned, or replicated, into as many copies as necessary to enable rapid deployment of virtual machines. This functionality allows the IPv6 test bed to quickly design and deploy virtual networks.

The ARA Ipv6 test bed operates behind a dedicated T1 line offering 1.544 Mbps link speeds. Moreover, this connection has allowed ARA to provide an IPv6 knowledge management portal that is used to store relevant IPv6 information and test cases for our external site partners and others interested in IPv6 migration techniques and testing scenarios.

The ESX Servers are hardened according to the Defense Information Systems Agency (DISA) UNIX Security Technical Implementation Guide (STIG). Following guidelines specified by the STIG ensures that the necessary steps are taken to harden these systems and mimic operating conditions. All system activity is logged and access to the systems is secured by both physical and logical access controls.

ARA also uses vendor specific hardware such as Cisco routers and switches, Linksys wireless routers, Dell laptops and desktops, and Fortress Technologies wireless bridges. These devices allow ARA to design and build complex networks for IPv6 testing and pemits modeling of physical networks.

ARA specially configured four Dell D820 Latitude laptop computers to use in conjunction with the ESX servers. These laptops are built around the Intel T7600 processor with 4 GB of RAM. Each laptop runs VMware workstation to enable multiple operating systems to run concurrently on a single system. Furthermore, each laptop has a Matrox device displaying information across multiple monitors. All four laptops are housed in a room directly attached to the ESX servers. The room is wired to allow full access from each laptop to the ESX servers.

4.) Windows Migration

Test Justification and Objectives

The use of Windows domains is pervasive in today's networking environment. While older versions of Windows including Windows XP and Windows Server 2003 have some support for IPv6, it is limited. IPv6 connectivity is supported in Windows Server 2008 as well as Exchange Server 2007. These server products support the use of dual-stack connectivity, harnessing both IPv4 and IPv6.

Several improvements have been made to the IPv6 network stack in both Windows Vista and Windows Server 2008. The Windows XP and Windows 2003 implementations of IPv6 and IPv4 have separate transport layers, while the new implementation of the IPv6/IPv4 TCP/IP stack is a single component, sharing the same Transport and Framing layers. This allows performance enhancements including Receive Window Auto Tuning and Compound TCP to increase speed and reliability in environments characterized by large amounts of packet loss and transmission delay.

Both Windows Vista and Windows Server 2008 have IPv6 installed and enabled by default. IPv6 is the preferred communications protocol for both of these operating systems as well, which means applications will attempt to communicate via IPv6 before using IPv4 as a failover. Another improvement in these operating systems is the full support of Internet Key Exchange (IKE). Windows XP and Windows Server 2003 do not support IKE, and IPSec configuration can only be done via text files in these operating systems. Windows Vista and Windows Server 2008 support IKE and allow for the configuration of IPSec via a snap-in module.

The goal of this test is to successfully execute the migration of a Windows domain running IPv4 to a dual-stack environment that allows the communication of network nodes in using both IPv4 and IPv6, including the use of IPv6 with Exchange Server 2007.

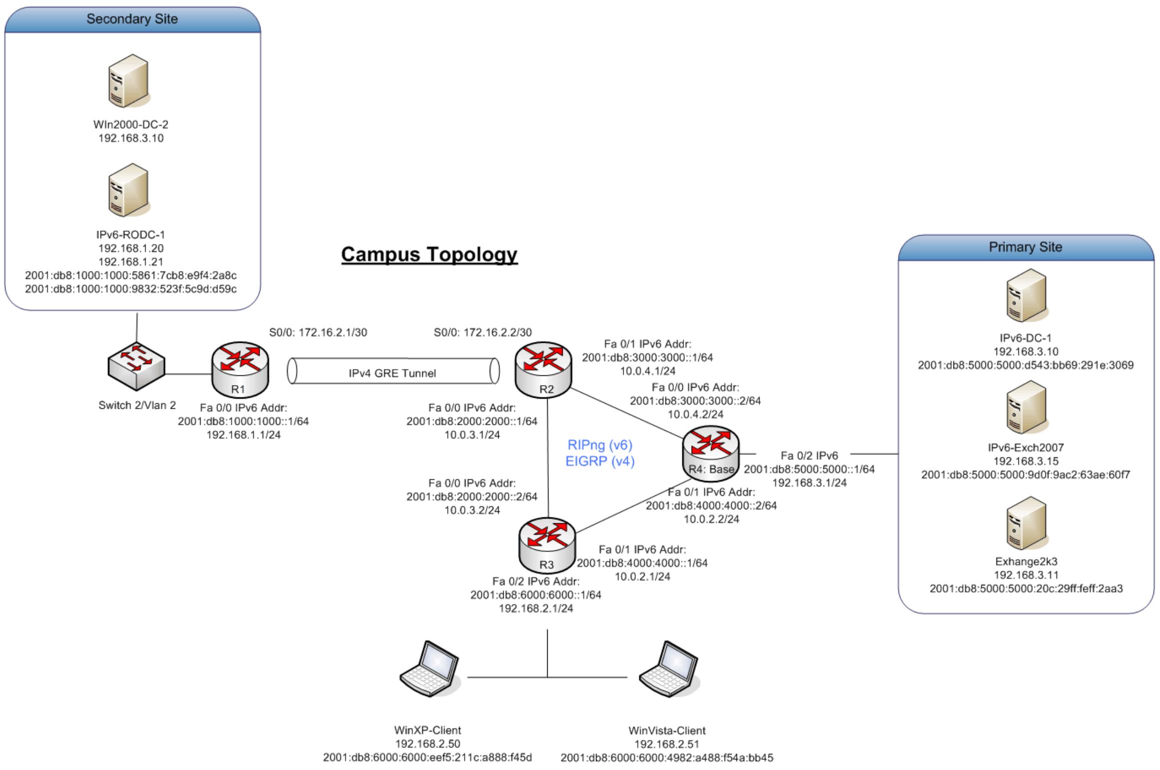
Test Protocols / Functionality

Windows 2003: IPv4 / SMTP / DNS
Windows 2008: IPv4 / IPv6 / SMTP / DNS

Environment Variables (Hardware and Software Setup):

Windows Server 2003/2008 Domain Controller, DNS:	IPv6-DC-1
Windows Server 2000 Domain Controller, DNS:	Win2000-DC-2
Windows Server 2003, Exchange Server 2003:	Exchange2k3
Windows Server 2008, Exchange Server 2007:	IPv6-Exch2007
Windows Server 2008 Core, RODC, DNS:	IPv6-RODC-1

Network Topology



Test Procedures

The test methodology for this migration study consists of establishing a legacy Windows domain using a Windows Server 2003 domain controller, a second domain controller running Windows Server 2000, and an Exchange Server 2003 running on the Windows 2003 Server operating system, and then migrating the entire domain to a Windows Server 2008 environment. An Exchange Server 2007 machine will be created, and the mailboxes from the Exchange Server 2003 machine will be migrated to the newer architecture.

DNS is essential to the proper function of a Windows Active Directory environment. Domain controllers use DNS for name resolution of AD objects. Because of this highly coupled integration of DNS and Active Directory, it should be noted that each domain controller will be running DNS. This service will therefore be migrated to Windows Server 2008 as well.

After establishing the legacy domain and migrating the entire network to a fully functional IPv6 environment, two clients will be used to confirm IPv6 connectivity and communications: a Windows XP SP2 client and a Windows Vista client.

Legacy Domain Setup

IPv6-DC-1 (VM: Win2k3_DC-1) Win2k3Configuration

Virtual machine IPv6-DC-1 was initially configured with Windows Server 2003 Standard. The operating system was installed on ESX Server, updated using Windows Update, and then configured with the following IP information:

NIC1:

IP:	192.168.3.10/24
IPv6:	2001:0db8:5000:5000:020c:29ff:fe10:129c
	fe80::020c:29ff:fe10:129c
GW:	192.168.3.1
	fe80::21e:7aff:fee4:ea4e%9
DNS:	2001:0db8:5000:5000:020c:29ff:fe10:129c
	192.168.3.10
	192.168.1.10

The IPv6 addressing was accomplished via router advertisement, while the IPv4 information was entered manually. Using the dcpromo.exe tool, IPv6-DC-1 was added to the IPv6Community.org domain as a domain controller. DNS was selected to be installed.

DNS Configuration:

Zone:	ipv6community.org
Lookup type:	forward and reverse
Replicate to:	all domain controllers
Update Policy:	allow only secure dynamic updates
Zone Transfers:	allow zone transfers (checkbox)
	only to servers listed on the name servers tab
Name Servers:	IPv6-DC-1; Win2000-DC-2
Change Notification:	Automatically notify, servers listed on the name servers tab

Records (automatically generated via DDNS):

IPv6-DC-1	A	192.168.3.10
Exchange2k3	A	192.168.3.11
Win2000-DC-2	A	192.168.1.10
Exchange2k3	AAAA	2001:0db8:5000:5000:020c:29ff:feff:2aa3
IPv6-DC-1	AAAA	2001:0db8:5000:5000:020c:29ff:fe10:129c

Reverse zone:	192.168.3.x Subnet

Replicate to:	all domain controllers
Update Policy:	allow only secure dynamic updates
Zone Transfers:	allow zone transfers (checkbox)
	Only to servers listed on the name servers tab
Name Servers:	IPv6-DC-1; Win2000-DC-2
Change Notification:	Automatically notify, servers listed on the name servers tab

Reverse zone:	192.168.1.x Subnet
Replicate to:	all domain controllers
Update Policy:	allow only secure dynamic updates
Zone Transfers:	allow zone transfers (checkbox)
	Only to servers listed on the name servers tab
Name Servers:	IPv6-DC-1; Win2000-DC-2
Change Notification:	Automatically notify, servers listed on the name servers tab

After DNS configuration, Active Directory was populated with multiple user and computer objects, including the user Kenny.Ayers and computer IPv6-Exchange2k3 in preparation for the addition of this Exchange Server 2003 machine.

Active Directory Site and Services were configured for two sites, Primary-Site and Secondary-Site. Between the sites is a network link with transfer speeds equivalent to that of a satellite link, 1.544 mbps.

Please see section 4.0 Network Topology for an overview of the overall network site configuration.

Active Directory Sites and Services configuration:

- Changed Default Site name to: Primary-Site
- Created new site: Secondary-Site
- Changed Default site link name to: IP-Site-Link
    - Includes: Primary-Site, Secondary-Site
- Added SMTP site link: SMTP-Site-Link
    - Includes: Primary-Site, Secondary-Site
- Added Subnet 192.168.3.0/24: Primary-Site
- Added Subnet 192.168.1.0/24: Secondary-Site
- Right-Click IPV6-DC-1 Server Object > Properties >
- Add IP and SMTP transports to preferred bridgehead

The default functional level of a newly installed domain on Windows Server 2003 is Mixed Mode. We raised the domain functional level to Native Mode:

Open Active Directory Users and Computers > Right click domain name “ipv6community.org” > “Raise domain functional level” > “Windows 2000 Native”

Exchange Server 2003 (Machine Name: Exchange2k3) Configuration

Virtual machine Exchange2k3 uses Windows 2003 Server Standard as the base operating system. The operating system was installed on ESX Server, updated using Windows Update, and then configured with the following IP information:

NIC1:

IP:	192.168.3.11/24
IPv6:	2001:0db8:5000:5000:020c:29ff:feff:2aa3
GW:	192.168.3.1
	fe80::21e:7aff:fee4:ea4e%9
DNS:	2001:0db8:5000:5000:020c:29ff:fe10:129c
	192.168.3.10
	192.168.1.10

Next Exchange2k3 was added to the IPv6Community.org domain: My Computer > Properties > Computer Name > Change > Enter domain name and credentials when prompted.

Exchange Server 2003 requires some preparation before installation:

Installed NNTP, SMTP, WWW services, ASP.NET
Installed Windows support tools from CD (D:\SUPPORT\TOOLS)
Installed Windows Server Reskit:
(http://www.microsoft.com/downloads/details.aspx?familyid=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en)
Downloaded updated Exchange 2003 Deployment Tools:
(http://www.microsoft.com/downloads/details.aspx?familyid=271e51fd-fe7d-42ad-b621-45f974ed34c0&displaylang=en)

Next DCDiag.exe and NetDiag.exe should be run to verify a properly configured environment:

Run DCDiag.exe > All tests passed
Run NetDiag.exe > Passed
ForestPrep from Exchange2k3 disc (using deployment tools wizard):
    D:\SETUP\I386\SETUP.exe /forestprep
DomainPrep from Exchange2k3 disc (using deployment tools wizard):
    D:\SETUP\I386\SETUP.exe /domainprep

Finally Exchange Server 2003 was installed from the installation media, which was followed by the installation of Service Pack 2 for Exchange.

Win2000-DC-2 (VM: Win2000_DC-2) Configuration

Virtual machine Win2000-DC-2 was setup using Windows Server 2000 Professional. Windows updates were executed and Service Pack 4 was installed. The IP information is as follows:

NIC1:

IP:	192.168.1.10/24
GW:	192.168.1.1
DNS:	192.168.3.10
	192.168.1.10

The dcpromo.exe tool was used to promote Win2000-DC-2 to domain controller status. Win2000-DC-2 was added as an additional domain controller for the ipv6community.org domain that was setup in section 5.1.1.

DNS Configuration:

DNS installed via “Windows 2000: Configure Your Server”

Add forward lookup zone: Right click “Forward lookup zones” > “New Zone” > “Standard Secondary” > “Name: ipv6community.org” > “IP address of Primary: 192.168.3.10”

Add reverse lookup zone: Right click “reverse lookup zones” > “New zone” > “Standard secondary” > “Network ID: 192.168.3.x” > Master DNS Server address: “192.168.3.10”

Add reverse lookup zone: Right click “reverse lookup zones” > “New zone” > “Standard secondary” > “Network ID: 192.168.1.x” > Master DNS Server address: “192.168.3.10”

Using Active Directory Site and Services, Win2000-DC-2 was designated as a Secondary-Site domain controller, and used as the preferred bridgehead for both IP and SMTP traffic:

Right-Click WIN2000-DC-2 > Properties > Add IP and SMTP transports to preferred bridgehead

Migration Planning and Notes

The migration process used in this document can be considered a phased migration, as each component is added to the domain in a sequential fashion without causing unnecessary breaks in services.

First IPv6-DC-1 will be upgraded in-place to Windows Server 2008. Our goal is to deploy a Windows Server 2008 RODC at Secondary-Site, however that requires the domain be at the Windows Server 2003 functional level. The Win2000-DC-2 box must be demoted, and then the functional level of the domain must be raised on IPv6-DC-1. Next a RODC will be installed at Secondary-Site (where Win2000-DC-2 was located). DNS entries must be changed to reflect the use of IPv6-RODC-1 instead of Win2000-DC-2. Finally, because Exchange Server 2007 does not have the option to upgrade older versions of exchange, a new server will be established and the mailboxes present on Exchange2k3 will be moved to the Exchange Server 2007 virtual machine.
To add Windows Server 2008 machines to a domain, the domain must be in either Windows 2000 Native or Windows Server 2003 mode. Servers that are upgraded in-place (that is, Windows Server 2008 is installed overtop the original operating system) must be Windows 2003 SP1 or greater. Finally all domain controllers must be Windows Server 2000 SP or greater.

Windows Server 2003 (Machine Name: IPv6-DC-1) Migration to Windows Server 2008

Virtual machine IPv6-DC-1 will be upgraded in-place to Windows Server 2008 (from Windows Server 2003). Hardware and application compatibility need to be confirmed. Software is vendor specific, and for our purposes will not present any issues as IPv6-DC-1 is a dedicated virtual machine, server only as a Domain Controller running DNS. Hardware compatibility is confirmed via:

http://www.microsoft.com/whdc/hcl/default.mspx.

The first step in the migration process is to run the DCDIAG (domain controller diagnostic) tool, from the Support Tools folder, (\support\tools\suptools.msi):

```
dcdiag.exe /e
```

The /e switch runs the tests on all directory servers in the entire enterprise. This diagnostic is essential as the health of the domain must be verified before migration changes are made. Verify all tests pass, and resolve any issues that may be identified by the diagnostic test.

Next we must execute schema updates and prepare the domain. Execute the following commands from the Windows Server 2008 DVD on the Flexible Single Master Operations (FSMO) role holder:

```
d:>\sources\adprep\adprep.exe /forestprep
d:>\sources\adprep\adprep.exe /domainprep /gpprep
d:>\sources\adprep\adprep.exe /rodcprep
```

The first command extends the AD schema, while the second prepares the domain. The final command prepares the domain to allow for the use of RODCs. To ensure proper compatibility Windows Server 2000 domain controller machines must be Service Pack 4 or later. The first domain controller to be migrated to Windows Server 2008 in the domain must be FSMO role holder.

To upgrade IPv6-DC-1 to Windows Server 2008:

1) Insert DVD
2) Setup.exe > Click "Install Now"
3) Click "Go Online to Get the Latest Updates" button.
4) Enter product key.
5) Accept the Agreement
6) Upgrade
7) Review compatibility
8) Installation restarts, installs OS, completes.

After Win2000-DC-2 is demoted via the dcpromo.exe tool, the functional level of the domain and forest must be raised to permit the installation of the RODC at Secondary-Site:

1) Open Active Directory snap-in.
2) Right-click ipv6community.org > Raise Domain Functional Level > Windows Server 2003
3) Open up AD Domains and Trusts
4) Right-click Active Directory Domains and Trusts > Raise Forest Functional Level > Windows Server 2003

Now virtual machine IPv6-DC-1 has been migrated to Windows Server 2008. The IPv6 addresses are assigned according to adapter specific information which changes after the installation of Windows Server 2008, thus some of the IP information changed:

NIC1:	
IP:	192.168.3.10/24
IPv6:	2001:0db8:5000:5000:d543:bb69:291e:3069
GW:	192.168.3.1
	fe80::21e:7aff:fee4:ea4e%9
DNS:	2001:0db8:5000:5000:d543:bb69:291e:3069
	192.168.3.10
	192.168.1.10

Because the IPv6 address of IPv6-DC-1 changed, the DNS entry for all the machines on the domain need to be updated to reflect the use of the new IPv6 address.

Windows Server 2000 Domain Controller (Machine Name: Win2000-DC-2) Migration

Virtual machine Win2000-DC-2 will be replaced with a Windows Server 2008 RODC. Win2000-DC-2 is installed in the AD site: Secondary-Site. Read-only domain controllers require domain and forest preparation that in turn requires a functional level of Server 2003 for both the domain and forest. Because of this, Win2000-DC-2 must be demoted, and the RODC domain prep must be executed. To demote Win2000-DC-2 from domain controller status, the following was executed:

1) Run DCPromo.exe from command prompt
2) "Next" to remove AD services
3) Follow prompts to remove AD

Installation of Windows Server 2008 Core RODC (IPv6-RODC-2.ipv6community.org):

Virtual machine IPv6-RODC-2 was setup using Windows Server 2008 Core. Because the version of ESX Server used did not support this operating system, the Windows 2003 Enterprise server must be selected as the type of operating system in the ESX Server client shell. This allows for the installation of VMWare tools which allows the network interface card(s) to be recognized.

