

## IPv6 TRANSITION TECHNIQUES FOR LEGACY APPLICATION

A. Dutta, J. Alberi, A. Cheng, B. Horgan, T. McAuley, D. Chee, B. Lyles  
Telcordia Technologies, Piscataway, New Jersey, 08854

**Abstract**— *As part of Army's IPv6 transition initiative there are several deployment issues that need to be solved. These include transition at several layers of the protocol stack. Army has a suite of legacy applications that currently work on IPv4 systems. Before Army moves to an all IPv6 network, it will need to go through a period of transition where both IPv4 and IPv6 networks, applications will need to co-exist and interoperate. While the IETF has proposed several ways the transition can take place, we have focused our discussion on application transition. As part of this effort we choose the Army's heavily used legacy application MCS-L (Maneuver Control System-Lite) and applied several transition technologies. We took advantage of Telcordia's Dynamic Slicing Tool to study the MCS-L code and made appropriate changes to make MCS-L IPv6 compatible. We tested several transition technologies such as dual stacking, Application Layer Gateway (ALG), Tunnel Broker and NAT-PT with MCS-L application. Both multicast and unicast mode of communication associated with MCS-L were also tested.*

**Index Terms**— IPv6 Transition, Application Layer Gateway, Legacy Application

### I. INTRODUCTION

The Internet has grown to its current size and complexity while using version 4 of the Internet Protocol (IPv4). However, much like population growth and the growth of cellular phones has exhausted telephone area codes and forced area code splits, the growth of the Internet in the developing world and IP-based mobile devices are causing IPv4 to run out of addresses. IPv6 was developed in response to this problem and is designed to accommodate 340 undecillion (340 followed by 36 zeros!) addresses; a tremendous increase compared to the 4.2 billion addresses for IPv4. This massive increase in the number of addresses not only allows all nodes to have their own globally routable addresses, but it also enables much simpler address administration. As a result, IPv6 provides significantly improved support for dynamically configured networks, including Peer-to-Peer (P2P) networks and ad hoc sensor or mesh networks. In addition, it offers

This work was sponsored by C-E LCMC SEC, Ft. Monmouth, New Jersey. Authors would like to acknowledge Bruce Weimer of C-E LCMC SEC, Daniel Chan and Ed Kierman of CERDEC SED, Kwai-Fung Chan and Ranga Reddy of CERDEC S&TCD, Bob Grillo and Richard Mayo of SRI for their input and feedback during this collaborative project.

improvements in many important networking functions, notably in security, mobility, auto-configuration, quality of service, and multicasting.

There are many ongoing initiatives to support IPv6 hosts, routers, hardware, and applications. For example, IPv6 is already being supported in virtually all common office and scientific computers. Router vendors such as Cisco and Juniper also support IPv6 with their network products. Most networking applications and network tools have IPv6 support such as Web servers and video conferencing tools.

Around the world and spanning over five continents, twenty six different countries have national IPv6 initiatives. Many include test-beds for testing the various IPv6 implementations and interoperability. One such test-bed with strong US Government Agencies participation is Moonv6 where participants test product capabilities and interoperability. There are also several other events sponsored by the International IPv6 Forum and the North America IPv6 Task Force (NAv6TF) where IPv6 vendors and customers can obtain and exchange IPv6 information and test their products interoperability. Some of the latest information can be found at [www.ipv6forum.com](http://www.ipv6forum.com) and [www.nav6tf.org](http://www.nav6tf.org).

As part of our project with the Army, we investigated various transition alternatives. However, our primary focus was on the application layer transition from IPv4 to IPv6. We planned an approach for the transition, applied Telcordia's Dynamic Slicing tool to analyze the MCS-L system and then modified the system to be IPv6 capable. In this paper we describe our detailed approach to the problem, analysis of the tools that we used and some of the methodologies that we experimented with.

This paper is organized as follows. Section II discusses some possible transition methodologies involving networking layer and application layer transitions. Section III focuses on the specific issues involving application layer transitioning and highlights the usefulness of the dynamic analysis tools for IPv6 transition. Section IV describes the test-bed and various transition methodologies that were implemented. Finally Section V concludes the paper.

## II. IPV6 TRANSITION ALTERNATIVES

It is expected that for some time there will be a need to maintain connectivity between old IPv4-based devices and new IPv6-based devices. To assist the transition challenges, the IETF devoted considerable effort in identifying IPv4-to-IPv6 transition mechanisms [1], [2]. These mechanisms and associated training guides are provided by many open-source and commercial products, such as, Cisco, Microsoft, and SUN. We categorize the transition methodologies primarily into two, such as network layer and application layer.