VMWare tools must be installed using the using the option "Upgrade VMware tools automatically without interacting with the guest OS" in the ESX Server client shell. Once the virtual machine is restarted, the NICs will have the proper drivers installed and should function.

To assign IP information, the NIC IDs must be determined:

```
>netsh interface ipv4 show interfaces
```

```
2       Local Area Connection
4       Local Area Connection 2
```

The NIC IDs are 2 and 4. Now the NICS must be configured:

```
>netsh interface ipv4 set address name="2" source=static
address=192.168.1.20 mask=255.255.255.0 gateway=192.168.1.1
>netsh interface ipv4 set address name="4" source=static
address=192.168.1.21 mask=255.255.255.0 gateway=192.168.1.1
>netsh interface ipv4 add dnsserver name="2"
address=192.168.3.10
>netsh interface ipv4 add dnsserver name="4"
address=192.168.3.10
```

Windows update should be run to ensure the operating system has the latest security patches and updates, however it cannot be accessed using the GUI (as there is no GUI in Server Core). The following commands were used:

```
    >Cscript c:\windows\system32\scregedit.wsf /au 4
    >Net stop wuauserv
    >Net start wuauserv
```

To force and update check:

```
    >Wuauclt /detectnow
```

To check for installed updates (patches):

```
    >wmic qfe list
```

To rename the computer, the following command was used:

```
    >WMIC ComputerSystem Where Name="%COMPUTERNAME%" Call
Rename Name="IPV6-RODC-1"
```

Options for the DCPromo.exe tool can be found here: http://technet.microsoft.com/en-us/library/cc732887.aspx

IPv6-RODC-1 was promoted to a domain controller using the following command:

```
    >dcpromo /answer:C:\unattend.txt
```

The contents of unattend.txt:

```
    [DCInstall]
    ReplicaOrNewDomain=ReadOnlyReplica
    ReplicaDomainDNSName=ipv6community.org
    ReplicationSourceDC=IPV6-DC-1.ipv6community.org
    SiteName=Secondary-Site
    InstallDNS=Yes
    ConfirmGc=Yes
    CreateDNSDelegation=No
    UserDomain=ipv6community.org
    UserName=administrator
    Password=
    DatabasePath="C:\Windows\NTDS"
    LogPath="C:\Windows\NTDS"
    SYSVOLPath="C:\Windows\SYSVOL"
    SafeModeAdminPassword=****
    RebootOnCompletion=Yes
```

After the completion of the dcpromo.exe tool, IPv6-RODC-1 was a fully functional read-only domain controller for Secondary-Site. The IP configuration information is:

NIC1:	
IP:	192.168.3.10/24
IPv6:	2001:0db8:1000:1000:5861:7cb8:e9f4:2a8c
GW:	192.168.3.1
	fe80::20a:41ff:fe5f:b80%2
DNS:	2001:0db8:5000:5000:d543:bb69:291e:3069
	192.168.3.10
	192.168.1.10

The DNS address of the current network nodes includes an entry for 192.169.1.10 (Win2000-DC-2) as a DNS server. This needs to be removed, and 2001:0db8:5000:5000:d5

Exchange Server 2003 (Machine Name: Exchange2k3) Migration:

Exchange Server 2007 is only supported on 64-bit processors, running the x64 version of Windows Server. Furthermore Exchange Server 2003 cannot be upgraded to Exchange Server 2007 (Redmond 17). The only option for installing Exchange Server 2007 is a new installation, thus to migrate our Windows domain to an environment which supports IPv6 communication with Exchange, we will have to create a new 64-bit virtual machine, install Exchange 2007, and then move the mailboxes from the older Exchange Server 2003 machine.

The domain in which the Exchange Server 2007 machine is installed has specific requirements for the schema master, domain controllers, and global catalog servers: they must all be Windows Server 2003 SP1, R2 or later (Redmond 18). The installation site must contain a global catalog server. The domain in which the Exchange Server 2007 is installed is recommended (however now required) to be at the Windows 2003 functional level. Any older Exchange servers must be Exchange Server 2003 SP2 or greater. Finally, it should be noted that any Windows Server 2000 domain controllers in the same site as the Exchange Server 2007 will case the Exchange installation to fail.

Exchange Server 2007 (Machine Name: IPv6-Exch2007) Setup:

Virtual machine IPv6-Exch2007 was setup as a 64-bit virtual machine with Windows Server 2008 Enterprise x64. Windows updates were done. The IP addressing information is as follows:

IP:	192.168.3.15
IPv6:	2001:db8:5000:5000:9d0f:9ac2:63ae:60f7
SM:	255.255.255.0
GW:	192.168.3.1
DNS:	192.168.3.10
	192.168.1.20

Windows Powershell was installed via the Server Manager interface:

```
     Server Manager > Features > Add Features > Check:
Windows Powershell > Next
```

IPv6-Exch2007 was added to the IPv6Community.org domain.

The Ldifde.exe tool was installed:

```
     c:\> ServerManagerCmd –I RSAT-ADDS
```

The Exchange Server 2007 setup program was used to prepare the domain, extend the AD schema, and setup permissions to allow legacy versions of Exchange to coexist peacefully in the same domain as the new version of exchange. These commands were run from the root of the drive containing the setup disk:

```
     D:\>Setup /PrepareAD          'create forest-wide objects
     D:\>Setup /PrepareSchema  'extend AD schema for Exchange 2007
     D:\>Setup /PrepareDomain
     D:\>Setup /PrepareLegacyExchangePermissions   'Legacy
Permissions
```

The Exchange Best Practices Analyzer tool was run (ExPBA).

Next IIS was installed:

1) Server Manager > Add Roles > Click Web Server (IIS), Next > Next >
    a. Add Role Services, in addition to default selections: IIS 6 Metabase Compatibility, IIS 6 Management Console, Dynamic Content Compression, Basic Authentication, Windows Authentication, Digest Authentication
2) Next > Install > Close

Exchange Server 2007 was installed via Setup.exe:

1) Open Setup.exe > Next > (Accept Terms) Next > (No Error Reporting) Next > Typical Exchange Server Installation > Browse, Select Exchange2k3 > Install > Finish

Client mailboxes were moved over via the Exchange Mailbox Move tool.

Conclusions

After configuring the Windows Domain and network nodes as detailed in section 5.0, Windows XP and Windows Vista clients were added to the domain. The Windows XP client was configured to use Outlook 2003 while the Windows Vista client was configured to use Outlook 2007. Both clients were configured to use the Exchange Server 2007 service from IPv6-Exch2007. By disabling IPv4 connectivity on the Windows XP client, we verified via the Wireshark packet interrogation tool that it communicated with the Exchange Server 2007 via IPv6 successfully. Finally because the Windows Vista machine prefers IPv6, we left IPv4 connectivity enabled. Again we verified using the Wireshark packet inspection tool that the Vista client was communicating with IPv6-Exchange2007 via IPv6 successfully, and was able to successfully use the Exchange service.

This functionality is inline with what we expected, there were no exceptional circumstances during the verification of IPv6 connectivity and communications between the client and server nodes.

References:

http://www.microsoft.com/technet/network/ipv6/ipv6faq.mspx

http://blogs.msdn.com/exchangefaqs/archive/2008/02/01/will-there-be-any-support-for-ipv6-in-exchange-2003.aspx

http://technet.microsoft.com/en-us/library/bb629624(EXCHG.80).aspx

http://technet.microsoft.com/en-us/library/aa997281(EXCHG.80).aspx

Morimoto, R., et al. Windows Server 2008 Unleashed. USA: Sams Publishing, 2008.

Morimoto, R., et al. Windows Server 2003 Unleashed. USA: Sams Publishing, 2006.

Mueller, J. P. Administering Windows Server 2008 Server Core. Indianapolis, Indiana: Wiley Publishing, Inc., 2008.

Redmond, T. Microsoft Exchange Server 2007 with SP1. USA: Elsevier, 2008.

5.) DNS

Test Justification and Objectives:

The domain name service (DNS)

RFC 4074 presents an issue wherein DNS clients (resolvers) send AAAA resource record (RR) requests, and receive unexpected behavior from an authoritative DNS server. These unexpected responses can include ignored queries, "Name Error" returns, erroneous code returns, and broken responses. This issue applies to not only to AAAA RR requests, but also to mail exchange (MX), name server (NS), and start of authority (SOA) RR requests. RFC 4472 expounds upon this issue:

"The problems are serious because when looking up a DNS name, typical getaddrinfo() implementation, with AF_UNSPEC hint given, first try to query the AAAA records of the name, and after receiving a response, query the A records. This is done in a serial fashion – if the first query is never responded to (instead of properly returning a negative answer), significant time-outs will occur.

In consequence, this is an enormous problem for IPv6 deployments, and in some cases, IPv6 suport in the software has even been disabled due to these problems."
(Durand et al 6)

RFC 4472 presents an issue whereby caching resolvers that contain cache entries for both A and AAAA records with different time to live (TTL) values for a single domain encounter resolution issues. This scenario may occur as the result of a caching DNS resolver querying for an MX record, and receiving additional information containing A and AAAA records with the different TTLs. RFC 4472 describes a scenario where caching DNS resolving contains an A record with a TTL of 300 and an AAAA record with a TTL of 100:

"The difference to courtesy additional data is that the A/AAAA records served by the parent zone cannot be queried explicitly. Therefore, after 100 seconds the AAAA record is removed from the cache(s), but the A record remains. Queries for the remaining 200 seconds (provided that there are no further queries from the parent that could refresh the caches) only return the A record, leading to a potential operational situation with unreachable servers."
(Durand et al 10)

There are two objectives for the tests described herein. The first test will consist of recreating a scenario wherein an authoritative DNS server contains an A record but not AAAA record for a network host. We will query the DNS server for the AAAA record and examine the DNS response data for RFC compliance. Out second test will be to create a scenario wherein we may determine the behavior of the caching resolver in Windows XP SP2. The objective in this test is to determine how the resolver behaves when the resolver cache contains entries with differing TTLs.

Discussion of Message Format:

The key area of inspection for test data for the DNS AAAA test will be the communication messages, described in RFC 1035:

```
"All communications inside of the domain protocol are carried in a
single
format called a message.  The top level format of message is divided
into 5 sections (some of which are empty in certain cases) shown below:

    +---------------------+
    |        Header       |
    +---------------------+
    |       Question      | the question for the name server
    +---------------------+
    |        Answer       | RRs answering the question
    +---------------------+
    |      Authority      | RRs pointing toward an authority
    +---------------------+
    |      Additional     | RRs holding additional information
    +---------------------+"
```

(Mockapetris 25)

Specifically, we will be looking in the header of the message, to examine the *response code*, found in the RCODE location, per RFC 1035:

```
"The header contains the following fields:

                                    1  1  1  1  1  1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                      ID                       |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    QDCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ANCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    NSCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
    |                    ARCOUNT                    |
    +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+"
```

(Mockapetris 26)

The possible responses found in the RCODE portion of the message header are:

```
"RCODE           Response code - this 4 bit field is set as part of
                responses.  The values have the following
                interpretation:

                0               No error condition

                1               Format error - The name server was
                                unable to interpret the query.

                2               Server failure - The name server was
                                unable to process this query due to a
                                problem with the name server.

                3               Name Error - Meaningful only for
                                responses from an authoritative name
                                server, this code signifies that the
                                domain name referenced in the query
                                does not exist.

                4               Not Implemented - The name server does
                                not support the requested kind of
                                query.

                5               Refused - The name server refuses to
                                perform the specified operation for
                                policy reasons.  For example, a name
                                server may not wish to provide the
                                information to the particular
                                requester, or a name server may not
                                wish to perform a particular operation
                                (e.g., zone transfer) for particular
                                data.

                6-15            Reserved for future use."
```

(Mockapetris 26, 27)

Test Protocols / Functionality:

DNS for IPv4 and IPv6 as served by the Berkeley Internet Name Domain (BIND) daemon: test for responses to AAAA record requests.

Windows XP SP2 resolver caching test

Environment Variables (Hardware and Software Setup):

Physical ESX Server 3.0.1
2x Dual Core Xeon processors @ 3.0 GHz
16 GB RAM

Virtual BIND 9.5.0 DNS Primary Server:
CentOS 5.2 (assigned 4x processors, 768 MB RAM)

Virtual BIND 9.5.0 DNS Slave Server:
CentOS 5.2 (assigned 4x processors, 256 MB RAM)

Virtual DNS Resolvers:
Windows XP SP2 (assigned 2x processors, 512 MB RAM)
CentOS 5.2 (assigned 4x processors, 256 MB RAM)

DiG 9.3.4-P1 (tool used to query DNS information)

Windows XP S2 ipconfig (tool used to view and modify Windows IP stack configuration)

Network Topology:

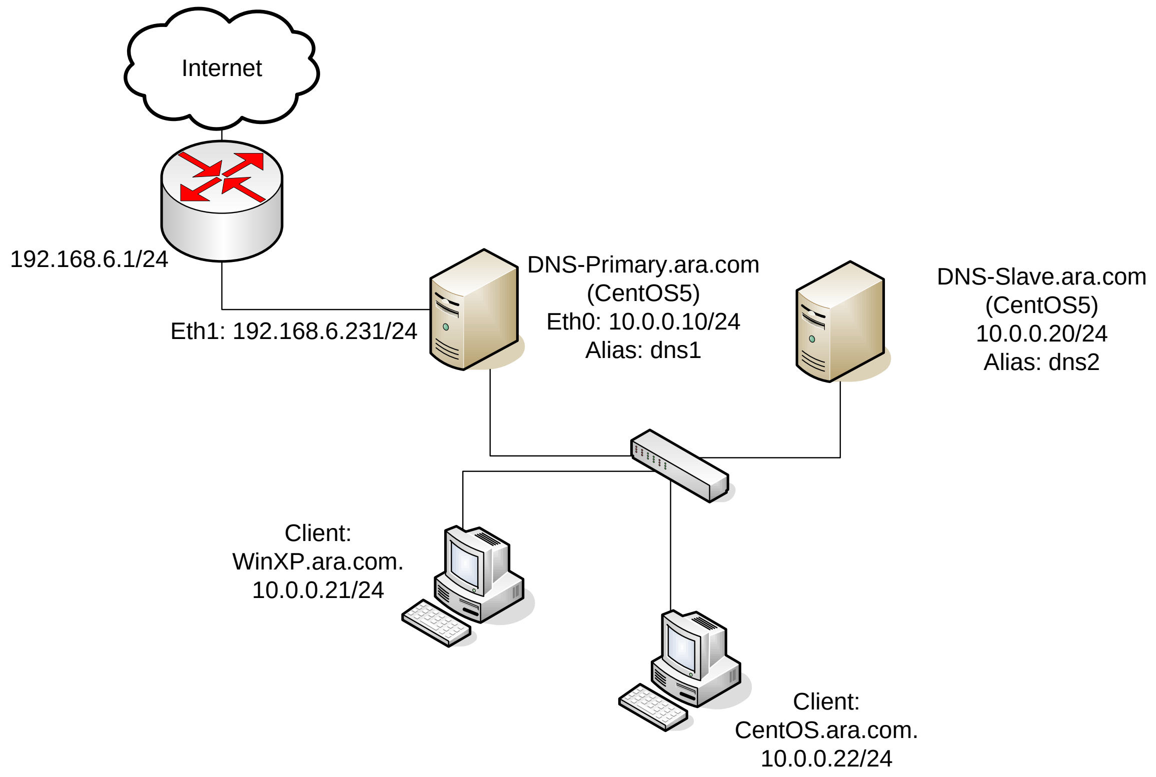


Figure 1: DNS Test Network Topology

Test Procedures:

Install/Configure Master DNS (CentOS5 Primary-DNS):

1)

```
% cd /tmp
% ftp ftp.isc.org.
Name: ftp
ftp> cd /isc/bind9/9.5.0/
ftp> get bind-9.5.0.tar.gz
ftp> quit
% tar zxvf bind-9.5.0.tar.gz
% ./configure
% make all
% make install
```

Installed!

Start named

```
% /usr/local/sbin/named
```

Check for errors

```
% grep daemon /etc/syslog.conf
```

2) Create needed files for DNS:

/var/named/db.ara.com

```
$TTL 3h
ara.com. IN SOA DNS-Primary.ara.com. kayers.ara.com (
   1     ; Serial
   3h    ; Refresh after 3 hours
   1h    ; Retry after 1 hour
   1w    ; Expire after 1 week
   1h )  ; Negative caching TTL of 1 hour

;
;  Name Servers
;
ara.com. IN NS DNS-Primary.ara.com.
ara.com. IN NS DNS-Slave.ara.com.

;
;  Addresses for the canonical names
;
localhost.ara.com.       IN A  127.0.0.1
DNS-Primary.ara.com.     IN A  10.0.0.10
DNS-Slave.ara.com        IN A  10.0.0.20
WinXP.ara.com.           IN A  10.0.0.21
```

```
CentOS.ara.com     300  IN A 10.0.0.22
CentOS.ara.com     100  IN AAAA fe80::20c:29ff:fed3:7568

;CentOS AAAA record is a link local address
;this isn't recommended, however will be used to
;demonstrate functionality for our caching
;resolver test case

;
; Aliases
;
dns1.ara.com. IN CNAME DNS-Primary.ara.com.
dns2.ara.com. IN CNAME DNS-Slave.ara.com.
```

/var/named/db.10.0.0

```
$TTL 3h
0.0.10.in-addr.arpa. IN SOA DNS-Primary.ara.com.
kayers.ara.com (
  1     ; Serial
  3h    ; Refresh every 3 hours
  1h    ; Retry after 1 hour
  1w    ; Expire after 1 week
  1h )  ; Negative caching TTL of 1 hour

;
; Name Servers
;
0.0.10.in-addr.arpa. IN NS DNS-Primary.ara.com.
0.0.10.in-addr.arpa. IN NS DNS-Slave.ara.com.

;
; Addresses to point to canonical name
;
10.0.0.10.in-addr.arpa. IN PTR DNS-Primary.ara.com.
20.0.0.10.in-addr.arpa. IN PTR DNS-Slave.ara.com.
21.0.0.10.in-addr.arpa. IN PTR WinXP.ara.com.
22.0.0.10.in-addr.arpa. IN PTR CentOS.ara.com.
```

/var/named/db.127.0.0

```
$TTL 3h
0.0.127.in-addr.arpa. IN SOA DNS-Primary.ara.com. kayers.ara.com.
(
    1     ; Serial
    3h    ; Refresh every 3 hours
    1h    ; Retry after 1 hour
    1w    ; Expire after 1 week
    1h )  ; Negative caching TTL of 1 hour

0.0.127.in-addr.arpa.     IN NS DNS-Primary.ara.com.
```

```
0.0.127.in-addr.arpa.     IN NS DNS-Slave.ara.com.

1.0.0.127.in-addr.arpa.   IN PTR localhost.
```

/etc/named.conf

```
//Bind Configuration file

options {
    listen-on { 10.0.0.10/24; 127.0.0.1 };
    listen-on-v6 { any; };
    directory "/var/named";
};

zone "ara.com" in {
    type master;
    file "db.ara.com";
};

zone "0.0.10.in-addr.arpa" in {
    type master;
    file "db.10.0.0";
};

zone "0.0.127.in-addr.arpa" in {
    type master;
    file "db.127.0.0";
};

zone "." in {
    type hint;
    file "db.cache";
};

//Sets controls for rndc
controls {
    inet * allow { any; } keys { "rndc-key"; };
};

//Password
key "rndc-key" {
    algorithm hmac-md5;
    secret "IPv6";
};
```

/etc/rndc.conf

```
//Bind rndc configuration file

options {
     default-server localhost;
     default-key "rndc-key";
};

server localhost {
     key "rndc-key";
};

server DNS-Slave.ara.com {
     key "rndc-key";
};

key "rndc-key" {
     algorithm hmac-md5;
     secret "IPv6";
};
```

3) Download the root hints file:

ftp.rs.internic.net (anonymous login as "ftp")
/domain/named.root

Copy to /var/named/db.cache

4) Start name server daemon

(Ensure network settings are correct)

Eth0:

IP:	10.0.0.10
Subnet:	255.255.255.0
Gateway:	192.168.6.1

Eth1:

IP:	192.168.6.231
Subnet:	255.255.255.0
Gateway:	192.168.6.1

/usr/local/sbin/named

5) Modify network and nameserver settings:

Change hostname in /etc/sysconfig/network:

```
HOSTNAME=DNS-Primary.ara.com
```

Put nameserver and domain in /etc/resolv.conf:

```
domain ara.com
nameserver 10.0.0.10
nameserver 10.0.0.20
```

6) Set named to execute on startup

Create startup script (/root/named.sh):

```
# vi /root/named.sh

      #!/bin/sh

      /usr/local/sbin/named
      exit 0

# chmod +x named.sh
```

Modify startup file to include script (/etc/rc.local):

```
# vi /etc/rc.local
```

Add line: `/bin/sh /root/named.sh`

7) Configure system to allow DNS requests to be made to internet DNS servers:

(/etc/sysctl.conf)
```
net.ipv4.ip_forward = 1
```

Enables IP forwarding allows DNS requests to go from Eth0, to Eth1, then to router, then to internet.