### 1.1 Network Layer Transition Alternatives

There are several standard transition technologies available such as: dual-stack Operating Systems, tunneling (automatic, 6 over 4, 6 to 4), tunnel broker, RSIP dual-stack gateway, translator, and NAT-PT. Reference 1 discusses the basic transition technologies. Each of these transition technologies may be applicable and suitable to a specific type of scenario and application. For example, deploying a dual-stack network allows full interoperability with IPv4 networks, but increases network complexity (e.g., managing two (IPv4/IPv6) sets of protocols for unicast routing, multicast routing, configuration, QoS, etc.). On the other hand, deploying an IPv6 only network and NAT-PTs or tunnels is much simpler, but is less flexible and robust.

Simplified descriptions of the three basic transition strategies, shown in the figure 1 are:

- **Dual:** New hosts and routers support both IPv4 and IPv6; use IPv4 among nodes if IPv6 is not supported by both communicating parties. This is analogous to having everyone in the post-office read both English and Greek and letters writers address their letters in either language.
- **Tunneling:** New hosts and routers support only IPv6. When entering a different domain packets are encapsulated (an IPv6 packet inside an IPv4 packet or an IPv4 packet inside an IPv6 packet). Encapsulation is analogous to placing a letter inside of another envelope. If a letter addressed in English has to transverse a Greek postal system, the original envelope is placed in an outer envelope addressed in Greek. Then at the end of the tunnel the outer packet (letter envelope) is opened and the original packet (letter) continues.
- **Translation:** New host and routers support only IPv6. At borders packet headers are converted via a dedicated

translation device. This is analogous to having a dedicated team of translators who spend their days putting a completely new mailing labels on a letter envelopes.

Each of these transition technologies is best suited to specific environments, scenarios and applications. However, an organization also needs to understand that it may find impossible to perfectly satisfy all requirements with a single technology. Each technology involves tradeoffs. For example, deploying a dual network allows full interoperability with IPv4 networks, but increases network complexity. On the other hand, deploying an IPv6 only network and tunnels or translators is much simpler, but is less flexible and robust. Each application will likely have a different weight given to these issues. Similarly, the mechanisms appropriate for a next-generation cellular network may not be appropriate in a military context.

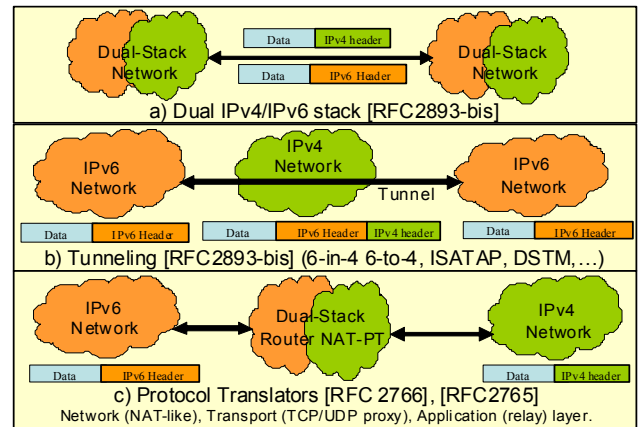


Figure 1. Basic IPv6 Transition Scenarios

Figure 2 shows a deployment scenario for IPv6 transition in Army's network. In a real deployment scenario, there will be a combination of IPv4 nodes and IPv6 nodes. Some of these nodes may have dual stack application, while some other will have only legacy application and thus will need other transition techniques to be able to communicate with another node running IPv6 application. Among other things, these transition scenarios would need to consider, security, mobility, auto-configuration and network management.

### 1.2 Legacy Application Transition Issues

However there are several issues that need to be investigated before any legacy application is ported to IPv6. We provide a brief analysis of these issues and describe how the transitioning tool such as TSVAT can help the transitioning effort. The nature of the TCP/IP protocol stack means that the effect of IPv6 is also felt all

the way up to the applications. Thus, the Army must also consider application transition to provide full IPv6 interoperability in applications running on IPv6 or dual stack hosts. The transition of any set of significant applications from IPv4 to IPv6 is highly likely to introduce new faults and new modes of failure.

2. Transport layer API: Functions to establish communications and to exchange information.
3. Name and address resolution: Conversion functions between hostnames and IP addresses.
4. Specific IP dependencies: More specific IP version dependencies, such as IP address selection, application framing, and storage of IP addresses.
5. Multicast applications: One must find the IPv6 equivalents to the IPv4 multicast addresses and use the right socket configuration options.