Install/Configure Slave DNS (CentOS5)

1)
```
% cd /tmp
% ftp ftp.isc.org.
Name: ftp
ftp> cd /isc/bind9/9.5.0/
ftp> get bind-9.5.0.tar.gz
ftp> quit
% tar zxvf bind-9.5.0.tar.gz
% ./configure
% make all
% make install
```

Installed!

Start named
```
% /usr/local/sbin/named
```

Check for errors
```
% grep daemon /etc/syslog.conf
```

2) Create needed files for DNS:

/var/named/db.127.0.0

```
$TTL 3h
0.0.127.in-addr.arpa. IN SOA DNS-Primary.ara.com. kayers.ara.com.
(
    1     ; Serial
    3h    ; Refresh every 3 hours
    1h    ; Retry after 1 hour
    1w    ; Expire after 1 week
    1h )  ; Negative caching TTL of 1 hour

0.0.127.in-addr.arpa.     IN NS DNS-Primary.ara.com.
0.0.127.in-addr.arpa.     IN NS DNS-Slave.ara.com.

1.0.0.127.in-addr.arpa.   IN PTR localhost.
```

/etc/named.conf

```
//Bind Configuration file

options {
    listen-on { 10.0.0.20/24; 127.0.0.1; };
    listen-on-v6 { any; };
    directory "/var/named";
};

zone “ara.com” in {
    type slave;
    file “bak.ara.com”;
    masters { 10.0.0.10; };
};

zone “0.0.10.in-addr.arpa” in {
    type slave;
    file “bak.10.0.0”;
    masters { 10.0.0.10; };
};

zone “0.0.127.in-addr.arpa” in {
    type master;
    file “db.127.0.0”;
};

zone “.” in {
    type hint;
    file “db.cache”;
};

//Sets controls for rndc
controls {
    inet * allow { any; } keys { “rndc-key”; };
};

//Password
key “rndc-key” {
    algorithm hmac-md5;
    secret “IPv6”;
};
```

/etc/rndc.conf

```
//Bind rndc configuration file

options {
    default-server 127.0.0.1;
    default-key “rndc-key”;
};

server localhost {
    key “rndc-key”;
};

server DNS-Primary.ara.com {
    key “rndc-key”;
```

```
  };

  key "rndc-key" {
      algorithm hmac-md5;
      secret "IPv6";
  };
```

3) Download the root hints file:

ftp.rs.internic.net (anonymous login as “ftp”)
/domain/named.root

Copy to /var/named/db.cache

4) Start name server daemon.

Ensure network settings are correct:

```
% system-config-network

IP:         10.0.0.20
Subnet:     255.255.255.0
```

/usr/local/sbin/named

5) Modify network and nameserver settings:

Change hostname in /etc/sysconfig/network:

```
HOSTNAME=DNS-Slave.ara.com
```

Put nameserver and domain in /etc/resolv.conf:

```
domain ara.com
nameserver 10.0.0.10
nameserver 10.0.0.20
```

6) Set named to execute on startup

Create startup script (/root/named.sh):

```
# vi /root/named.sh

      #!/bin/sh

      /usr/local/sbin/named
      exit 0
```

```
# chmod +x named.sh
```

Modify startup file to include script (/etc/rc.local):

```
# vi /etc/rc.local
```

Add line: `/bin/sh /root/named.sh`

Install/Configure CentOS 5 VM Client:

Setup the following files, according to the Network Topology diagram in Figure 1.

/etc/resolv.conf
/etc/sysconfig/network
system-config-network

Install VMWare tools:

```
      mount –t iso9660 /dev/cdrom /mnt
      cp /mnt/VMwareTools-3.0.1-75314.tar.gz /tmp
      umount /dev/cdrom
      cd /tmp
      tar zxf VMwareTools-3.0.1-75314.tar.gz /tmp
```

Install/Configure Windows XP Client:

Setup the network configuration for the virtual adapter according to the Network Topology diagram in Figure 1.

Query for AAAA records:

Our first test will consist of querying the authoritative server, DNS-Primary.ara.com for AAAA records. Per RFC 4074, when an authoritative DNS server for a name holds an A record, but no AAAA record, it should return a response with an empty answer section and a response code (RCODE) of 0. Any other response code could cause multiple issues, as detailed in RFC 4074, and would be in violation of proper DNS functionality, as detailed in the same RFC. (1-3)

In our test network topology, the DNS server DNS-Primary.ara.com serves as the authoritative server for the ara.com domain. Using the *dig* tool, a request is sent for the AAAA record of WinXP.ara.com, a client on the ara.com domain for which DNS-Primary.ara.com is the authoritative server:

```
# dig @DNS-Primary.ara.com. WinXP.ara.com AAAA
```

This command asks the DNS server specified after the @ modifier, DNS-Primary.ara.com to look up the AAAA record for the network host WinXP.ara.com.

DNS-Primary name server is the authoritative server for the WinXP A record, but contains no AAAA record. This scenario recreates the possible error condition described previously. The goal of this test is to ensure the proper functionality of the BIND 9.5.0 implementation.

Windows XP Caching DNS Resolver Test:

RFC 4472 describes a condition whereby differing TTL values between arbitrary resource records (for our example, A and AAAA records) can cause caching resolvers, such as the one used in Window XP, to contain incomplete information. During this time period, connectivity issues may result.

The purpose of this test is to verify that the Windows XP resolver does or does not continue to cache arbitrary records for a network node after the expiration of a single record, thus resulting in incomplete cache information.

To prompt the WinXP.ara.com. resolver to request DNS information for the CentOS.ara.com. network host, enter CentOS.ara.com. into the address bar of the Internet Explorer web browser. The host address information can now be viewed in the DNS resolver cache:

Start > Run > “cmd”

```
 > ipconfig /displaydns
```

The DNS cache is queried until the TTL of one of the network hosts resource records expires, at which point it can be verified that the WinXP.ara.com. resolver either drops all records, or reissues a request for expired record, or does nothing.

Results and Data Quantification:

Query for AAAA records:

This test case consists of recreating the network topology described in the test setup section, and issuing the dig command to query the authoritative DNS server:

```
# dig @DNS-Primary.ara.com. WinXP.ara.com AAAA
```

The output from the dig query is as follows:

```
; <<>> DiG 9.3.4-P1 <<>> @DNS-Primary.ara.com. winxp.ara.com. AAAA
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16104
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION:
;winxp.ara.com.          IN    AAAA

;; AUTHORITY SECTION:
ara.com.           3600  IN    SOA  DNS-Primary.ara.com.  kayers.ara.com.
3 10800 3600 604800 3600

;; Query time: 1 msec
;; SERVER: 10.0.0.10#53(10.0.0.10)
;; WHEN: Tue Jul 22 12:21:25 2008
;; MSG SIZE rcvd: 86
```

The `status: NOERROR` section denotes an RCODE of zero, which means the query returned without error. Also, the `ANSWER: 0` section denotes the return of an empty answer section. The AAAA query returned a blank answer and no error codes.

Windows XP Caching DNS Resolver Test:

The results of this test consist of the output from the `ipconfig /displaydns` command at two time intervals after the Windows XP SP2 network host queries the DNS server for information for CentOS.ara.com.

At time interval one (60 seconds after the DNS query results are cached), there are two cache entries for CentOS.ara.com., one AAAA record, and one A record:

```
centos.ara.com
____________________________________________________
Record Name . . . . . : centos.ara.com
Record Type . . . . . : 28
Time To Live. . . . . : 40
Data Length . . . . . : 16
Section . . . . . . . : Answer
AAAA Record . . . . . : fed80::20c:29ff:fed3:7568


centos.ara.com
____________________________________________________
Record Name . . . . . : centos.ara.com
Record Type . . . . . : 1
Time To Live. . . . . : 240
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 10.0.0.22
```

At time interval two (121 seconds after the DNS query results are cache), there is only one cache entry for Centos.ara.com., an A record:

```
centos.ara.com
____________________________________________________
Record Name . . . . . : centos.ara.com
Record Type . . . . . : 1
Time To Live. . . . . : 179
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 10.0.0.22
```

Conclusions:

Query for AAAA records:

When the authoritative DNS server (DNS-Primary.ara.com.) for the network host CentOS.ara.com. was queried for an associated AAAA record, the server response included an RCODE of zero and an empty answer section. This response is in line with the expected functionality as explained in RFC 4074:

"Suppose that an authoritative server has an A RR but no AAAA RR for a host name. Then the server should return a response to a query for an AAAA RR of the name with the response code (RCODE) being 0 (indicating no error) and with an empty answer section… Such a response indicates that there is at least one RR of a different type than AAAA for the queried name, and the stub resolver can then look for A RRs." (Morishita and Jinmei 2)

Our AAAA record request using the CentOS 5.2 implementation of BIND 9.5.0 verifies that given this possible problem scenario, this specific version and configuration works properly.

Windows XP Caching DNS Resolver Test:

The DNS resolver built into Windows XP SP2 drops cached DNS query results after the TTL expires. This means that incongruous TTL values for RRs containing information for a specific network host will cause the WinXP.ara.com. to contain incomplete cache information. The Windows XP SP2 resolver neither re-queries the DNS server for the missing information, nor does it drop all associated records for that network host. No determination will be made regarding the performance or functionality ramifications of this condition. This test was only meant to observe the caching functionality of the Windows XP SP2 resolver.

References:

Durand, A., Ihren, J., & Savola, P. Request for Comments: 4472, Operational Considerations and Issues with IPv6 DNS. The Internet Society, 2006.

Huston, G. Request for Comments: 4159, Deprecation of “ip6.int”. The Internet Society, 2005.

Mockapetris, A. Request for Comments: 1035, Domain Names – Implementation and Specifications. The Internet Society, 1987.

Morishita, Y., Jinmei, T. Request for Comments: 4074, Common Misbehavior Against DNS Queries for IPv6 Addresses. The Internet Society, 2005.

Thomson, S., Huitema, C., et. al. Request for Comments: 3596, DNS Extensions to Support IP Version 6. The Internet Society, 2003.

6.) DHCPv6

Background Information

DHCPv6 allows for stateful autoconfiguration, a connection oriented server-client model, assigning non-link-local ipv6 addresses along with other information such as DNS to client devices. DHCPv6 uses the multicast address of FF02::1:2 for all DHCPv6 Relay Agents and servers or FF02::1:3 for only DHCPv6 servers. Furthermore, DHCPv6 uses UDP port 546 for client listening and UDP port 547 for server listening.

DHCPv6 Header Format

DHCPv6, as specified in RFC 3315, has a fixed header with a variable part for options. The header format is as follows:

Message Type:	1 byte
Transaction ID:	3 bytes
Options:	variable

Option fields can be expanded as defined in RFC 3315:

Option Code:	2 bytes
Option length:	2 bytes
Option Data:	variable

Relay Agent-Server Message Format

Relay Agents act as an intermediary for client and server messages if the client and server reside on different links.

Header Format:

Message Type:	1 byte	Type 12 = Relay forward Type 13 = Relay reply
Hop Count:	1 byte	Number of relays (32) that forwarded the message
Link Address:	16 bytes	Global address
Peer Address:	16 bytes	Address of client or relay
Options:	Variable	

DHCP Unique Identifier (DUID)

DUID's are used to identify clients and servers. RFC 3315 identifies three types of DUID's:

1. Link-Layer address plus time (DUID-LLT)
2. Vendor-specific unique ID based on enterprise number (DUID-EN)
3. Link-Layer address (DUID-LI)

Identity Association (IA)

An Identity Association is an object used by a server and the client to identify and manage a group of addresses. Each IA has an Identity Association Identification (IAID). Each client has at least one IA per interface.

A DHCP server chooses the configuration information for the IA according to the address allocation policies based on:

1. The link to which the client is connected
2. The DUID of the client
3. Other information provided by options
4. Other information provided by options, which have been added by Relay Agents.

Client-Server Communication

Clients use a multicast solicit message to find a DHCPv6 Server. If a client wants a specific server it will use the DUID in a server identifier server option (option type 2). The client will then receive one or more advertise messages in answer to its solicit message. If this happens the criteria for choosing a DHCP server is as follows:

1. The message with the highest server preference value is preferred.

2. If the server preferences are equal, it chooses the one with the preferred configuration.
3. The client may choose a message with a lower server preference value if it contains more appropriate configuration parameters.

The client has to perform Duplicate Address Detection (DAD) for each address allocated by the DHCP server. Duplicate Address Detection is in an integral part of the neighbor discovery process.

Typical DHCPv6 server-client communication begins with the client sending a solicit message, via UDP port 547, to the server or Relay Agent. The DHCPv6 server replies with an advertise message back to the client, via UDP port 546. Once the client has discovered the DHCPv6 server, it responds with a request message to obtain information. Finally, the DHCPv6 Server sends the reply message back to the client. (Hagan 2006).

This process can be shortened with the Rapid Commit option. With the rapid commit option the client sends a solicit message. The server replies with a reply message that also contains the rapid commit option. A couple of issues to look out for when using the rapid commit option are:

1. Depending on the configuration and the number of DHCP servers, it could result in wasted address space.
2. A situation where multiple DHCPv6 servers believe that they each assigned addresses to requesting clients.

Security Concerns with DHCPv6

Security should be a concern with all DHCPv6 devices. Common attacks to DHCPv6 devices are:

1. External, unknown DHCP servers allocating false addresses to DHCP clients.
2. Faulty or malicious DHCP servers in the intranet that assign false addresses or other false configuration info to clients.
3. Unknown, external clients that attach to the corporate network and receive internal addresses.
4. Intentional exhaustion of IP addresses by malicious clients, resulting in valid clients being unable to obtain a valid IP address and/or configuration options.
5. Malicious clients(s) transmitting such high volumes of requests that a DHCP server is unable to respond to valid requests.

Authentication and DHCPv6 services

The authentication of DHCP messages can be accomplished through the use of the Authentication option (option 11). Two protocols for authentication are defined in RFC 3315 section 21:

1. Delayed Authentication protocol.
2. Reconfigure Key Authentication protocol.

No authentication was used during this test.

Test Justification and Objectives:

A functionality test of DHCPv6 communication between a DHCPv6 server, Relay Agent, and client in networks where these devices all reside within different network segments using different IP protocol versions can provide insight into mechanics of DHCPv6 message passing in heterogeneous IP networks. Figure 1 illustrates this scenario and shows the DHCPv6 agents residing within IPv6 networks. The respective IPv6 network segments are connected via a 6to4 tunnel, traversing an IPv4 network cloud:

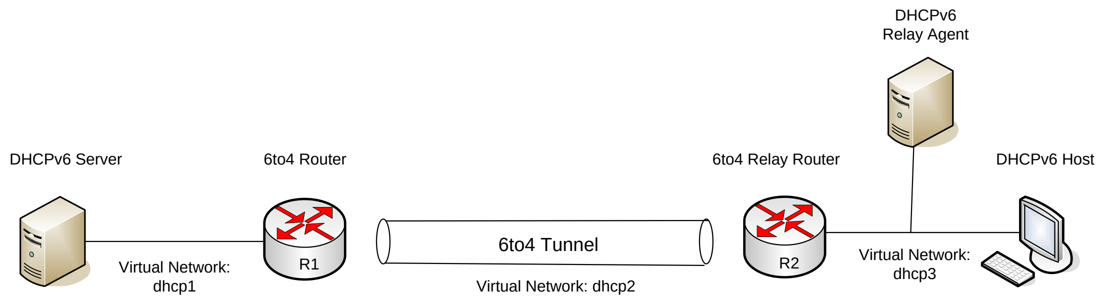


Figure 2: DHCPv6 Test Network Topology

Several potential issues arise with the network topology shown in figure 1. Questions relevant to this scenario are:

How are the UDP messages encapsulated and does this encapsulation introduce any problems with the clients ability to obtain DHCPv6 information from the DHCPv6 Server? Are there any issues with the DHCPv6 messages traversing a 6to4 tunnel? How are the DHCPv6 messages handled with large volumes of TCP/UDP traffic present in the network? With the UDP based solicit, advertise, request, and reply messages, how does packet loss impact server-client communications? How are the DHCPv6 UDP messages handled when high traffic volume and low-bandwidth exists between routers R1 and R2? Finally, how do the multicast addresses behave when forwarded through tunnels?

Test Protocols / Functionality:

DHCPv6

- UDP Port 546 – Client Port
  - Clients listen on UDP port 546 for their DHCP messages. Servers and Relays use this port as their destination port to clients
- UDP Port 547 – Server Port
  - DHCP Servers and Relays listen on UDP port 547 for DHCP messages. Clients use this port as their destination port.
- Multicast Address FF02::1:2
  - Multicast address FF02::1:2 is used by clients to reach all DHCP Servers and Relay Agents. Link-Scoped.
- Multicast Address FF02::1:3
  - Multicast Address FF02::1:3 is used by clients to reach DHCP Servers only. Link-Scoped.