We recognize that thorough analysis and testing of significant IPv4/IPv6 application components, to determine potential interface, logic, and dataflow dependencies are essential to the correct transition from IPv4-centric networks to mixed IPv4/IPv6 networks and finally into IPv6-centric networks. Usually, typical errors from the analysis turn out to be coding faults, semantically incorrect IP references, and the incorrect use of old IPv4 references in recoded applications. Hence, areas depicted by the above bullets must be identified and tested to assure the correctness of applications in the new network environments. Detected problems must be promptly resolved. Specifically, problems associated with items 1-5 above must be identified, remediated and tested to assure the correctness of applications in the new network environments.

We have considered Army's one of the legacy application MCS-L for analysis and demonstration.

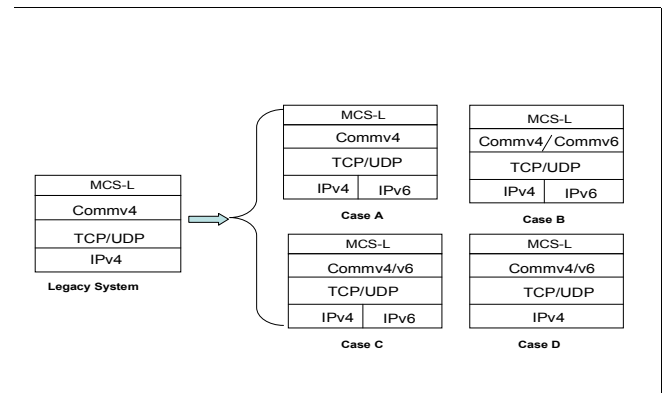


Figure 3. Application Transition for MCS-L

Figure 3 shows some of the possible application transition scenarios that we have considered. We have analyzed the following few cases as plan of attack. MCS-L has a Commserver component that is responsible for communication with other nodes in the network.

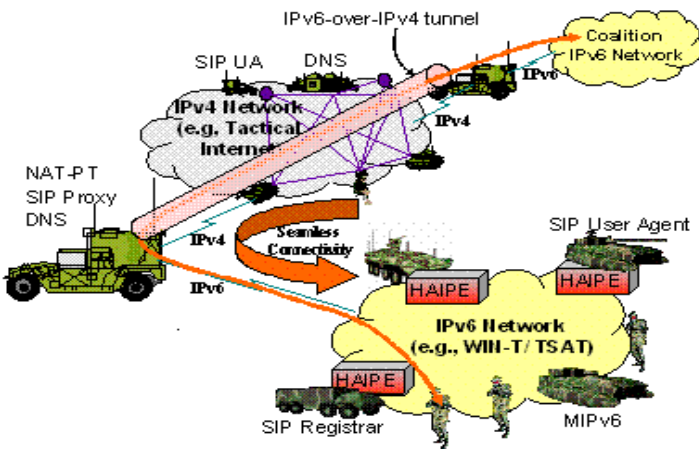


Figure 2. IPv6 transition scenario in Army's deployment

Many IPv4 applications will have to be recoded into IPv6 idioms. This recoding can introduce coding faults (the "fault on fault" rate is often over 25% in such recodings, e.g., Y2K recodings). IPv4 applications which are ported to IPv6 can have "subtle" faults in the use of IPv6 idioms. The easiest way to port an IPv4 application is to substitute the old IPv4 API references with the new IPv6 APIs with one-to-one mapping often using automated tools. If this is done incorrectly some old IPv4 references may remain in seldom exercised sections of code. These hidden old references can be difficult to detect during ordinary testing. The translation of IPv4 references to IPv6 references can also be syntactically correct but semantically incorrect (e.g. interchange of parameters. For example, when an application is supporting both IPv4 and IPv6 through IPv4-mapped IPv6 addresses programmers may mix IPv4 and IPv6 idioms in strange ways. When application source code is written in a protocol-dependent way, the application will support IPv4 and IPv6 explicitly by using two separate sockets. There can easily be errors in whether one can first bind to IPv6 wildcard addresses, and then to those for IPv4.) Semantic and coding errors can arise in any of the dealings with any of the following IP version dependencies:

1. Presentation format for an IP address: An ASCII string that represents the IP address, a dotted-decimal string for IPv4, and a hexadecimal string for IPv6.

**Case A: IPv4 applications in a dual-stack node:** Here Commsserver is an IPv4 application. IPv6 protocol is introduced in a node, but application (i.e., Commsserver) is not yet ported to support IPv6.

**Case B: IPv4-only applications and IPv6-only applications in a dual stack mode:** Applications are ported for IPv6 only. There are two similar applications, one for each protocol version (i.e., Commsserverv4 and Commsserverv6)

**Case C: Applications supporting both IPv4 and IPv6 in a dual stack node:** Applications are ported for both IPv4 and IPv6 support. Thus existing Commsserver IPv4 version can be replaced by the modified Commsserver that has the support for both IPv4 and IPv6.