Environment Variables (Hardware and Software Setup):

Hardware Specifications:

VMWare ESX Server 3.0.1 Virtual Machines:

6to4 Router1:	Centos5.2
6to4 Router2:	Centos5.2
DHCPv6 Server:	Centos5.2
DHCPv6 Client:	Centos5.2
DHCPv6 Relay Agent:	Centos5.2
Iperf Server:	Cenots5.2
Iperf Client:	Centos5.2

Software Specifications:

Iperf	version: 2.0.2 (http://dast.nlanr.net/Projects/Iperf/)
DHCPv6	version: 1.0.10-4.el5_2.2
Wireshark	version: 0.99.7

Network Diagrams of Test Plans:

Test Case #1

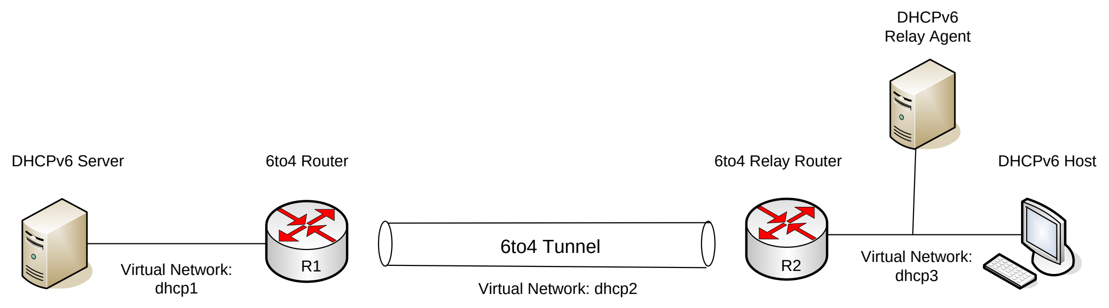


Figure 3: Test Case #1 Network Topology

Questions relevant to potential issues:
1. How are the UDP messages encapsulated and does this encapsulation introduce any problems with the Clients ability to obtain DHCPv6 information from the DHCPv6 Server?
2. Are there any issues with the DHCPv6 messages being tunneled from an IPv6 to an IPv4 network?
3. With regards to the UDP based solicit, advertise, request, and reply messages, how does packet loss impact server-client communications?
4. How do the multicast addresses behave when forwarded through tunnels?

Test Case #2

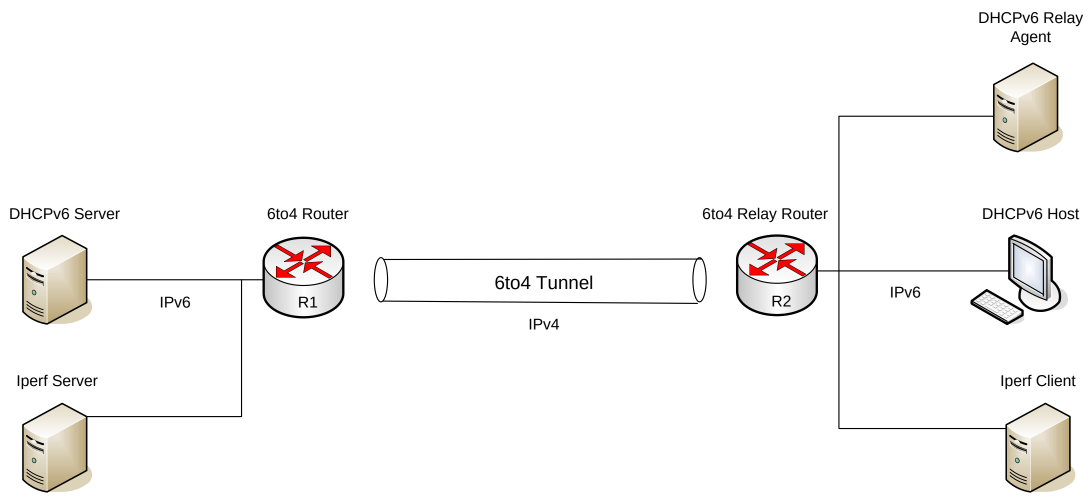


Figure 4: Test Case #2 Network Topology

Questions relevant to potential issues:
1. How are the DHCPv6 messages handled with large volumes of TCP/UDP traffic present in the network?

2. How are the DHCPv6 TCP/UDP messages handled in high volume, low-bandwidth conditions between routers R1 and R2?

Test Configuration:

IP Addresses:

DHCPv6 Server
eth0 (RA): 2002:db8:1:0:20c:29ff:fead:6c82/64

Router1
eth0: 72.16.0.1/30
eth1: 2002:db8:1::5/64
Tun6to4: 2002:4810:1:1000::1/64

Router2
eth0: 72.16.0.2/30
eth1: 2002:db8:2::5/64
Tun6to4: 2002:4810:2:1000::1/64

DHCPv6 Client
eth0 (RA): 2002:db8:2:0:20c:29ff:fe1c:f892/64
eth0 (dhcp): 2002:db8:2::1b/64

DHCPv6 Relay Agent
eth0 (RA): 2002:db8:2:0:20c:29ff:fe1a:b8e5/64

Use the following Script to convert ipv4 address to Hex:
```
$printf "%02x%02x:%02x%02x\n" 72 16 0 1
```

Convert 72.16.0.1 to Hex → 4810:0001
Convert 72.16.0.2 to Hex → 4810:0002

Allow IPv4 and IPv6 forwarding on Router1 and Router2 by editing /etc/sysctl.conf and adding:
```
net.ipv4.ip_forwarding = 1
net.ipv6.conf.all.forwarding = 1
```

Router1 Configuration:

Eth0: 72.16.0.1
Eth1: 2002:db8:1::5/64
Tun6to4: 2002:4810:0001:1000::1/64

Tunnel script:
```
#!/bin/sh
/sbin/ip tunnel add tun6to4 mode sit ttl 255 remote 72.16.0.2
local 72.16.0.1
/sbin/ip link set dev tun6to4 up
/sbin/ip addr add 2002:4810:0001:1000::1/64 dev tun6to4
/sbin/ip -6 addr add dev eth1 2002:db8:1::5/64
/sbin/ip route add 2002::/16 dev tun6to4
Exit 0
```

/etc/radvd.conf:
```
interface eth1
{
     AdvSendAdvert on;
     MinRtrAdvInterval 30;
     MaxRtrAdvInterval 100;
     Prefix 2002:db8:1:0::/64
          {
               AdvOnLink on;
               AdvAutonomous on;
               AdvRouterAddr off;
     };
};
```

Router2 Configuration:

Eth0: 72.16.0.2
Eth1: 2002:db8:2::5/64
Tun6to4: 2002:4810:0002:1000::1/64

Tunnel script:
```
#!/bin/sh
/sbin/ip tunnel add tun6to4 mode sit ttl 255 remote 72.16.0.1
local 72.16.0.2
/sbin/ip link set dev tun6to4 up
/sbin/ip addr add 2002:4810:0002:1000::1/64 dev tun6to4
/sbin/ip -6 addr add dev eth1 2002:db8:2::5/64

/sbin/ip route add 2002::/16 dev tun6to4

Exit 0
```

/etc/radvd.conf:
```
interface eth1
{
      AdvSendAdvert on;
      MinRtrAdvInterval 30;
      MaxRtrAdvInterval 100;
      Prefix 2002:db8:2:0::/64
            {
                  AdvOnLink on;
                  AdvAutonomous on;
                  AdvRouterAddr off;
      };
};
```

Relay Agent Configuration:

Eth0: 2002:db8:2:0:20c:29ff:fe1a:b8e5 -assigned by r2 router advertisement

Execute the relay command:
```
$dhcp6r –sf eth0+2002:db8:1:0:20c:29ff:fead:6c82
```

DHCPv6 Server Configuration:

/etc/dhcp6s.conf:
```
interface eth0
      Server-preference 255;
      Renew-time 60;
      Rebind-time 90;
      Prefer-life-time 130;
      Valid-life-time 200;
      Option dns_servers 2002:200::2 ibm.com;

      Link AAA {
            Pool{
                  Range 2002:db8:2:0::10/64 to 2002:db8:2:0::20/64;
                  Prefix 2002:db8:2:0::/64;
                  Relay 2002:db8:2:0::/64;
            };
      };
};
```

Execute on DHCPv6 Server to start the service:
```
$dhcp6s –f eth0 –c /etc/dhcp6s.conf
```

DHCPv6 Client:

EXECUTE on DHCPv6 client to start the service:
```
$dhcp6c –f eth0 –c /etc/dhcp6c.conf
```

Iperf Client:

Usage for 2.0.2:
```
$iperf –c host2 –V –w 5K –t 60
```

-c, host2
 remote host (SERVER)
-V,
 enable ipv6
-w,
 set the window size to 5Kbytes
-t,
 run for 60 seconds
-u,
 UDP

Iperf Server:

Usage for version 2.0.2:
```
$iperf –V –u
```

Other flags used:
-c, host2,
 remote host; the Iperf
-V,
 enable ipv6
-u,
 UDP only
-w
 Set the tcp window size

Test Procedures:

Testing of DHCPv6 Communications.

Gather the packet information from Wireshark at each step of the process:

1) Have Wireshark running on all devices (DHCPv6 Server, R1, R2, and Relay Agent).
2) Turn off the dhcp6c client and reboot the machine to clear all routing tables and IP addresses.
3) Start Wireshark on the client.
4) Start dhcp6c on the client. You should see the messages populating Wireshark.
5) On each device save all packet information as a text file.
6) Gather this Wireshark information. This will contain the entire process of the DHCPv6 Client communicating with the DHCPv6 Server.

Test Case #1

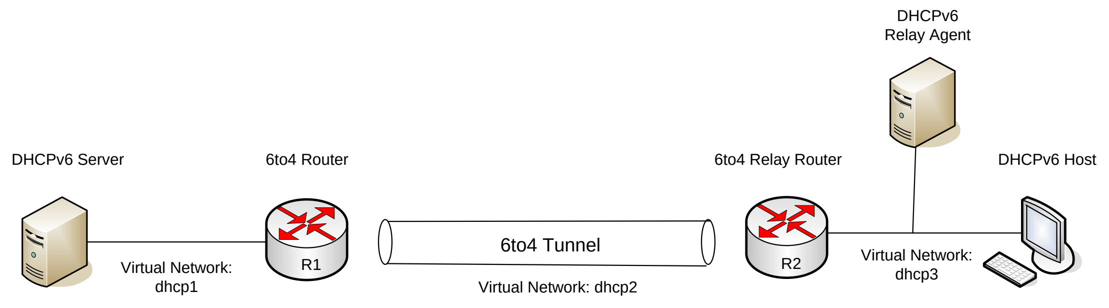


Figure 5: Test Case #1 Network Topology

1. How are the UDP messages encapsulated and does this encapsulation introduce any problems with the Clients ability to obtain DHCPv6 information from the DHCPv6 Server?

Inspection of the Wireshark output from the packets captured at Router1 and Router2 revealed no issues with the clients ability to obtain the UDP Reply, Renew, and Rebind messages from the DHCPv6 Server. There where no issues with the UDP messages traveling through the 6to4 tunnel from the client to the server and back again. The client communicates with the Relay Agent and the Relay Agent then forwards these client requests to the server. The Relay Agent has knowledge of where the DHCPv6 Server resides because the relay agent is started with the `dhcp6r -sf eth0+<IPADDR OF SERVER>` command. Where `<IPADDR OF SERVER>` is the IPv6 address of the DHCPv6 Server. Wireshark data confirms the packets are arriving at the DHCPv6 Server.

2. Are there any issues with the DHCPv6 messages being tunneled from 6to4?

No observable issues were found with the DHCPv6 UDP messages being tunneled from IPv6 to IPv4. The messages are getting through the tunnel as verified in the Wireshark packet data.

3. With the UDP based solicit, advertise, request, and reply messages, how does packet loss impact server-client communications?

The network link between Router1 and Router2 was severed and the Wireshark data was monitored on the client. Furthermore, the messages were received by Router1 from the DHCPv6 Server and by Router2 from the client. After turning the link between Router1 and Router2 "off" the client sends a Renew message to multicast address ff02::1:2 to the server that originally provided the address/information to extend the lifetimes on its IP address. After no response from the server, the client then sends the Rebind message to any server to extend the lifetime of its IP address. When no response is given to either the Renew or Rebind messages the client will then send a Solicit message to locate any available DHCPv6 servers. This happens until the client receives a reply from a DHCPv6 server. When the link between Router1 and Router2 was reestablished, the DHCPv6 messages communication continued as expected.

4. How do the multicast addresses behave when forwarded through tunnels?

The multicast messages never traverse the tunnel. The client sends out the multicast on its local link and multicast messages are received by the Relay Agent. The Relay Agent then encapsulates the messages and forwards these messages to the DHCPv6 Server. The Relay Agent is started with the `dhcp6r -sf eth0+<IPADDR OF SERVER>` command telling it to use the eth0 interface and specifying the IPv6 address of the DHCPv6 Server. The multicast messages are link-scoped multicast (FF02) therefore they are not routed. The Relay Agent handles all communication with the DHCPv6 server.

Test Case #2

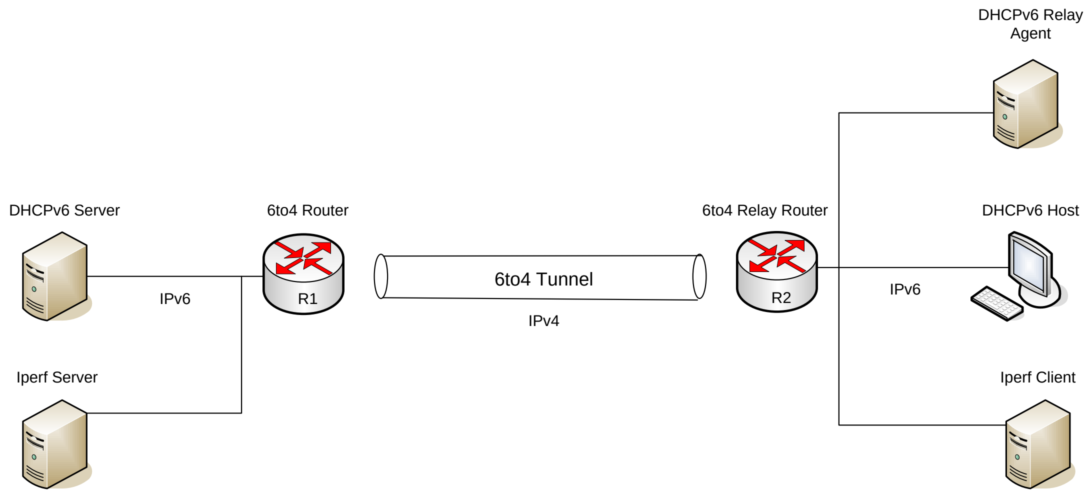


Figure 6: Test Case #2 Network Topology

1. How are the DHCPv6 messages handled with large volumes of TCP/UDP traffic present in the network?

The network was flooded with TCP and UDP data from the Iperf client to the Iperf server as shown in the diagram above. Wireshark was running on the DHCPv6 Client and the Renew and Reply messages were monitored during a 6 minute flood. There was no loss of the UDP Renew and Reply messages on the client and the messages arrived to the client every 60 seconds as specified in the DHCPv6 Server configuration file (/etc/dhcp6s.conf).

2. How are the DHCPv6 UDP messages handled in high volume, low-bandwidth conditions between routers R1 and R2?

The test modeled satellite and serial links from128Kbps to 1.544Mbps in increments of 128Kbps. No observable issues in these environments were found. The client sends the solicit multicast and the Relay Agent forwards this message to the DHCPv6 server. Regardless of the bandwidth setting, the messages are sent and received by all devices; however if the ESX virtual switch bandwidth limit set in VI Client is below the transfer rate of Iperf a Denial of Service condition occurs.

A burst size of 1 KB was used in this test.
*Note: ESX burst size is the amount of traffic that can be transferred when the bandwidth exceeds the peak setting before it is capped, independent of time.

Conclusions:
Dynamic Host Configuration Protocol version 6 (DHCPv6) is an important tool that provides stateful autoconfiguration to client network nodes. Typically, DHCPv6 clients initiate communication to a DHCPv6 server by sending a link-local multicast solicit message on its local-link to any listening DHCPv6 server. The DHCPv6 server responds to this solicit message with an advertise message indicating that it can provide IP addresses and DNS information. The client then sends a Request message requesting an IP address and DNS information. Finally, the server sends the Reply message with the IP address and DNS information. A rapid-commit option can used to make soliciting the server a two step process. These tests where conducted using the default four step process described above.

A Relay Agent can be placed between the client and server if the DHCPv6 server and client reside on different network segments. The Relay Agent sends the server a relay-forward message containing the encapsulated solicit and request messages that arrived from the client. The server will then respond to the Relay Agent with a relay reply containing the encapsulated advertise and reply messages for the client.

IPv6 migration will involve bringing IPv6-only networks online while communicating through legacy IPv4-only networks. This will involve the use a transition mechanism such as tunneling. This test set out to analyze the impact on DHCPv6 services when used in a tunneled environment.

The network used for this test consisted of two IPv6-only networks segmented by an IPv4-only network. A 6to4 tunnel was used to transmit the IPv6 data through the IPv4 network, as illustrated in Figure 1, and back again. Wireshark captured the packet data at each node.

The network between Routers 1 and 2 was flooded with both TCP and UDP data and bandwidth limitations put into place to model various networks such as satellite communication and serial links. Furthermore, the bandwidth was varied from 128 Kbps to 1.544 Mbps in 128 Kbps increments. There were no problems with the DHCPv6 communication in these environments and the client successfully exchanged information with the DHCPv6 server via the Relay Agent.

References:

Hagan, Silvia. IPv6 Essentials. Sebastopol: O'Reilly Media, Inc., 2006.

Droms, R., et al. Request for Comments: 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6). The Internet Society, 2003.

Toain, O., and R. Droms. Request for Comments: 3633, IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. The Internet Society, 2003.

7.) OSPFv3

Test Justification and Objectives:

Open Shortest Path First version 3 (OSPFv3) is a link-state routing protocol designed to provide fast convergence using Dijkstra's shortest path first algorithm. Many of the features supported in OSPFv2 such as electing a designated router, multiple area support, and flooding remain unchanged. The default Hello and Dead Intervals remain unchanged at 10 seconds and 40 seconds, respectively. A few key differences between OSPFv2 and OSPFv3 is support for the larger address size of IPv6, the use of a link-local unicast address as the OSPFv3 routers source address, and IPSec implementation. OSPFv3 will be instrumental in developing future IPv6 networks and proper functionality must be proven.

Virtualized environments are fast becoming the standard for network testing environments due to their ability to rapidly deploy virtual machines and create dedicated virtual networks. Virtualized networking environments by definition do not use physical routers, so virtual machines running software based routing protocol suites must be used in their place. These test environments must reliably recreate traffic typical of physical networks in addition to being able to communicate with physical networks in a standards-based fashion. Because of this, the software based routing protocols must comply with RFC standards.

Quagga is an IPv4/IPv6 routing software protocol suite available for UNIX type systems providing RIP, RIPng, OSPFv2, OSPFv3, and BGP-4+ routing protocols. This test will utilize the Quagga implementation of OSPFv3 and test this protocol for RFC compliance.