**Case D: Applications supporting both IPv4 and IPv6 in an IPv4-only mode:** Applications are ported for both IPv4 and IPv6 support, but the same application (i.e., Commsserver) may have to work when kernel does not have IPv6 support

### III. METHODS FOR THE TRANSITION OF IPV4/IPV6 SENSITIVE APPLICATIONS

In the following paragraph we describe the ways we have tried to attack this transitional problem. We break our analysis into three phases, 1) Analyze, 2) Localize, and 3) Remediate

#### 3.1 Analyze

We analyze MCS-L application for network functionality under the following conditions:

1) Buildable/executable source code available a) *with good design documentation and/or training* b) *without design information (Current Commsserver case)*

2) Unexecutable source code available: a) *with good design documentation and/or training*, b) *without design information*

3) Executable only: COTS, GOTS and Legacy fall in this category

#### 3.2 LOCALIZE

Based on different case, we localize the problem accordingly.

Case 1a) or 1b): Use dynamic and static analysis tools to recover networking features and functionality from the code. Good design information always makes this phase

easier. Analysis tool such as Telcordia Software Visualization and Analysis Toolsuite can be used to localize.

Case 2a) or 2b):

- Use static analysis as dynamic application analysis is unavailable
  - Call graphs, path analysis, etc.
  - Design information is more important here than 1a) or 1b)
- Case 3: Use network analyzers to correlate features to network traffic. Good low-level design information is essential here

#### 3.3 Remediate

- Case 1a), 1b), 2a), 2b): Determine optimal placement of IPv4/IPv6 conversion features as a function of the issues outlined on the following slide – possibilities:
  - Inside application
  - Another application
  - At the kernel level (task or state-machine)
  - Gateway along the network
- Case 3): Changing the application is impossible. Must gateway somewhere between application, kernel and network

##### 3.3.1 Issues affecting remediation method

Following are the issues that affect the remediation method

- Separation of concerns: Each component must steward its own set of concerns without untoward mixing leading to excess complexity
- Information exposure and hiding
  - Retained internally in the application
  - Exposed externally to other processes, gateways or unauthorized humans
- Performance
  - More moving parts tend to imply poorer performance
- Integration
  - System installation and configuration
  - Coordinating copies of dynamic data
  - Eventual integration with OPNET simulation environment

An initial investigation of application transition showed that, the transition can either be 1) peer-to-peer mode or 2) client-server mode. Figure 4 shows different possible combination of how the transition can co-exist. As shown an application itself can support both IPv4 and IPv6 while the stack can be either IPv4, IPv6 or both.

Our methods of program analysis, remediation, and testing form the basis of a technology to transform IPv4 applications into IPv4/IPv6 or IPv6 applications.

		Host 1							
Application	Stack	V4	V4	V6	V64	V64	V64	V6	
		V4	V64	V6	V6	V4	V64	V64	
V4	V4	Current	Current	Relay + App upgrade	Relay + App Upgrade	App upgrade	App Upgrade	Relay + App upgrade	
	V64	Current	Current	Relay + App upgrade	Relay + App upgrade	App upgrade	App upgrade	Relay + App upgrade	
V64	V6	Relay + App upgrade	Relay + App upgrade	App Upgrade	App Upgrade	Relay + App upgrade	App Upgrade	App upgrade	
	V64	App upgrade	App upgrade	Relay + App upgrade	Relay + App Upgrade	App upgrade	App upgrade	Relay + App upgrade	
V64	V64	App upgrade	App upgrade	App upgrade	App Upgrade	App Upgrade	App Upgrade	App upgrade	
	V6	Relay + App upgrade	Relay + App upgrade	App upgrade	App upgrade	Relay + App upgrade	App Upgrade	App upgrade	
V6	V64	Relay + App upgrade	Relay + App upgrade	App upgrade	App upgrade	Relay + App Upgrade	App upgrade	App upgrade	
	V6	Relay + App upgrade	Relay + App upgrade	App upgrade	App upgrade	Relay + App Upgrade	App upgrade	App upgrade	

Figure 4. Possible combination of application transition

We have demonstrated the capability of these methods in identifying IP sensitivities in significant applications. Figures 5 and 6 illustrate how software analysis tools can locate a feature in an application by using dynamic slicing tools. The two figures show the result of recovering the *Receive Message* feature in a network messaging application. After executing the application, the tool<sup>1</sup> compares algorithmically test cases that invoke the feature with tests that exclude it. Figure 5 identifies the modules that contain the feature and Figure 6 highlights the code implementing the feature in one part of a file. Extended experimentation and trials are important to refine and perfect the methodology and tools. This involves experimentation with appropriate IPv4 applications that are available in source code and that represent significant network activity consistent with expected network activities.

In addition to the above example, several kinds of analysis can be performed on these test applications. Dynamic and static program slicing [4] and other dataflow analysis of the source of these applications can identify locations in

<sup>1</sup> The Telcordia Software Visualization and Analysis Toolsuite™ is used here but many other tools are available and should contribute to the IPv4 to IPv6 transition.

the code that invoke or are dependant on IPv4 functionality.

The interface signatures of the applications can be analyzed to determine potential weaknesses and risks in interactions with new IPv6 applications.

The calling and call-by structure of the applications can be determined to identify indirect dependencies among the applications that may pose difficulties when operating with new IPv6 applications.

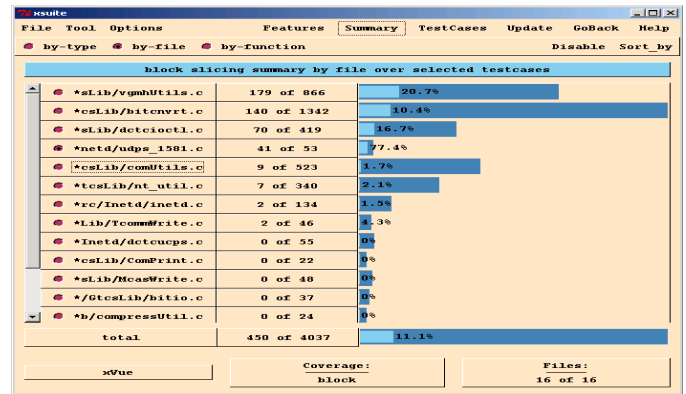


Figure 5. The Receive Message feature recovered by dynamic code slicing

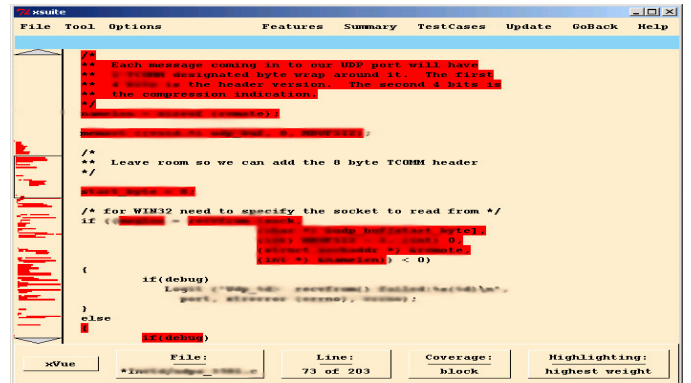


Figure 6. Location in the specific module of code implementing the *Receive Message* feature,

The methods we propose to transform and test IP sensitive applications are essential for organizations facing the evolution of IPv4 applications into IPv6 applications. Following are the three options one can have while making any application transition choices

- A) Buildable and Executable Source Code available
  - Kernel is upgradeable
  - Kernel cannot be upgraded

B) Unexecutable Source Code available

C) Executable Code Only

i) Unknown Protocol

ii) Unknown protocol with weird data

We describe these options in details below

***Option A: Buildable and Executable Source Code Available***

*Approach 1: Convert Application to IP Agnostic*

Desirability: Highest (solution of choice for Commsserver)

Plan of Attack: Use dynamic and static application analysis tools to recover networking features and functionality from the application code.

End State: The application has both IPv4 and IPv6 addresses for any target node and reaches that node by using the correct address from apriori knowledge or trail and error. The following is a concern common to all strategies for IPv4/IPv6 interoperation: Many applications embed IP addresses in the IP packet payload. A common example of this practice is FTP. The SIP protocol used in VoIP is another example. Translation of the embedded IP addresses is an application-level activity that is amenable to remediation by several strategies. However, we believe from the aspect of information hiding and application security that the best place to perform these transformations is with the application and not in an intermediate process between the applications.

Position in the System: IPv4/IPv6 selection is entirely within the application.

Pros: Maximum interoperability in all situations. Best performance because there is no packet rewriting.

Cons: Requires skilled software engineering talent to recover application functionality with no design information available.

*Approach 2 - No IP Address in Payload Content*

Desirability: Low

Plan of Attack: Use dynamic and static application analysis tools together with network analyzers to recover all protocols in use in the application. The application analysis tools insure coverage of all cases on the network.

End State: The solution is a "gateway" that converts packet format and addresses between IPv4 and IPv6.