The first test will demonstrate the election process of a designated router (DR) and the impact on a network when this designated router fails.The election of a DR is based the values found in the Router ID and Priority fields (found in the Hello messages), using the following criteria:

1) The router with the highest priority setting becomes the designated router.
2) If there is a tie in the priority setting, the router with the highest Router ID (RID) becomes the designated router and usually the router with the second highest ID becomes the Backup Designated Router (BDR).
3) A router priority setting of zero indicates that a router will not be involved in the election process and will never become a designated or backup designated router.

Test two will show the impact of having mismatched Hello Intervals. OSPFv3 sends and receives multicast (ff02::5) Hello messages to discover other OSPF listening routers. These Hello messages are sent every ten (10) seconds by default. A mismatch in a Hello Interval should result in the adjacent OSPFv3 routers not becoming neighbors.

Finally, test three will reveal the fast convergence of OSPFv3. Disabling a single router interface during an active connection will cause the Hello messages to stop arriving through that interface. When the router does not receive a Hello message for the time specified by the Dead Interval (40 seconds), OSPFv3 interprets this as a link failure. When a link fails, a router will update its Link-State Advertisement (LSA) for that subnet to reflect the change. That router will then send an updated LSA to its neighbors and they will then send it to their neighbors. This process continues until all routers have an identical copy of the Link-State Database (LSBD). Dijkstra's algorithm is then used to recalculate the fastest routes.

Environment Variables (Hardware and Software Setup):

Hardware:

Physical ESX Server 3.0.1
- 2x Dual Core Xeon processors @ 3.0 GHz
- 16 GB RAM

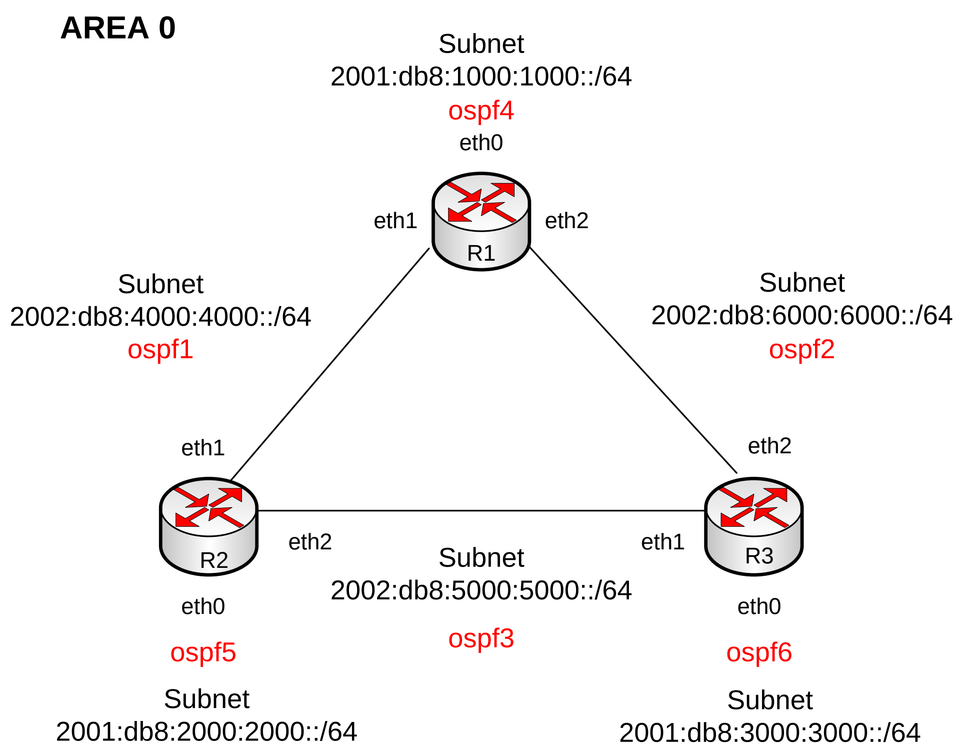
Three (3) Virtual OSPFv3 Linux Routers:
- CentOS 5.2 (assigned 4x processors, 256 MB RAM)

Two (2) Virtual Linux Hosts:
- CentOS 5.2 (assigned 4x processors, 256 MB RAM)

Software:

Quagga version 0.99.10 (http://www.quagga.net):
Quagga is an IPv4/IPv6 routing software protocol suite available for UNIX type systems providing RIP, RIPng, OSPFv2, OSPFv3, and BGP-4+.

Network Diagram:



OSPFv3 Router Configuration

Quagga Installation

Quagga was installed on routers R1, R2, and R3 using the following procedure:

1. Create the group and user `quagga` an each router
2. Grant write permissions to the user Quagga /var/run and /usr/local/etc:
```
$chown -R root.quagga /var/run
$chmod -R 770 /var/run
$chown -R root.quagga /usr/local/etc
$chmod -R 770 /usr/local/etc
```
3. Execute `$./configure && make && make install`

This installs the routing protocol configuration software in /usr/loca/etc/ directory.

IPv6 Addresses Assignment

Router R1
  eth0: 2001:db8:1000:1000::1/64
  eth1: 2002:db8:4000:4000::1/64
  eth2: 2002:db8:6000:6000::1/64

Router R2
  eth0: 2001:db8:2000:2000::1/64
  eth1: 2002:db8:4000:4000::2/64
  eth2: 2002:db8:5000:5000::1/64

Router R3
  eth0: 2001:db8:3000:3000::1/64
  eth1: 2002:db8:5000:5000::2/64
  eth2: 2002:db8:6000:6000::2/64

Host1
  eth0: 20001:db8:1000:1000:20c:29ff:fe8c:2d59/64 (from RAdvd)

Host2
  eth0: 20001:db8:3000:3000:20c:29ff:feba:162/64 (from RAdvd)

IPv6 Address Script

Scripts were created on routers R1, R2, and R3 to automatically assign these IPv6 addresses on a reboot. The script was placed in the /root directory and the `/etc/rc.local` file was modified to run this script at boot time.

Router R1
```
#!/bin/sh
/sbin/ip -6 addr add 2001:db8:1000:1000::1/64 dev eth0
/sbin/ip -6 addr add 2002:db8:4000:4000::1/64 dev eth1
/sbin/ip -6 addr add 2002:db8:6000:6000::1/64 dev eth2
exit 0
```

Router R2
```
#!/bin/sh
/sbin/ip -6 addr add 2001:db8:2000:2000::1/64 dev eth0
/sbin/ip -6 addr add 2002:db8:4000:4000::2/64 dev eth1
/sbin/ip -6 addr add 2002:db8:5000:5000::1/64 dev eth2
exit 0
```

Router R3

```
#!/bin/sh
/sbin/ip -6 addr add 2001:db8:3000:3000::1/64 dev eth0
/sbin/ip -6 addr add 2002:db8:5000:5000::2/64 dev eth1
/sbin/ip -6 addr add 2002:db8:6000:6000::2/64 dev eth2
exit 0
```

IP forwarding:
Allow ipv6 forwarding on routers R1, R2, and R3 by editing `/etc/sysctl.conf` and adding:

```
net.ipv6.conf.all.forwarding = 1
```

Turn off IPv4
Disable IPv4 addressing by not assigning any IPv4 addresses in

`/etc/sysconfig/network-scripts/ifcfg-ethx.`

OSPF6d Configuration

Router R1

```
/usr/local/etc/ospf6d.conf

hostname ospf6d@plant
password zebra
log stdout
service advanced-vty
debug ospf6 neighbor state
interface eth0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
interface eth1
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
interface eth2
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
router ospf6
  router-id 255.1.1.1
  redistribute static route-map static-ospf6
  interface eth0 area 0.0.0.0
  interface eth1 area 0.0.0.0
  interface eth2 area 0.0.0.0
ipv6 access-list access6 permit 2001:db8:1000:1000::/64
ipv6 access-list access6 permit 2002:db8:4000:4000::/64
ipv6 access-list access6 permit 2002:db8:6000:6000::/64
ipv6 prefix-list test-prefix seq 1000 deny any
route-map static-ospf6 permit 10
  match ipv6 address prefix-list test-prefix
  set metric-type type-2
  set metric 2000
line vty
  access-class access4
  ipv6 access-class access6
  exec-timeout 0 0
```

Router R2

```
/usr/local/etc/ospf6d.conf

hostname ospf6d@plant
password zebra
log stdout
service advanced-vty
debug ospf6 neighbor state
interface eth0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 3
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
interface eth1
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 3
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
interface eth2
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 3
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
router ospf6
  router-id 255.1.1.2
  redistribute static route-map static-ospf6
  interface eth0 area 0.0.0.0
  interface eth1 area 0.0.0.0
  interface eth2 area 0.0.0.0
ipv6 access-list access6 permit 2001:db8:2000:2000::/64
ipv6 access-list access6 permit 2002:db8:4000:4000::/64
ipv6 access-list access6 permit 2002:db8:5000:5000::/64
ipv6 prefix-list test-prefix seq 1000 deny any
route-map static-ospf6 permit 10
  match ipv6 address prefix-list test-prefix
  set metric-type type-2
  set metric 2000
line vty
  access-class access4
  ipv6 access-class access6
  exec-timeout 0 0
```

Router R3

```
/usr/local/etc/ospf6d.conf

hostname ospf6d@plant
password zebra
log stdout
service advanced-vty
debug ospf6 neighbor state
interface eth0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
Interface eth1
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
interface eth2
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id0
router ospf6
  router-id 255.1.1.3
  redistribute static route-map static-ospf6
  interface eth0 area 0.0.0.0
  interface eth1 area 0.0.0.0
  interface eth2 area 0.0.0.0
ipv6 access-list access6 permit 2001:db8:3000:3000::/64
ipv6 access-list access6 permit 2002:db8:5000:5000::/64
ipv6 access-list access6 permit 2002:db8:6000:6000::/64
ipv6 prefix-list test-prefix seq 1000 deny any
route-map static-ospf6 permit 10
  match ipv6 address prefix-list test-prefix
  set metric-type type-2
  set metric 2000
line vty
  access-class access4
  ipv6 access-class access6
  exec-timeout 0 0
```

Router Advertising Daemon (RAdvd)

Modify the `/etc/radvd.conf` and start radvd by invoking `$/etc/init.d/radvd` start or you can start radvd by issuing

```
$radvd –C /etc/radvd.conf.
```

Router R1

```
/etc/radvd.conf
interface eth0
{
      AdvSendAdvert on;
      MinRtrAdvInterval 30;
      MaxRtrAdvInterval 100;
      Prefix 2001:db8:1000:1000::/64
      {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

Router R2

```
/etc/radvd.conf
interface eth0
{
      AdvSendAdvert on;
      MinRtrAdvInterval 30;
      MaxRtrAdvInterval 100;
      Prefix 2001:db8:2000:2000::/64
      {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