Position in the System: Variations of "Bump in the Wire" (BIW) or "Bump in the Stack" (BIS) schemes. The gateway may be anywhere from the kernel, to a process on the sending or receiving nodes to a separate translator within the network. Popular implementations are Toredo, ISATAP or 6to4 wrapping on a dual stack system. We should discuss what are the pros and cons of these rather than reinventing again in a separate gateway.

Pros: Simple technology that is readily available to use or modify. It is probably available in open source implementations now or will be shortly.

Cons: Low performance because of packet rewriting. IPv4/IPv6 mappings are visible in the gateway, which raises issues of data synchronization, security and information hiding.

*Approach 3 - IP Address in Payload Content*

Desirability: Lowest

Plan of Attack: Use dynamic and static application analysis tools together with network analyzers to recover all protocols and payload formats in use in the application. The application analysis tools insure coverage of all cases on the network.

End State: The solution is a "gateway" that converts packet format, addresses and payload content between IPv4 and IPv6.

Position in the System: Similar to the previous case above. Possible to implement as a two stage gateway, where the first stage uses BIW or BIS technology for packet conversion and the second stage handles payload conversion.

Pros: None when the source code is available and buildable

Cons: Low performance because of packet rewriting. Application-specific information must be captured and implemented in the gateway. Similar issues of synchronization, security and information hiding as in the above case but with application data now to manage also.

***Option B: Unexecutable Source code available***

*Approach 1: Convert Application to IP Agnostic*

This case is not applicable since source code is not executable

*Approach 2: No Gateway IP Addresses in Payload Content*

Desirability: Conditionally Acceptable

Plan of Attack: Use static application analysis tools together with network analyzers to recover all protocols in use in the application.

Position in the System: Variations of "Bump in the Wire" or "Bump in the Stack" schemes. The gateway may be anywhere from the kernel, to a process on the sending or receiving nodes to a separate translator within the network. Popular implementations are Torero, ISATAP or 6to4 wrapping on a dual stack system.

End State: The solution is a gateway that converts packet formats and addresses between IPv4 and IPv6.

Pros: Hobson's choice as we cannot make the application IP agnostic in this case.

Cons: Low performance because of packet rewriting. IPv4/IPv6 mappings are visible in the gateway, which raises issues of data synchronization, security and information hiding. Depending on the circumstances and the completeness of the source code it might be easier to rewrite the application.

*Approach 3: Gateway IP Addresses in Payload Content*

Desirability: Conditionally acceptable

Plan of Attack: Use application analysis tools and network analyzers to reverse engineer the application

End State: The solution is a "gateway" that converts packet format, addresses and payload content between IPv4 and IPv6.

Pros: Hobson's choice but if some source code is available, one might be able to understand network payload information.

Cons: Low performance because of packet rewriting. Application-specific information must be captured and implemented in the gateway. Similar issues of synchronization, security and information hiding as in the

above case but with application data now to manage also. But depending on the circumstances and the completeness of the source code it might be easier to rewrite the application.

**Option C: Executable only**

Approach 1: *Convert Application to IP Agnostic is not applicable*

Not applicable since there is no source code available

*Approach 2: Gateway - No IP Address in Payload Content*

Desirability: Acceptable

Plan of Attack: Use network analyzers to recover all protocols in use in the application.

End State: The solution is a gateway that converts packet formats and addresses between IPv4 and IPv6.

Position in the System: Variations of "BIW" or "BIS" schemes. The gateway may be anywhere from the kernel, to a process on the sending or receiving nodes to a separate translator within the network. Popular implementations are Torero, ISATAP or 6to4 wrapping on a dual stack system. We should discuss what are the pros and cons of these rather than reinventing again in a separate gateway.

Pros: Hobson's choice as we cannot make the application IP agnostic in this case.

Con: Performance degradation problems because of packet rewriting and IPv4/IPv6 address data in the gateway.

*Approach 3: Gateway - IP Addresses in Payload Content*

Desirability: Not Desirable

Plan of Attack: Use network analyzers to reverse engineer the application including packet payload formats

End State: A gateway that converts packet format, addresses and payload content between IPv4 and IPv6