Router R3

```
/etc/radvd.conf
interface eth0
{
      AdvSendAdvert on;
      MinRtrAdvInterval 30;
      MaxRtrAdvInterval 100;
      Prefix 2001:db8:3000:3000::/64
      {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

Test Procedures:

Test #1: Designated Router (DR) failure test

1. Configure R2 to have a higher priority (3) than R1 and R3. R2 and R3 will have the same priority.
2. Enable OSPFv3 on all router interfaces on R2, R3, and R1. R2 should be started first followed by R3 and R1.
3. Configure all routers to have the same Hello Interval (10 seconds) and Dead Interval (40 seconds).
4. Disable OSPFv3 on R2.
5. Capture the packets in the network at R1 and R3 and verify that a new DR and BDR are elected.
6. Verify the new DR and BDR on R1 and R3 and record neighbor data.
7. Verify Host1 and Host2 can communicate after R2 OSPFv3 is disabled.

Test #2: Hello Mismatch

1. Configure all three routers to have a Dead Interval of 40 seconds and Area 0.0.0.0.
2. Configure R2 to have a Hello Interval of 30 seconds. R1 and R3 will have a default Hello Interval of 10 seconds.
3. Enable OSPFv3 on all router interfaces on R1, R2, and R3. R2 should be started first followed by R3 and R1.
4. Capture Wireshark data from routers R1, R2, and R3.
5. Observe that Host1 can ping Host2.

Test #3: Interface Failure

1. Configure R3 to have a higher priority (3) than R1 and R3. Routers R2 and R3 will have the same priority.
2. Enable OSPFv3 on all router interfaces on R1, R2, and R3. R2 should be started first followed by R3 and R1.
3. With Host2 pinging Host1 via R3's eth2 interface, disable the eth2 interface on R3.
4. Observe the Wireshark output on R2.
5. Verify the ping continues to Host1 after the alternate route is used.

Results:

Test #1: Designated Router Election and Failure of the DR

Network Topology

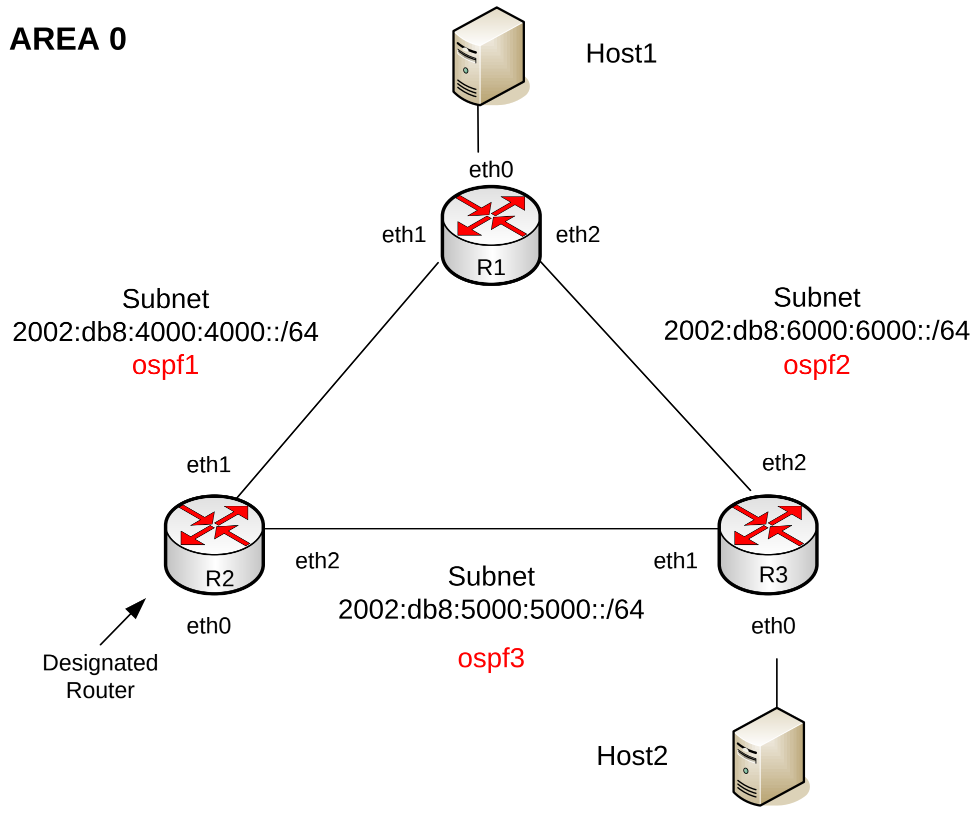


Figure 7: OSPFv3 Designated Router

1. Set the OSPFv3 router priority in ospf6d.conf
R1 priority = 1 on all interfaces
R2 priority = 3 on all interfaces

R3 priority = 1 on all interfaces

2. Turn Wireshark on for each router interface

3. Start ospf6d on R1, R2, and R3. The Wireshark output verifies the Link-State Advertisement (LSA) and Link-State Database (LSDB) exchange between the three OSPFv3 routers. The routers become fully adjacent after a few seconds. R2 wins the designated router election due to its higher priority setting.

Router R1
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.2	3	00:00:31	Full/DR	00:02:28	eth1 [BDR]
255.1.1.3	1	00:00:33	Full/DR	00:02:31	eth2 [BDR]

Router R2
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.1	1	00:00:36	Full/BDR	00:03:09	eth1 [DR]
255.1.1.3	1	00:00:32	Full/BDR	00:02:57	eth2 [DR]

Router R3
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.2	3	00:00:32	Full/DR	00:01:45	eth1 [BDR]
255.1.1.1	1	00:00:38	Full/BDR	00:01:20	eth2 [DR]

4. Turn off OSPFv3 on R2 and record the Wireshark output.

Router R1
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.3	1	00:00:35	Full/DR	01:09:40	eth2 [BDR]

Router R3
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.1	1	00:00:37	Full/BDR	01:06:32	eth2[DR]

After OSPFv3 is disabled on R2, R3 then becomes the new designated router. With a tie in the priority settings of R1 and R3, the Router ID is used. A router with a higher RID will become the new designated router. Thus, R3 is the new designated router.

Host1 and Host2 can successfully communicate after R2 is disabled as verified by an ICMPv6 message exchange.

Test #2: Hello Mismatch

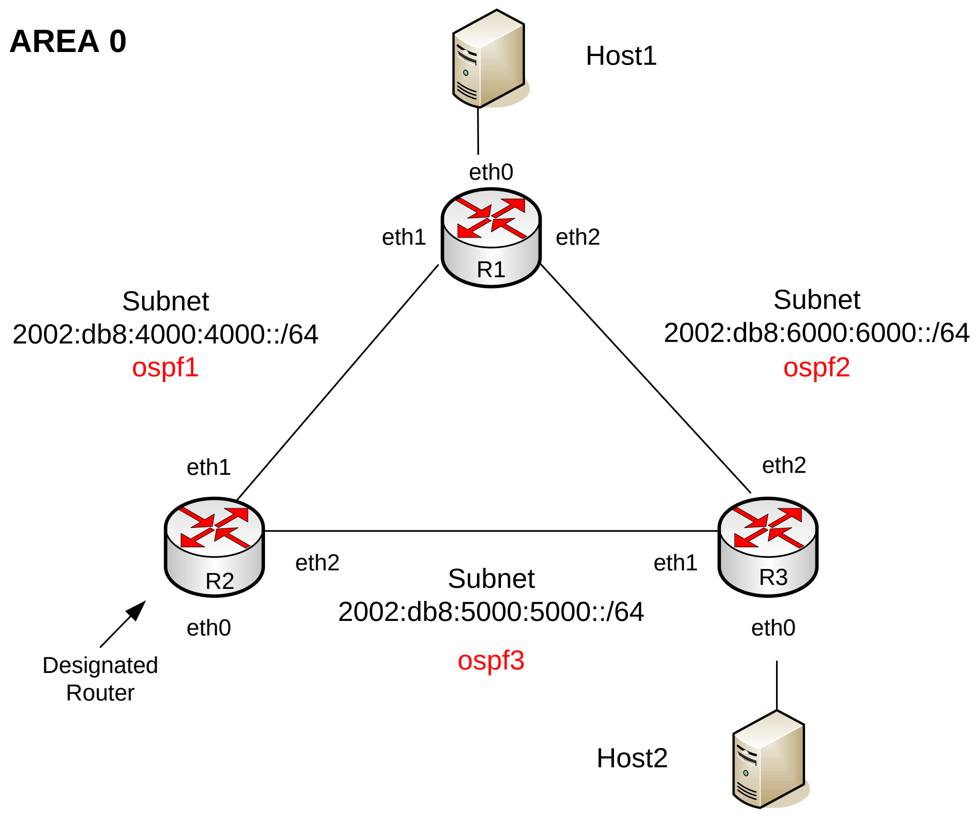


Figure 8: OSPFv3 Hello Mismatch

1. Configure R2 to have a Hello Interval of 30 seconds. Configure R1 and R3 to have a 10 second Hello Interval.

2. Start Wireshark and capture output.

3. Record OSPFv3 data

Router R1
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.3	1	00:00:40	Full/DR	00:01:30	Eth2 [BDR]

Router R3
```
$telnet 127.0.0.1 2606
#show ipv6 ospf6 neighbor
```

Neighbor ID	Priority	Deadtime	State/IFstate	Duration	I/F [State]
255.1.1.1	1	00:00:38	Full/BDR	00:00:59	Eth2 [DR]

As expected, R2 did not become neighbors with either R1 or R3 due to the Hello interval mismatch. Host1 and Host2 can successfully communicate after R2 is disabled as verified by an ICMPv6 message exchange.

Test #3: Interface Failure

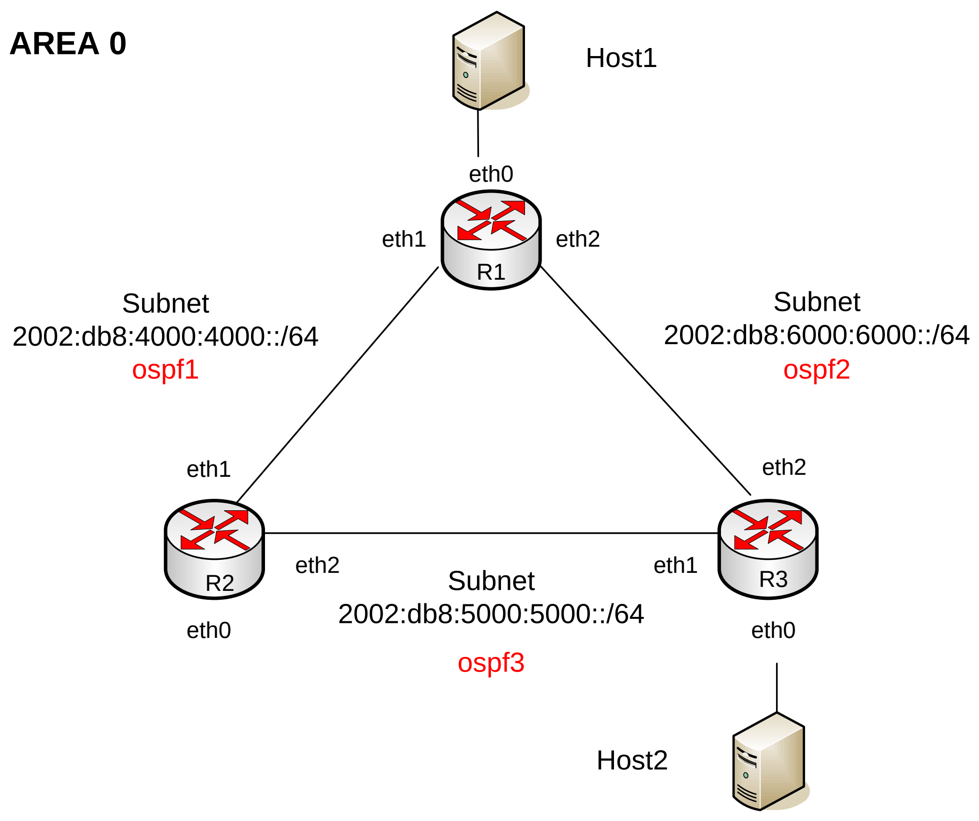


Figure 9: OSPFv3 R3 Interface Failure

1. Set the OSPFv3 router priority in ospf6d.conf
R1 priority = 1 on all interfaces
R2 priority = 3 on all interfaces
R3 priority = 1 on all interfaces

2. Turn Wireshark on for each router interface

3. Begin pinging Host1 from Host2. Disable interface eth2 on R3.

ICMPv6 output from Host2:
64 bytes from 2001:db8:1000:1000:20c:29ff:fe8c:2d59 icmp_seq=16 ttl=62 time=1.95ms

Disabled eth2 interface on R3. ICMPv6 messages stop.

Convergence time = 32 seconds.

ICMPv6 messages continue after the alternate route was calculated.

Conclusions:

The tests conducted in this document set out to observe and verify RFC compliance of the Quagga implementation of OSPFv3.

The first test demonstrated the election and failure of a designated router and the assignment of a new designated and backup designated router. A single OSPFv3 router R2 was given a higher priority setting than its neighbors such that router R2 would win the designated router election. Once all of the routers in the network fully converged, they shared an identical copy of the link-state database. The designated router R2 was then disabled. As expected, a new designated router emerged as specified in RFC 2740. This test verified RFC compliance.

The second test showed the impact of having an OSPFv3 router R2 with a mismatched Hello timer. Hello timers are used by OSPFv3 routers to dynamically discover other OSPFv3 routers. The default Hello time is 10 seconds. A single OSPFv3 router R2 was given a Hello time setting of 30 seconds while the other two routers kept the default setting of 10 seconds. R2 did not become a neighbor with the other routers, thus demonstrating RFC compliance.

The third and final test illustrated the impact of having an OSPFv3 interface disabled. Having one interface lose connectivity will cause the dead interval to be reached on that interface's subnet. Once this happens, all the routers update their link-state databases and calculate new routes. A continuous ping was issued from Host2 to Host1. The best route from R3 to R2 was via the eth2 interface on R3. The eth2 interface on R3 was then disabled causing a stop in the ping. After 32 seconds the route to Host1 went through R2 via the R3 eth1 interface and the ping resumed.

The Quagga implementation of OSPFv3 performed according to RFC 2740 guidelines during the three tests performed and described in this document. While this does not prove full RFC compliance it does provide a discrete set of proofs for RFC compliance with regards to DR failure, mismatch Hello timers, and interface disabling.

8.) RIPng

Test Justification and Objectives:

RIPng, as defined in RFC 2080, is a distance-vector routing protocol supported by both Cisco IOS and Quagga software routing protocol suite. Quagga is an IPv4/IPv6 routing software protocol suite available for UNIX like systems providing RIP, RIPng, OSPFv2, OSPFv3, and BGP-4+ support. RIPng first became available in the Cisco IOS version 12.3(22a).

RIPng originates from RIPv2, but is not an extension of RIPv2. RIPng and RIPv2 share many similarities such as the same timers, message types, and processes. The default update time of 30 seconds is still incorporated in RIPng as well as the 180 second timeout period, 180 second holddown timer, and the 120 second garbage-collection timer. Furthermore, RIPng uses the same maximum hop-count metric of 16.

One major difference between RIPng and RIPv2 is authentication. RIPng does not provide any mechanisms for authentication; rather RIPng relies on the features built into the IPv6 protocol.

RIPng sends and receives its messages on UDP port 521. There is no set message size RIPng. Moreover, the message size only depends on the link MTU. RIPng multicasts the Request and Response messages to address FF02::9.

RIPng implementations in software such as the Quagga networking stack, and the implementations found in Cisco hardware must be interoperate per RFC specifications. The tests conducted within this document attempt to functionally test the capabilities of the Quagga routing software protocols as they interact with a network segment containing Cisco routers.

This document details the configuration of two Cisco 871 series routers, router1 and router2, and one Centos5 router, router3, with RIPng enabled on each router's interface. A node is placed on each routers LAN interface to test each router's ability to exchange route information with and without link failures.

Host1 is a Windows XP SP2 client residing on subnet 2001:db8:3000:3000::/64. Host2 is a Linux client residing on subnet 2001:db8:2000:2000::/64. Finally, there is a Linux Apache webserver on subnet 2001:db8:1000:1000::/64. Figure 1, found in Section 3.0 illustrates this network topology.

Environment Variables (Hardware and Software Setup):

Hardware:

(2) Cisco 871 Series Routers: c870-advipservicesk9-mz.124-15.T3.bin
(1) Virtual Centos 5 Linux Router
(1) Virtual Windows XP SP2 Host1
(1) Virtual Centos 5 Host2
(1) Virtual Centos 5 Web Server

Software:

Quagga version 0.99.10 (http://www.quagga.net)

> Note: Quagga is an IPv4/IPv6 routing software protocol suite available for UNIX like systems providing RIP, RIPng, OSPFv2, OSPFv3, and BGP-4+.

Apache: httpd-2.2.3-11.el5_1.centos.3
Firefox (Linux): 1.5.0.12
Firefox (Windows): 3.0.1
Wireshark: 0.99.7, 1.0.0

Network Diagram

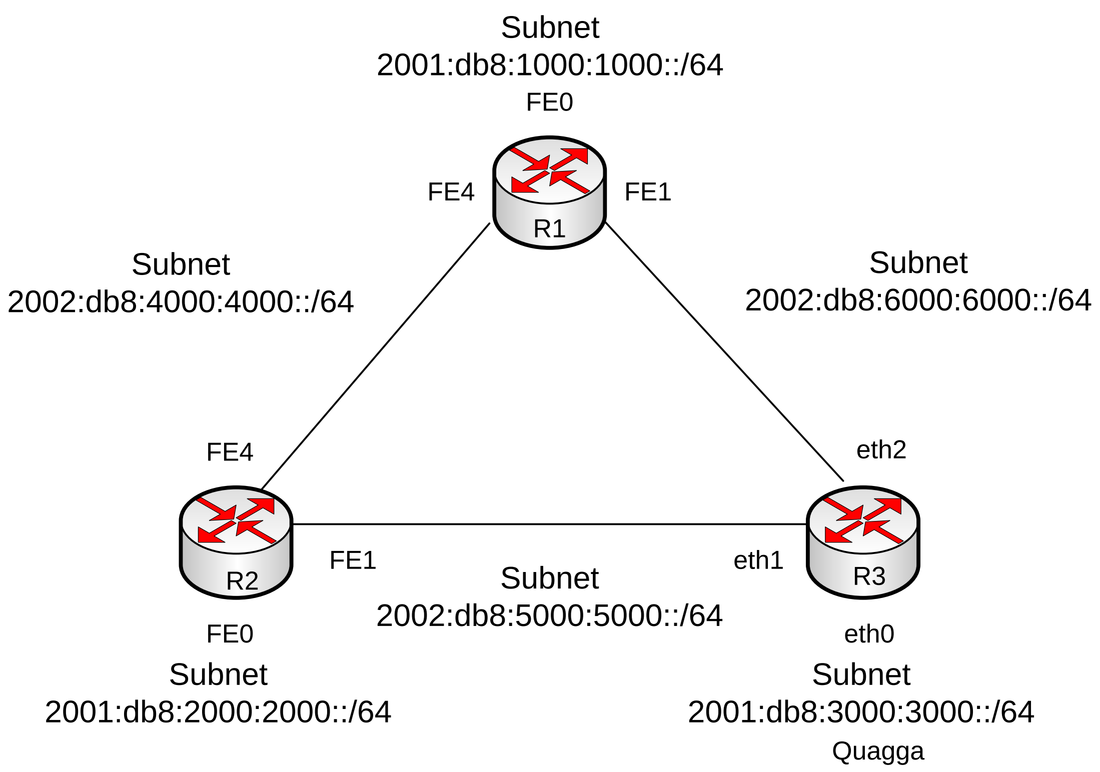


Figure 10: General RIPng Topology.
Routers 1 & 2 Are Cisco 871 Series Routers.
Router 3 is a Centos 5 Router

RIPng Router Configuration

QUAGGA Installation

Quagga was installed on router3 using the following procedure:
1. Create the group and user `quagga` an each router
2. Grant write permissions to the user Quagga /var/run and /usr/local/etc:

```
$chown -R root.quagga /var/run
$chmod -R 770 /var/run
$chown -R root.quagga /usr/local/etc
$chmod -R 770 /usr/local/etc
```

3. Execute `$./configure && make && make install` to install the Quagga software

This installs the routing protocol configuration files in the /usr/local/etc/ directory.

IPv6 Addresses Assignment

Router1
```
   FE0: 2001:db8:1000:1000::1/64
   FE4: 2002:db8:4000:4000::1/64
   FE1: 2002:db8:6000:6000::1/64
```

Router2
```
   FE0: 2001:db8:2000:2000::1/64
   FE4: 2002:db8:4000:4000::2/64
   FE1: 2002:db8:5000:5000::1/64
```

Router3
```
   eth0: 2001:db8:3000:3000::1/64
   eth1: 2002:db8:5000:5000::2/64
   eth2: 2002:db8:6000:6000::2/64
```

Windows XP Host1
```
    eth0: 20001:db8:3000:3000:384f:763d:8a86:1a5c/64 (from RAdvd)
```

Centos Host1
```
    eth0: 2001:db8:2000:2000:208:74ff:fe48:ba5a/64  (from RAdvd)
```

Centos Webserver
```
    eth0: 2001:db8:1000:1000:210:18ff:fe03:e4b2/64 (from RAdvd)
```

IPv6 Configuration

Router1

Set hostname:
```
(config)hostname router1
```

Manually configure ipv6 address on FE4 (FE4: 2002:db8:4000:4000::1/64):
```
(config)ipv6 unicast-routing
(config-if)ipv6 address 2002:db8:4000:4000::1/64
(config-if)no shut
```

By default, interfaces FE0-FE3 are in the default vlan 1. To change FE0 to vlan 2 do the following:
```
(config)interface fastethernet 0
(config-if)switchport mode access
(config-if)switchport access vlan 2
```

To Change FE1 to vlan 3 do the following:
```
(config)interface fastethernet 1
(config-if)switchport mode access
(config-if)switchport access vlan 3
```

Add IPv6 address for vlan 2:
```
(config)interface vlan 2
(config-if)ip address 2001:db8:1000:1000::1/64
(config-if)no shut
```

Add ipv6 address for vlan 3:
```
(config)interface vlan 3
(config-if)ip address 2002:db8:6000:6000::1/64
(config-if)no shut
```

Router2

Set hostname:
```
(config)hostname router2
```

Manually configure IPv6 address on fe4 (FE4: 2002:db8:4000:4000::2/64):
```
(config)ipv6 unicast-routing
(config-if)ipv6 address 2002:db8:4000:4000::2/64
(config-if)no shut
```

By default, interfaces FE0-FE3 are in the default vlan 1. To change FE0 to vlan 2 do the following:
```
(config)interface fastethernet 0
(config-if)switchport mode access
(config-if)switchport access vlan 2
```

To change FE1 to vlan 3 do the following:
```
(config)interface fastethernet 1
(config-if)switchport mode access
(config-if)switchport access vlan 3
```

Add IPv6 address for vlan 2:
```
(config)interface vlan 2
(config-if)ip address 200:db8:2000:2000::1/64
(config-if)no shut
```

Add IPv6 address for vlan 3:
```
(config)interface vlan 3
(config-if)ip address 2002:db8:5000:5000::1/64
(config-if)no shut
```

Router3

A script was created on router3 to automatically assign these IPv6 addresses upon reboot. The script was placed in the /root directory and the `/etc/rc.local` file was modified to run this script at boot time.

```
#!/bin/sh
/sbin/ip -6 addr add 2001:db8:3000:3000::1/64 dev eth0
/sbin/ip -6 addr add 2002:db8:5000:5000::2/64 dev eth1
/sbin/ip -6 addr add 2002:db8:6000:6000::2/64 dev eth2
exit 0
```

Enable IPv6 forwarding:
Allow ipv6 forwarding on router3 by editing `/etc/sysctl.conf` and adding:

```
net.ipv6.conf.all.forwarding = 1
```

Turn off IPv4:
Disable IPv4 addressing by not assigning any IPv4 addresses in:

```
/etc/sysconfig/network-scripts/ifcfg-ethx.
```

RIPng Configuration

Router1

Start RIPng:

```
(config-if)ipv6 rip ripng1 enable
```

Where `ripng1` can be any name. This command must be executed on each interface (vlan) with `ripng1` specified. The name does not matter, but the name must be the same across all interfaces in order for the interfaces to be part of the same RIPng process. Specifying different names will start unique instances of the RIPng process.

Router2

Start RIPng:

```
(config-if)ipv6 rip ripng1 enable
```

Router3

Modify /usr/local/etc/ripngd.conf:

```
hostname ripngd
password zebra
router ripng
 network eth0
 network eth1
 network eth2
 route 2001:db8:3000:3000::/64
 route 2002:db8:5000:5000::/64
 route 2002:db8:6000:6000::/64
log stdout
```

Start ripngd:

```
$ripngd &
```

Router Advertising Daemon (RAdvd)

Modify the `/etc/radvd.conf` and start radvd by invoking `$/etc/init.d/radvd` start or you can start radvd by issuing

```
$radvd –C /etc/radvd.conf.
```

Router3

/etc/radvd.conf:

```
interface eth0
{
      AdvSendAdvert on;
      MinRtrAdvInterval 30;
      MaxRtrAdvInterval 100;
      Prefix 2001:db8:3000:3000::/64
      {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

Test Procedures:

Test #1
Interoperability test of Cisco RIPng and Linux Quagga RIPng with all nodes being able to ping all other nodes. Turn on all routers and let the network converge. Once convergence has been accomplished, each node should have connectivity with each other.

Test #2
When a Cisco router link fails does Quagga recognize the failure? Stream data from the webserver and compute the time required to re-route when there is a failed link.

The data stream will be from the webserver to host2. The connection will be through router2's FE4 interface. Disable router2's FE4 interface and compute the time required to re-route traffic through router3's eth1 interface back to the webserver.

Test#3
When a Quagga interface goes down does a Cisco router recognize the failure?

The data stream will be from the webserver to host1. The route will be through router3's eth2 interface. Disable router3's eth2 interface and compute the time required to re-route traffic through router2's FE1 interface back to the webserver.

Results:

Test #1

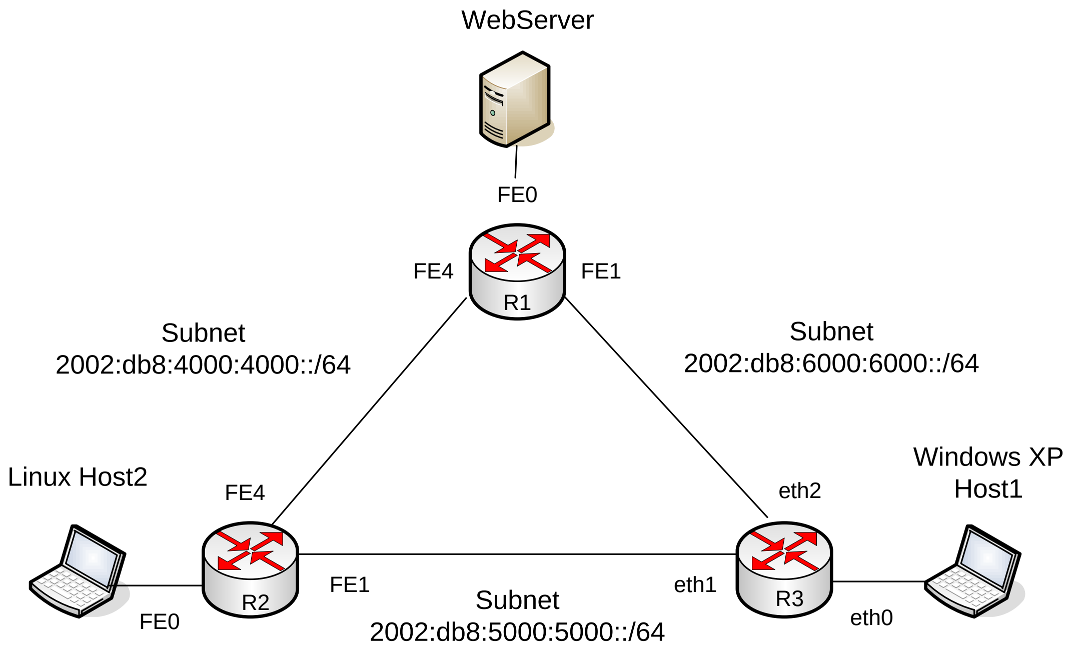


Figure 11: All Nodes Communicating

RIPng was configured and activated on each router's interface and allowed to fully converge. Each node then began a successful ICMPv6 echo request (PING) to the other nodes. This illustrated the interoperability of the hardware and software based RIPng implementations. The Quagga software successfully exchanged UDP Request and Response messages with the Cisco routers and all three routers learned new routes from each other.

The routing tables shown below show the learned routes provided by RIPng.

Router1:

Issue the following command to show the IPv6 routing:

```
router1#show ipv6 route
```

```
IPv6 Routing Table – 10 entries
Codes:  C – Connected, L – Local, S – Static, R – RIP, B – BGP
        U – Per-user Static route, M – MIPv6
        I1 – ISIS L1, I2 – ISIS LS, IA – ISIS interarea, IS – ISIS summary
        O – OSPF intra, OI – OSPF inter, OE1 – OSPF ext 1, OE2 – OSPF ext 2
        ON1 – OSPF NSSA ext 1, ON2 – OSPF NSSA ext 2
        D – EIGRP, EX – EIGRP external
C       2001:DB8:1000:1000::/64 [0/0]
         via ::, Vlan2
```

```
L         2001:DB8:1000:1000::1/128 [0/0]
           via ::, Vlan2
R         2001:DB8:2000:2000::/64 [120/2]
           via FE80::21E:7AFF:FEE4:EA58, FastEthernet4
R         2001:DB8:3000:3000::/64 [120/2]
           via FE80::201:2FF:FE8E:EE1C, Vlan3
C         2002:DB8:4000:4000::/64 [0/0]
           via ::, FastEthernet4
L         2002:DB8:4000:4000::1/128 [0/0]
           via ::, FastEthernet4
R         2002:DB8:5000:5000::/64 [120/0]
           via FE80::21E:7AFF:FEE4:EA58, FastEthernet4
           via Fe80::201:2FF:FE8E:EE1C, Vlan3
C         2002:DB8:6000:6000::/64 [0/0]
           via ::. Vlan3
L         2002:DB8:6000:6000::1/128 [0/0]
           via ::. Vlan3
L         FF00::/8 [0/0]
           via ::, Null0
```

Router2:

Issue the following command to show the IPv6 routing:

```
router1#show ipv6 route
```

```
IPv6 Routing Table – 10 entries
Codes:  C – Connected, L – Local, S – Static, R – RIP, B – BGP
        U – Per-user Static route, M – MIPv6
        I1 – ISIS L1, I2 – ISIS LS, IA – ISIS interarea, IS – ISIS summary
        O – OSPF intra, OI – OSPF inter, OE1 – OSPF ext 1, OE2 – OSPF ext 2
        ON1 – OSPF NSSA ext 1, ON2 – OSPF NSSA ext 2
        D – EIGRP, EX – EIGRP external
R         2001:DB8:1000:1000::/64 [120/0]
           via FE80::21E:F7FF:FE58:B2D3, FastEthernet4
C         2001:DB8:2000:2000::/64 [0/0]
           via ::, Vlan 2
L         2001:DB8:2000:2000::1/128 [0/0]
           via ::, Vlan 2
R         2001:DB8:3000:3000::/64 [120/2]
           via FE80::210:4BFF:FE2C:410E, Vlan 3
C         2002;DB8:4000:4000::/64 [0/0]
           via ::, FastEthernet4
L         2002:DB8:4000:4000::2/128 [0/0]
           via ::, FastEthernet4
C         2002:DB8:5000:5000::/64 [0/0]
           via ::, Vlan3
L         2002:DB8:5000:5000::1/128 [0/0]
           via ::, Vlan3
R         2002:DB8:6000:6000::/64 [120/2]
           via FE80::210:4BFF:FE2C:410E, Vlan 3
           via FE80::21E:F7FF:FE58:B2D3, FastEthernet4
L         FF00::/8 [0/0]
           via ::, Null0
```

Router3:

Issue the following command to show the IPv6 routing:

```
router1#show ipv6 ripng
```

```
Codes: R – RIPng, C – Connected, S – Static, O – OSPF, B – BGP
Sub-codes:
        (n) – normal, (s) – static, (d) – default, (r) – redistribute,
        (i) – interface, (a/S) – aggregated/Suppressed

Network                     Next Hop                    Via   Metric  Tag   Time
R(n) 2001:db8:1000:1000::/64  fe80::21e:f7ff:fe58:b2c9  eth2  2       0     02:55
R(n) 2001:db8:2000:2000::/64  fe80::21e:7aff:fee4:ea4e  eth1  2       0     02:58
R(s) 2001:db8:3000:3000::/64  ::                        self  1       0
R(n) 2001:db8:4000:4000::/64  fe80::21e:7aff:fee4:ea4e  eth1  2       0     02:58
R(s) 2001:db8:5000:5000::/64  ::                        self  1       0
R(s) 2001:db8:6000:6000::/64  ::                        self  1       0
```

Test #2

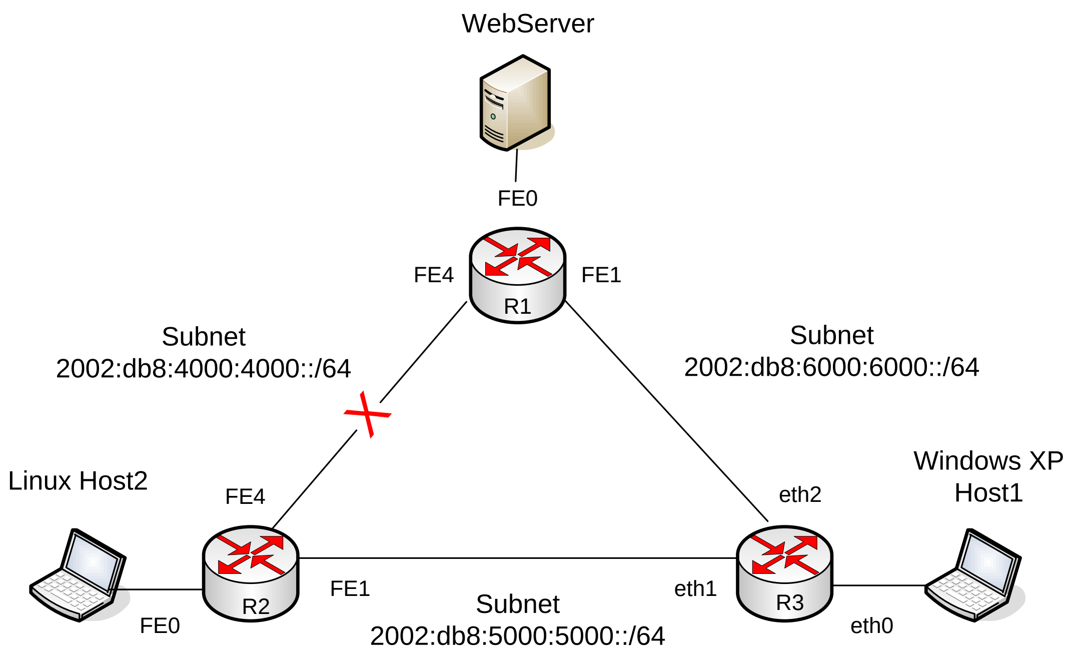


Figure 12: Subnet 2002:db8:4000:4000::/64 Fails

A Linux Apache webserver was configured and placed on subnet 2001:db8:1000:1000::/64. A single large file was then generated using the `yes` command as follows:

```
$cd /var/www/html/test
$yes ipv6test > largefile
```

The file `largefile` is approximately 1 GB in size and is used as the test file for hosts to download.

From host2, the `largefile` began downloading from the webserver. The link connecting routers 1 and 2 was disabled. Upon sensing a failed link, RIPng initiates a triggered update to inform its neighbors. After 15 seconds the routers had exchanged RIPng messages indicating a failed link and a new route was used to resume connectivity with the webserver. The new route path went through the subnet 2002:db8:5000:5000::/64 to the webserver and the connection continued.

Test #3

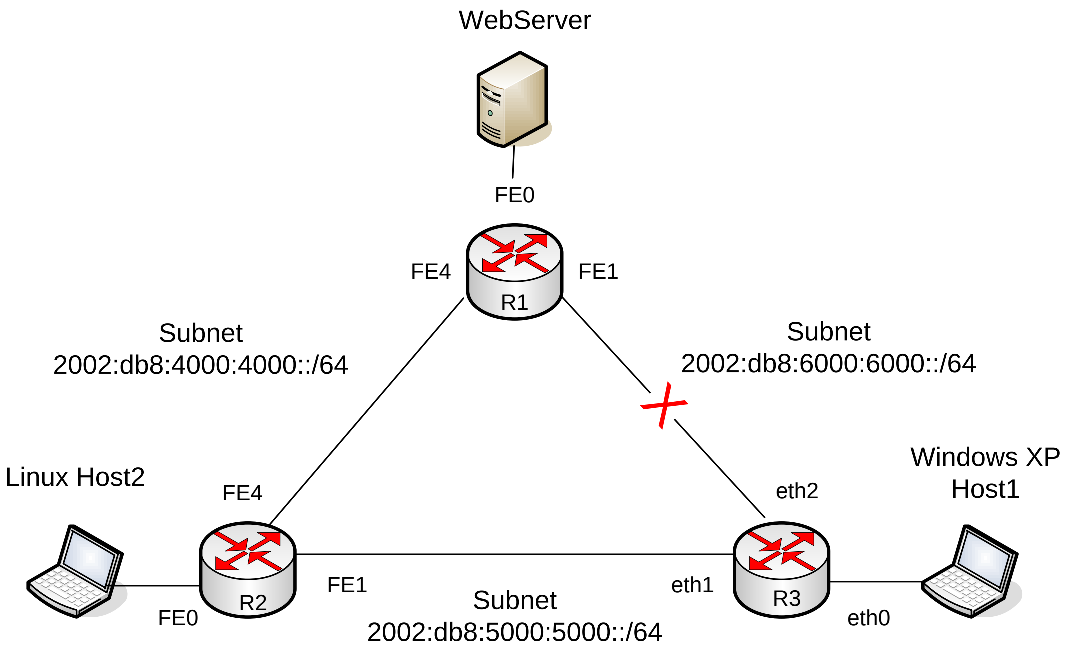


Figure 13: Subnet 2002:db8:6000:6000::/64 Fails

Test 3 is similar to test two. A Linux Apache webserver was setup and placed on subnet 2001:db8:1000:1000::/64. A single large file was then generated using the `yes` command as follows:

```
$cd /var/www/html/test
$yes ipv6test > largefile
```

The file `largefile` is approximately 1 GB in size and is used as the test file for hosts to download.

From host1, the `largefile` began downloading from the webserver. The link connecting routers 1 and 3 was disabled causing subnet 2002:db8:6000:6000::/64 to fail. Upon sensing a failed link, RIPng initiates a triggered update to inform its neighbors of the failure. After 28 seconds the routers had exchanged RIPng Request and Response messages indicating a failed link and a new route was used to resume connectivity with the webserver. The new route went through the subnet 2002:db8:5000:5000::/64 to the webserver.

Conclusion:

The interoperability of hardware and software implementations of the RIPng routing protocol can provide a means for network engineers to build and test simple network segments at a reduced cost compared to hardware only environments.

RIPng is based on RIPv2, but is not an extension of RIPv2. RIPv2 is a distance vector routing protocol that is slow to converge and consumes large quantities of bandwidth. However, RIPng can be useful in smaller network environments due to its ease of configuration and use.

Functional tests were conducted to show the interoperability of the Quagga RIPng software routing protocol in a Cisco router environment. The first test verified the exchange of RIPng Request and Response messages when a Linux router running the Quagga software is brought online in a Cisco network. The Quagga software performed according to RFC 2080 specifications. Network convergence occurred immediately in this small network.

Test 2 demonstrated the impact of a link failure during an active TCP transmission. An Apache webserver was placed on subnet 2001:db8:1000:1000::/64, per Figure 1. First, a Linux node on subnet 2001:db8:2000:2000::/64 began a file download from the webserver via interface FE4 on router2. Second, the FE4 subnet 2002:db8:4000:4000::/64 was then disabled and the route recalculated through interface FE1 on router2 to the webserver. Finally, a new route through router3 to the webserver was established. After 15 seconds the initial file transfer resumed.

Test 3 was similar to Test 2, but used a Windows XP host to initiate the file download from the webserver. The Windows host was on subnet 2001:db8:3000:3000::/64. Once the file transfer began the interface eth2 on router3 (subnet 2002:db8:6000:6000::/64) was disabled. After 28 seconds a new route was learned via interface eth1 on router3 to interface FE1 on router2. The file transfer then resumed successfully.

10.) MANET

Test Justification and Objectives:

Mobile Ad-hoc Networks (MANET) are self-healing, self-forming networks that are rapidly deployable and offer a decentralized network topology with no single point of failure. This test is designed to evaluate functionality of IPv6 in various MANET environments. These environments include: low-bandwidth interconnections, dynamically changing topologies, and heterogenous networks. The Optimized Link State Routing Protocol (OLSR), as defined in Request for Comment (RFC) document 3626, was used throughout the test design and execution. OLSR is based on the use of multipoint relays (MPR). A MPR is a node which is selected by its 1-hop neighbor, node X, to re-transmit all the broadcast messages that it receives from X, provided that the message is not a duplicate, and that the time to live field of the message is greater than one.

In OLSR, only nodes selected as such MPRs are responsible for forwarding control traffic, intended for diffusion into the entire network. Additional available link-state information may be utilized, e.g., for redundancy. In route calculation, the MPRs are used to form the route from a given node to any destination in the network. The protocol uses the MPRs to facilitate efficient flooding of control messages in the network. OLSR uses hop-by-hop routing, i.e., each node uses its local information to route packets. OLSR has the advantage of having routes immediately available when needed.

Test Protocols / Functionality:

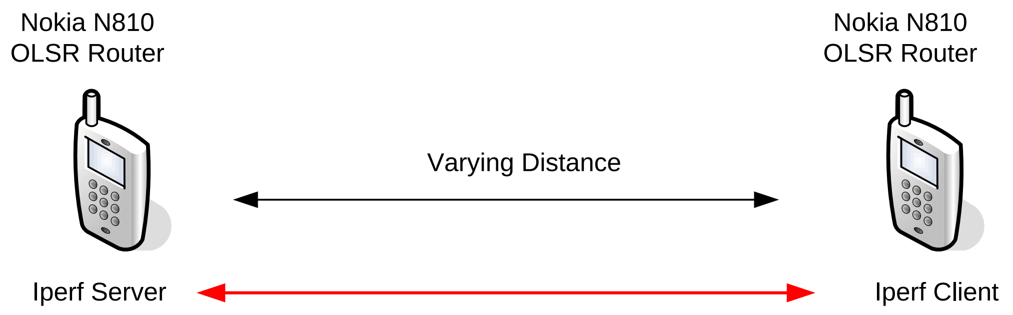
Optimized Link State Routing daemon - (OLSRd 0.5.5 for Maemo - http://www.mulliner.org/blog/blosxom.cgi/2008/07/28)

Environment Variables (Hardware and Software Setup):

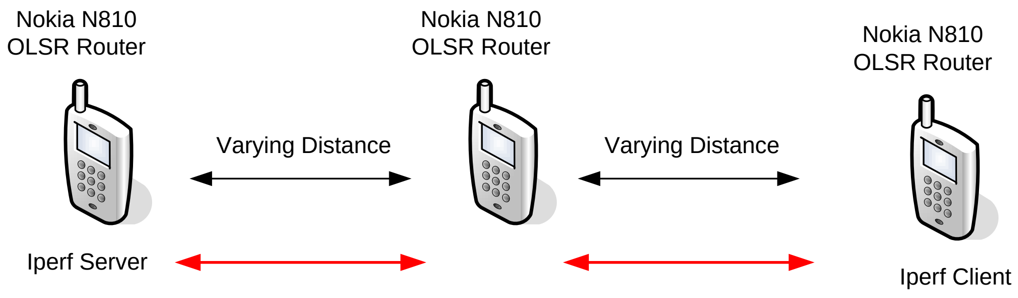
Nokia N810 tablet PCs x 4
Ubuntu 8.10 virtual machine (development workstation)
Linksys WRT-54 Wireless Router

Network Topology:

Baseline Test:



OLSRD Routing Test:



Test Procedures:

To establish a test network segment for the MANET test case, we setup four Nokia N810 tablets to run the OLSR daemon as well as the Iperf network performance monitoring tool.

Update Nokia N810 Operating System:

First the N810 tablets were flashed with Maemo Linux based OS2008 version 4.2008.36-5 using the Windows flashing tool provided by Nokia: http://nds1.nokia.com/files/support/global/phones/software/Nokia_Internet_Tablet_Software_Update_Wizard.exe

Various versions of the Maemo operating system can be found at this address, however you will need an N810 WLAN MAC address for access confirmation: http://tablets-dev.nokia.com/nokia_N810.php

Establish N810 Connectivity:

Next, the tablet connectivity package was installed to each tablet: http://maemo.org/downloads/product/raw/OS2008/maemo-pc-connectivity/?get_installfile

During the connectivity package installation, the user is queried for a root password. This is a key functionality as it allows you to become root on the N810 device, something you cannot do with the operating system until a 3rd party tool is installed. To gain root access on the tablet, establish an SSH connection with the localhost address and use the password setup during the connectivity package installation:

```
$ ssh root@localhost
```

The initial plan to interface with the N810 devices via SSH involved setting up USB networking, however we encountered some compatibility issues with the host operating system of the development machine running Ubuntu 8.10 and the USB networking functionality, so SSH connectivity with the devices was facilitated by adding them to the same wireless LAN network as the Ubuntu development machine.