Pros: Hobson's choice but hope this never happens

Cons: Low performance because of packet rewriting. Application-specific information must be captured and implemented in the gateway. Similar issues of

synchronization, security and information hiding as in the above case but with application data now to manage also.

```

// listen forever
for (;;)
{
    // fill fd set with my socks
    for (i = 0, p = sport[0]; i < MAXSPORTS; i++, p++)
    {
        FD_SET (p->sock, &sel_data.ssock);
    }

    //wait for one to be active
    rc = select (sel_data.nsock, &sel_data.ssock, 0, 0, &sel_data.timeout);
    if (rc == SOCKET_ERROR)
    {
        if (debug)
            Logit ("Inetd: select error!");
    }
    else
    {
        if (sel_data.ssock[0].fd > 0) // some socket
        {
            if (debug)
                Logit ("Inetd: some socket is ready.\n");
            //a socket is ready
            for (i = 0, p = sport[0]; i < MAXSPORTS; i++, p++) //loop thru sockets
            {
                if (FD_ISSET (p->sock, &sel_data.ssock))
                {
                    //this socket is ready
                    rc = recv (&sel_data.ssock, &data, sizeof (data), 0);
                    if (!rc) // destination is STILL_ACTIVE
                }
            }
        }
    }
}

```

Figure 7. Sample code analysis using TSVAT Dynamic Analysis Tool

#### IV. TRANSITION TESTBED IMPLEMENTATION

A prototype test bed is constructed to demonstrate several transition mechanisms discussed in previous sections. Figure 8 below depicts the network topology of this test bed. Our testbed consists of three types of IP networks, namely IPv4-only legacy network, IPv6-only network, and an IPv4/IPv6 dual-stack network. Each type of network includes at least one host running MCS-L application. Hosts are laptops running Windows XP as required by the MCS-L application. The host MCS-4 runs legacy MCS-L application in its original form. It is used to establish baseline conditions and demonstrate backward compatibility of our dual-stack solution as well as IPv4 and IPv6 interoperability.

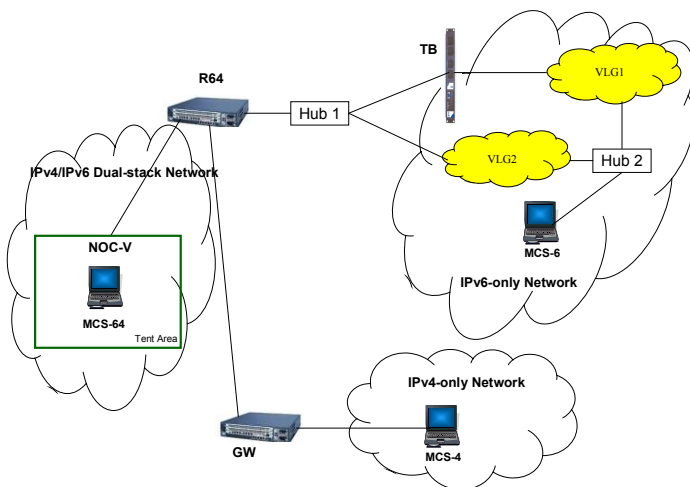


Figure 8. IPv6 Testbed diagram.

MCS-6 and MCS-64 are hosts running the dual-stack MCS-L application developed as part of this project. Specifically, we ported a critical network component of the MCS-L application to support IPv4 and IPv6 dual-stacking. This network component, CommServer, supports various MCS-L functionalities that require network communication and adopt the Joint Variable Message Format (JVMF). JVMF messages are widely used in MCS-L for both textual and graphical object transmissions. While MCS-6 operates in a pure IPv6-only network environment, MCS-64 operates in a network that has both IPv4 and IPv6 support. Further, MCS-64 is situated in the Network Operations Center-Vehicle (NOC-V) environment that can emulate tactical operations. MCS-64 has the additional functionality as an Application Layer Gateway (ALG) to provide transparent transition of JVMF messages between pure IPv4 and pure IPv6 nodes.

VLG1 and VLG2 are IPv6 modeling and simulation (M&S) environments using OPNET version 11.0 and a unique concept called Virtual/Live Gateway (VLG) to represent an Army Future Force backbone network such as the Warfighter Information Network-tactical (Win-T). VLG acts as the physical interface between the live environment and OPNET's M&S environment. This is achieved by modifying the Ethernet driver applet on network interface cards to accept, inspect, and pass IPv6 packets to the opposite end of the VLG environment.

GW (gateway) is a commercial off-the-shelf (COTS) laptop running BSD version 5.4 that acts as a router between IPv4-only network and a IPv4/IPv6 dual-stack network. GW includes two additional pieces of software:

- 1) open source IPv6 multicast routing tool (<http://sourceforge.net/projects/mcast-tools>) that supports Protocol Independent Multicast (PIM) protocol
- 2) open source IPv4-IPv6 multicast gateway (<http://www.uninett.no/testnett/multicast/mcgw/>) that translates between IPv4 and IPv6 multicast, and provide an entire IPv6 PIM-SM domain access to IPv4 multicast.

While IPv6 PIM support exists in some of the most advanced routers, IPv4-IPv6 multicast gateway is not currently available in commercial products. Thus, a BSD machine is required to provide this advanced functionality.