The SSH client and server were installed to the development machine:

```
$ sudo apt-get install openssh-client openssh-server
```

SSHFS was then installed on desktop machine to allow for tablet file system modification via host computer:

```
$ sudo apt-get install sshfs
```

Cross Compile and Install IPerf:

The Iperf network performance monitoring tool must be cross-compiled for the ARM architecture the N810 devices use. To accomplish this, we used the Scratchbox2 cross compilation tool. The following procedure is based on the user guide provided by Maemo.org (http://maemo-sdk.garage.maemo.org/user-guide.html).

On the development machine, add the following line to /etc/apt/sources.list:

```
        deb http://maemo.sdk.garage.maemo.org/download/host ubuntu-
intrepid free
```

Update using apt-get, then install the maemo-sdk, maemo-tools (etch), and finally verify the maemo-tools toolchain 2005q3 is installed:

```
# apt-get update
# apt-get install maemo-sdk
# maemo-tools install etch
```

Verify installation of the 2005q3 toolchain installation:

```
# maemo-sdk list toolchains
2005q3
```

If the toolchain list does not include "2005q3" install it:

```
# maemo-tools install-toolchain 2005q3
```

Next install the Diablo ARM root-strap:

```
# maemo-rootstrap
```

This will bring up a menu, select "Option 2 . . . diablo4.1.1_armel For OS2008.36.5 (N810/N800)." Scratchbox2 is now installed and ready to cross compile. Download Iperf from http://sourceforge.net/projects/iperf and compile:

Extract iperf to directory ~/src/iperf-2.0.4/ then execute:

```
# sb2 -eR ./configure
# sb2 -eR make
# sb2 make install DESTDIR=/NokiaBin
# cd /NokiaBin/usr/local/bin/
# scp iperf root@192.168.6.187:/usr/local/bin
```

This procedure assumes the Nokia N810 device is at the IP address 192.168.6.187

Configure IPv6 on N810

The IPv6 module must be inserted on bootup. This was accomplished via a startup script (http://www.debian-administration.org/articles/28 used for reference):

```
#! /bin/sh
# /etc/init.d/insmodIPv6

/sbin/insmod /mnt/initfs/lib/modules/2.6.21-omap1/ipv6.ko

exit 0
```

After saving the script, the permissions were modified to make it executable. Next the update-rc.d command was used to add the script to the system startup:

```
# chmod 755 /etc/init.d/insmodIPv6
# update-rc.d insmodIPv6 defaults
 Adding system startup for /etc/init.d/insmodIPv6 ...
   /etc/rc0.d/K20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc1.d/K20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc2.d/S20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc3.d/S20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc4.d/S20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc5.d/S20insmodIPv6 -> ../init.d/insmodIPv6
   /etc/rc6.d/K20insmodIPv6 -> ../init.d/insmodIPv6
```

To set an IPv6 address on startup, a script was created (/etc/network/if-up.d/wlan0IPv6) and placed in the /etc/network/if-up.d/ folder. The scripts in this folder execute automatically when a network interface is brought up:

```
#! /bin/sh
# /etc/network/if-up.d/wlan0IPv6

ip -f inet6 addr add 2001:0db8:22::22/64 dev wlan0

exit 0
```

After placing the script in the correct folder, it must be made executable:

```
#chmod 755 /etc/network/if-up.d/wlan0IPv6
```

Nokia N810 OLSRd Configuration:

The OLSR daemon (version 0.5.5) was then installed to each device (http://www.mulliner.org/blog/blosxom.cgi/2008/07/28). Once downloaded, OLSRd installation is accomplished by double-clicking the "olsrd_0.5.5-1_armel.deb" icon in the Maemo interface. OLSRd is configured using the /etc/olsrd.conf file:

```
DebugLevel  2
IpVersion   6
FIBMetric   "flat"
ClearScreen yes

Hna 4
{
}

Hna 6
{
}

AllowNoInt  yes

IpcConnect
{
      MaxConnections    0
      Host        127.0.0.1
}

UseHysteresis           no
LinkQualityLevel  2
```

```
      LinkQualityWinSize      12
      Pollrate         3.0
      NicChgsPollInt          3.0
      TcRedundancy            2
      MprCoverage      3

      Interface "wlan0"
      {

            Ip6AddrType       global
            HelloInterval           2.0
            HelloValidityTime 20.0
            TcInterval        5.0
            TcValidityTime          30.0
            MidInterval       5.0
            MidValidityTime         30.0
            HnaInterval       5.0
            HnaValidityTime         30.0
      }
```

Linksys WRT54-GL OLSRd Configuration:

1. Load the ipv6 module

```
$insmod ipv6
```

2. assign an ipv6 address to the wireless adaptor

```
$ip -6 addr add 2001:db8:1:10::10/64 dev eth1
```

3. create the olsrd.conf

```
      DebugLevel  1
      IpVersion   6
      AllowNoInt  yes
      Pollrate    0.1
      TcRedundancy      2
      MprCoverage 1
      Willingness 4
      LinkQualityFishEye      0
      LinkQualityAging  0.1
      LinkQualityAlgorithm    "etx_fpm"
      LinkQualityDijkstraLimit      0 5.0
      UseHysteresis     no
      LinkQualityLevel  2
      LinkQualityWinSize      12
      Hna6
      {
      2001:db8:1:9000::1      64
      }
      Interface "eth1"
      {
            HelloInterval     5.0
            HelloValidityTime 90.0
            TcInterval  2.0
            TcValidityTime    270.0
            MidInterval 15.0
            MidValidityTime   90.0
      }
```

Network Tools Installation on Nokia N810:

The network tools nc, ping6, telnet, and traceroute were installed using the following reference: http://www.gossamer-threads.com/lists/maemo/users/15521:

From the Ubuntu development box, assuming the Nokia N810 device is using IP address 192.168.6.187:

```
      $ mkdir -p /tmp/n
      $ cd /tmp/n
      $ wget http://mummola.cs.tut.fi/n770/files/busybox_1.01-4.osso10-
ipv6.etc1_armel.deb
      $ ar x busybox_1.01-4.osso10-ipv6.etc1_armel.deb data.tar.gz
      $ tar xfz data.tar.gz
      $ mv ./bin/busybox ./bin/busybox2
      $ scp ./bin/busybox2 root@192.168.6.187:/bin/
```

From the N810 command line:

```
      $ ssh root@localhost
      # cd /usr/bin
      # ln -s /bin/busybox2 nc
      # ln -s /bin/busybox2 ping6
      # ln -s /bin/busybox2 telnet
      # ln -s /bin/busybox2 traceroute
      # chmod 755 /bin/busybox2
```

Results and Future Work

Initial stability testing of OLSR in conjunction with the Linksys router hardware yielded excellent results. The hardware and software were able to run without errors for a period of two hundred thirty eight hours with a configuration comprised of fourteen network nodes.

Range testing was also conducted on the Nokia N810 devices to serve as a baseline measurement for other testing purposes. During this testing it was found that range had a minimal impact on bandwidth, and in most cases communications would be completely lost rather than degrading to a certain point and then terminating.

Bandwidth and range comparisons were made for OLSR running in IPv6 versus running in IPv4 mode. There was no discernable difference in either range or bandwidth when comparing the different versions of the protocol.

Areas that would be of interest for future work and research would be the comparison of different styles of routing protocols. For instance, network convergence times in reactive networks versus proactive networks. Another area of interest would be the enabling of security features, for comparing performance characteristics, along with the addition and deletion of network nodes. Furthermore, testing a device's out-of-order packet processing capability, jitter tolerance, redundancy, and compression rates is also key to implementing fully functional MANETs for use with voice and video applications. These issues arise when video is sent over mobile meshing networks, e.g. to vehicles roving within the MANET.

11.) Testbed Knowledge Management and Information Portal

The ARA Embedded Web Technology (EWT) IPv6 Testbed, has developed a web portal to house a knowledge base of information relevant to IPv6 including Department of Defense (DoD) guidance, policy, and memorandums as well as industry white papers, technical documentation and any other information related to IPv6. All test procedures and documentation developed as part of the ARA EWT IPv6 Testbed will be published at this site. The portal site consists of 3 components – a wiki, a blog, and a forum.

11.1) Wiki

One of the components of theTestbed is a wiki site. The ARA EWT IPv6 Wiki is located at: https://www.ipv6community.org/wiki and makes use of MediaWiki (http://www.mediawiki.org) software. During the initial project specification, it was readily apparent that there was a need to not only manage information, but to make certain that information was easily accessible to decision makers. A wiki is a perfect medium for expressing ideas that are related in nature. A wiki also allows for the addition and modification of content quickly and easily by maintainers and members from the global information community.

The ARA EWT IPv6 Wiki is a central repository for all of the Testbed documentation and procedures, as well as an information store for material related to IPv6. All of the Testbed research and test cases are contained in this wiki. In addition, there is a plethora of IPv6 related information such as industry/vendor whitepapers, Internet Engineering Task Force (IETF) Request for Comment (RFC) documents, Deparment of Defense (DoD) memorandums, device configuration information, and information from other IPv6 researchers/developers as applicable.

11.2) Blog

The ARA EWT IPv6 Testbed research team uses a blog to provide periodic informal updates on status as well as furnish useful information on IPv6 implementation and deployment. The blog is a Wordpress product (http://wordpress.com/) and is located at https://www.ipv6community.org/blog.

11.3) Forum

The ARA EWT IPv6 Testbed also has a forum space set up for use by the general IPv6 global community. This forum space is intended for subject matter experts and other users to “meet” online and exchange ideas and information related to IPv6 deployment and implementation. The software that powers the forums is an open source program called Simple Machines Forum (www.simplemachines.org).

The forum is configured such that users are required to register and be approved by the forum administrator before being allowed to post. The forum has sections for different topics related to IPv6 including migration issues and questions, IPv6 testbed discussions, and threads for vendor-specific IPv6 information such as Cisco, Windows, Solaris, and Linux. The forum is located at https://www.ipv6community.org/forum.

12.) Summary

IPv6 is the emerging standard that is anticipated to replace the current, limited and outdated IPv4 protocol. IPv6 offers several benefits over IPv4. For example, regarding security, IPv6 supports several security features that were implemented directly into the protocol such as IP security (IPSec) and the removal of the checksum from the packet header. The implementation of IPSec is integrated directly into the IPv6 protocol, meaning that any implemntation of IPv6 will necessarily support it, whereas with IPv4, IPSec is independent from the protocol and may be vendor specific in regards to support for features. In addition to the security benefits, IPv6 also adds several key functionalities that IPv4 is unable to effeciently contend with. Network mobility, autoconfiguration, quality of service, the elimination of network address translation, multicast addressing, fixed packet header length, and packet option extensability are all benefits that IPv6 can offer over IPv4.

As IPv4 is phased out and IPv6 is phased in, there will need to be some intermediary steps and solutions to make sure that these protocols can operate along side and in conjunction with one another. IPv6 supports several transition mechanisms that allow for a transition of legacy networks and applications to the new protocol. Transition mechanisms including 4 over 6 tunnels, 6 over 4 tunnels, GRE tunnels, as well as dual network stacks can all be used to effieciently and successfully migrate to IPv6.

This document has outlined several transition mechanisms of IPv6 such as tunneling and dual stack operations, illustrated legacy application migration such as Microsoft Windows desktop and server applications, as well as taken a comparative look at IPv6 against IPv4 in several networking scenarios with regards to routing protocols.

In summarization, the ARA EWT IPv6 Testbed demonstrated functionality and researched applicability of IPv6 for several areas of interest to the DoN. The Testbed evaluated routing protocols such as OSFPv3, OLSR, and RIPng. The Testbed also compared key functionalities of IPv4 and IPv6 such as DNS and DHCPv6. Legacy application migration and support was evaluated with the migration and evaluation of IPv6 using Microsoft Server products, Microsoft Exchange products, and Microsoft Desktop operating system products. Additional platforms functionalities were demonstrated in several of the testcases such as Linus, Solaris, and variants of BSD.

13.) Further Considerations and Recommendations

The ARA EWT IPv6 Testbed demonstrates several benefits and applications of IPv6 technology that are relevant to the DoD and DoN. The Testbed is an ideal location for continuing research in areas of IPv6 that are still being evaluated by the government and private sectors. Key areas of evaluation include network mobility, wireless communications without line of sight, network topology convergence efficiency, secure man-portable communications with minimal or no configuration required and many others.

The transition to IPv6 will be by no means trivial, and it must be done in a judicious manner. The ARA EWT testbed is in an excellent position to leverage existing infrastructure and expertise for further research efforts. As the DoD moves closer to the migration deadline, the need for experience will be critical to the transition effort. ARA can offer a wellspring of information and practical experience for migrating to IPv6 on DoD networks.

Appendices

Appendix A –DHCPv6

Total UDP/TCP Traffic, Jitter, and Packet Loss for Test 2

All tests were executed with burst size:1 KB, UDP buffer size: 107 KB, Datagram receive size: 1470 bytes. The tests where run for 360 seconds allowing 5 DHCPv6 Renew and Rebind messages to cycle. Limiting bandwidth had zero impact on DHCPv6 message communication. There were no DHCPv6 messages lost during testing.

Connection from R1 to R2	TCP transfer Rate	TCP Data Transferred	UDP Data Transferred	UDP Transfer Rate	Jitter (ms)	Packet Loss	Messages Lost
Full Bandwidth: No Traffic	NA	NA	NA	NA	NA		0
Full Bandwidth: Max Traffic	150 Mbit/sec	6.27 GByte	45.0 MBytes	1.05 Mbits/sec	3.692	0/32101	0
128 Kbps Max Traffic	121 Kbits/sec	5.20 Mbyte	4.29 MBytes	99.9 Kbits/sec	28.366	0/3063	0
256 Kbps Max Traffic	239 Kbits/sec	10.3 Mbytes	8.50 Mbytes	198 Kbits/sec	21.132	58/6124 (0.95%)	0
384 Kbps Max Traffic	363 Kbits/sec	15.6 Mbytes	12.7 Mbytes	296 Kbits/sec	14.484	124/9185 (1.4%)	0
512 Kbps Max Traffic	483 Kbits/sec	20.8 Mbytes	16.8 Mbytes	391 Kbits/sec	11.051	275/12246 (2.2%)	0
640 Kbps Max Traffic	603 Kbits/sec	25.9 Mbytes	20.2 Mbytes	469 Kbits/sec	11.852	885/15300 (5.8%)	0
768 Kbps Max Traffic	727 Kbits/sec	31.2 Mbytes	23.0 Mbytes	547 Knit/sec	6.089	398/16839 (2.4%)	0
896 Kbps Max Traffic	847 Kbits/sec	36.4 Mbytes	25.2 Mbytes	588 Kbits/sec	8.215	363/18369 (2%)	0
1024 Kbps Max Traffic	973 Kbits/sec	41.8 Mbytes	33.3 Mbytes	775 Kbits/sec	5.781	3821/27554 (14%)	0
1152 Kbps Max Traffic	1.09 Mbits/sec	46.9 Mbytes	33.4 Mbytes	778 kbits/sec	5.409	675/24491 (2.8%)	0
1280 Kbps Max Traffic	1.21 Mbits/sec	52.0 Mbytes	37.6 Mbytes	875 Kbits/sec	5.506	765/27554 (2.8%)	0
1408 Kbps Max Traffic	1.34 Mbits/sec	57.3 Mbytes	40.3 Mbytes	939 Kbits/sec	12.825	1844/30614 (6%)	0
1544 Kbps Max Traffic	1.46 Mbits/sec	62.8 Mbytes	46.5 Mbytes	1.08 Mbits/sec	15.72	3533/36736 (9.6%)	0

Appendix B – OSPFv3

Quagga Usage

Once the Quagga software has been installed you can access the ospf6d command line interface by executing:

```
$telnet 127.0.0.1 2606
```

This brings you to a Cisco like prompt where you can issue commands and use debugging tools. A few useful commands are:

```
#show ipv6 ospf6 route
#show ipv6 ospf6 neighbor
#show ipv6 ospf6 database
```

Use the ? after a command to list available options.

Appendix C – RIPng

Quagga Usage

Once the Quagga software has been installed you can access the ospf6d command line interface by executing:

```
$telnet 127.0.0.1 2603
```

This brings you to a Cisco like prompt where you can issue commands and use debugging tools. A few useful commands are:

```
#show ipv6 ripng
#show ipv6 ripng database
```

Use the ? after a command to list available options.

References

1. http://www.microsoft.com/technet/network/ipv6/ipv6faq.mspx

2. http://blogs.msdn.com/exchangefaqs/archive/2008/02/01/will-there-be-any-support-for-ipv6-in-exchange-2003.aspx

3. http://technet.microsoft.com/en-us/library/bb629624(EXCHG.80).aspx

4. http://technet.microsoft.com/en-us/library/aa997281(EXCHG.80).aspx

5. Morimoto, R., et al. Windows Server 2008 Unleashed. USA: Sams Publishing, 2008.

6. Morimoto, R., et al. Windows Server 2003 Unleashed. USA: Sams Publishing, 2006.

7. Mueller, J. P. Administering Windows Server 2008 Server Core. Indianapolis, Indiana: Wiley Publishing, Inc., 2008.

8. Redmond, T. Microsoft Exchange Server 2007 with SP1. USA: Elsevier, 2008.

9. Hagan, Silvia. IPv6 Essentials. Sebastopol: O'Reilly Media, Inc., 2006.

10. Droms, R., et al. Request for Comments: 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6). The Internet Society, 2003.

11. Toain, O., and R. Droms. Request for Comments: 3633, IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. The Internet Society, 2003.

12. Hagan, Silvia. IPv6 Essentials. Sebastopol: O'Reilly Media, Inc., 2006.

13. Odom, Wendell. CCNA ICND2. Indianapolis: Cisco Press, 2008.

14. Coltun, R., et al. Request for Comments: 2740, OSPF for IPv6. The Internet Society, 1999.

15. Optimized Link State Routing daemon (OLSRd 0.5.5 for Maemo - http://www.mulliner.org/blog/blosxom.cgi/2008/07/28)

16. Connectivity kit: http://maemo.org/downloads/product/raw/OS2008/maemo-pc-connectivity/?get_installfile

17. Nokia N810 Update software : http://nds1.nokia.com/files/support/global/phones/software/Nokia_Internet_Tablet_Software_Update_Wizard.exe

18. Maemo OSs: http://tablets-dev.nokia.com/nokia_N810.php

19. Scratchbox user guide: http://maemo-sdk.garage.maemo.org/user-guide.html

20. Iperf: http://sourceforge.net/projects/iperf

21. Busybox IPv6 tools: http://www.gossamer-threads.com/lists/maemo/users/15521

22. Durand, A., Ihren, J., & Savola, P. Request for Comments: 4472, Operational Considerations and Issues with IPv6 DNS. The Internet Society, 2006.

23. Huston, G. Request for Comments: 4159, Deprecation of "ip6.int". The Internet Society, 2005.

24. Mockapetris, A. Request for Comments: 1035, Domain Names – Implementation and Specifications. The Internet Society, 1987.

25. Morishita, Y., Jinmei, T. Request for Comments: 4074, Common Misbehavior Against DNS Queries for IPv6 Addresses. The Internet Society, 2005.

26. Thomson, S., Huitema, C., et. al. Request for Comments: 3596, DNS Extensions to Support IP Version 6. The Internet Society, 2003.