R64 is another laptop running the same BSD version 5.4 as GW and acts as a router between IPv6-only network and a IPv4/IPv6 dual-stack network. A summary table of the five transition technologies and six test scenarios are summarized in Table 1.

Table 1. Summary of Testbed Transition Scenarios and Technologies.

Scenario #	Scenario Description	Transition Technologies
1	Sustain IPv4 legacy baseline interoperability	End host dual stacking
2	Send and receive IPv6 unicast messages	End host dual stacking; IPv6 OPNET simulation
3	Exchange IPv4 and IPv6 unicast messages via ALG	End host dual stacking; ALG; IPv6 OPNET simulation
4	Multi-destination unicast messages in an IPv4/IPv6 hybrid environment	End host dual stacking; ALG; IPv6 OPNET simulation
5	Multicast JVMF messages in an IPv4/IPv6 hybrid environment	End host dual stacking; ALG; Multicast Gateway; IPv6 OPNET simulation
6	IPv6-over-IPv4 automatic tunnel broker	End host dual stacking; Tunnel broker; IPv6 OPNET simulation

The router, R64, runs the same open source IPv6 multicast routing tool as GW. We have also confirmed that this host can be replaced by a Cisco router running an advanced version of IOS with IPv6 multicast routing support. The specific configuration on the router we tested is a Cisco model 3640 with 32MB flash memory, 128MB of RAM, and it is running IOS version 12.4. Hub1 and Hub2 are simple 10/100 BaseT Ethernet hubs with sufficient ports to support the topology.

We created six scenarios to validate it on this test bed to demonstrate five transition technologies discussed above. These six scenarios are:

1. Sustain IPv4 legacy baseline interoperability: This scenario demonstrates that the dual-stack MCS-L hosts retains IPv4 legacy operational functionalities. Tests that unicast JVMF messages are transmitted, received, and processed successfully between MCS-64 and MCS-4 hosts were performed to verify the baseline interoperability.
2. Send and receive IPv6 unicast messages: The purpose of this scenario is to demonstrate that IPv6 unicast JVMF messages can be transmitted, received, and processed between two dual-stack MCS-L hosts, namely MCS-64 and MCS-6.
3. Exchange IPv4 and IPv6 unicast messages via ALG: This scenario demonstrates that the ALG function carried by the MCS-64 host can transparently translate and forward intended unicast JVMF messages from and IPv4 node (MCS-4) to and IPv6 node (MCS-6) and vice-versa.

4. Multi-destination unicast messages in an IPv4/IPv6 hybrid environment: This scenario demonstrates that the dual-stack MCS-L host can distribute a single unicast JVMF messages to a mix of IPv4- and IPv6-only hosts simultaneously.
5. Multicast JVMF messages in an IPv4/IPv6 hybrid environment: This scenario demonstrates that dual-stack MCS-L application supports multicast messages by showing that multicast JVMF messages are successfully transmitted, received, and processed by both legacy IPv4 and dual-stack MCS-L hosts.
6. IPv6-over-IPv4 automatic tunnel broker: The purpose of this scenario is to demonstrate how a tunnel broker can be used to allow IPv6 traffic to traverse a legacy IPv4 network when both communicating endpoints are IPv6-capable.

## V. CONCLUSIONS

The experience obtained through MCS-L transition effort provides a good starting point for integrating and supporting IPv6 in Army's legacy systems. The proof of concept demonstrates the fact that, it is possible to move forward with transitioning the legacy application by way of integration and testing on IPv6 platform. For all practical purposes, the coexistence period for IPv4 and IPv6 is expected to last for many years and will extend the useful lifetime of many legacy applications and systems. Hence, any effort attributed to the IPv6 transition on a system must also ensure that it supports continual sustainment of system's interoperability over legacy IPv4 implementation upon successful IPv6 transition. Thus it is recommended that all maintainers of legacy systems develop an IPv6 transition strategy and transition plan that includes integration testing, regression testing, training, and budgeting to support additional operation, maintenance, and sustainment of dual-stacked hybrid networks.

## References

- [1] E. Nordmark, R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, IETF
- [2] M-K Shin et al, Application Aspects of IPv6 Transition , RFC 4038, IETF
- [3] C. Carver et al, "Integrating Heterogeneous Systems for Real-time Distributed Command and Control", <http://www.scs.org/scsarchive/>, Accessed June 12 2006
- [4] Hira Agrawal, James L. Alberi, Joseph R. Horgan, J. Jenny Li, Saul London, W. Eric Wong, Sudipto Ghosh, Norman Wilde: Mining System Tests to Aid Software Maintenance. IEEE Computer 31(7): 64-73 (1998)