Enhanced IP: IPv4 with 64 Bit Addresses

William Chimiak, Senior Member, IEEE,
Samuel Patton,
and Stephen Janansky

July 23, 2012

Abstract

This paper briefly surveys on IPv4, Carrier Grade NATs, and IPv6. Next, it introduces an extension to IPv4 called Enhanced IP (EnIP) that offers a solution to IPv4 address depletion. EnIP does not replace IPv4 but builds on top of it, attempting to maximize backward compatibility. EnIP is currently deployed between two nodes at the University of Maryland and the University of Delaware, connected via Internet 2. Applications such as http, samba, and ssh have been demonstrated between the two EnIP networks *without modification of the software or routers in the path.*

1 Introduction

The rapid increase of mobile devices has created a large demand for IP addresses. At present, there are approximately 7 billion people and 4 billion usable 32 bit IPv4 addresses. It would be useful to have a unique address for each mobile device so that any device could call another. Presently this is not possible. Since 1999, IPv6 has presented a similar argument that a longer address would make VoIP applications simpler to deploy. Carrier Grade Network Address Translation (CGN) has provided a solution so far.

One powerful feature of Enhanced IP (EnIP) is that it does not require updates to DHCP, ARP, and existing IPv4 routing protocols. By leveraging private address space in a similar manner of IP4+4[1], address space is increased by a factor of 17.9 million[1]. EnIP was designed to minimize impact on Core and Border Equipment and allow changes made and administered by an Internet Service Provider (ISP) or network administrators as close to network users as desired. In addition to this flexibility, it was also developed to allow, not mandate, a change in addressing. Thus, only necessity motivates a numbering change. So ISPs can potentially add 17.9 million new addresses for each valid address they have, when they decide to add addresses. It should also encourage some universities and sites that have highly-unused address block segments in

[1]Private addresses have /8, /12, and /16 address blocks resulting in $2^{24} + 2^{20} + 2^{16}$ or 1.789e+07 addresses

their /8 networks to give some of those small segments to countries with chronic shortages of IP addresses. A /29 block would give them more than 100 million EnIP IPv4 addresses, while losing only 6 of their 16 million addresses for every donated /29 address block.

EnIP packets transit the Internet as IPv4 packets carrying additional address bits and state in an IP option, eliminating routing table updates like IPv6. It does not require a stateful Network Address Translation (NAT) as in CGNs. Finally, because EnIP is stateless, gateways can be clustered or made redundant without complex state exchange. This is problematic for CGNs. EnIP supports end-to-end connectivity, a shortcoming of NAT, making it easier to implement mobile networks.

We do note that over time, our fixed-length IP option would need to be added to the fast path implementations available on core routers[2]. This should not be difficult. Host renumbering is also not required in EnIP as has been the case with other 64-bit protocol proposals[1]. Explaining how this is accomplished is central to the discussion in section 4.

The impending address depletion motivated us to develop EnIP. So we examined four topics. First, we looked into the fundamental transition in the history of ARPANET: the switchover from NCP to TCP/IP. Second, we examined IPv6 efforts. Third, we studied transitions during the life of IPv4. On four occasions, IPv4 evolved to remedy address space exhaustion issues. Finally, we performed a literature survey on Carrier Grade NAT (CGNs) technology. CGNs can offer an immediate fix to ISPs with limited IP address space. Deployment of CGNs appears to have some additional costs in terms of additional expense involved in law enforcement monitoring.

Section 2 of the paper discusses several transitions in the Internet Protocol suite. These transitions include NCP to TCP/IP, the evolution of IPv4, and IPv6. Section 3 describes the design decision to use IP options in EnIP. Section 4 is the heart of the paper, describing the mechanics of EnIP. Section 5 describes the University of Maryland and University of Delaware deployment. It also describes tests between 2 nodes using EnIP and using normal IPv4 and gives the results of the tests. The final section includes closing remarks with emphasis on some areas for further research.

2 A Survey of Layer 3 Protocol Transitions

The following section discusses transitions within three major protocols of the Internet Protocol Suite. The transitions discussed are NCP to TCP/IP, IPv4 Classless Numbering, Network Address Port Translation (usually called NAT), Carrier Grade NAT (CGN), and the current transition involving IPv6.

2.1 Network Control Protocol (NCP) to TCP/IP

NCP was a protocol that preceded TCP/IP as a transport protocol on the ARPANET. In November 1981, Jon Postel wrote RFC 801[3], which described the NCP to TCP/IP transition plan. The goal was to rapidly transition NCP on the ARPANET to TCP/IP. Not all sites were preparing to convert to TCP/IP, so Cerf, Postel, and the TCP/IP team turned off the NCP network channel numbers on the ARPANET IMP's for a full day in mid-1982[4]. The full switchover to TCP/IP occurred on January 1, 1983. A few sites were down for as long as three months while they upgraded their systems. This was possible because the ARPANET was experimental.

2.2 IPv4 transitions

Originally, the 4 byte IP address was divided into the *network number field* (the most significant 8 bits) and the *rest field*(the lower 24 bits). This allowed only 254 different networks on the Internet. In 1981, RFC 791[5] introduced classful network addressing. This prevented network number depletion, and increased addressing flexibility. Around 1993, the Internet was more than an academic curiosity. It was used by many people. Inefficient addressing and a large increase in the routing table size became a problem. To prevent an exhaustion of IP network numbers, without disrupting network services, variable length subnet masks were developed, allowing networks to be divided into even smaller sizes.

The introduction of network address port translation (NAPT), usually called NAT[6], was a surprise for network planners. It vastly increased nodes that could access the network. It was very popular. In addition, three major approaches to implementing CGN were developed: Nat444[7], Dual-Stack Lite[8], and NAT64[9]. In all of these approaches, the CGN is configured with a number of public IP addresses for use in NAT.

In the NAT 444 scenario, the host is NATed at the customer premise and again, at the CGN. With Dual-Stack Lite, the customer's LAN is IPv4, packets are encapsulated in IPv6 packets and at the CGN, the original IPv4 packets are stripped NATed with a public IPv4 address at the CGN. In NAT64, RFC 6144[10], the customer network is IPv6, not dual-stack. DNS lookups resolve IPv4 host names and embed them in AAAA records. A NAT rewrites packets between the IPv6 and IPv4 networks. RFC 6586[11] looks at NAT64 and suggests NAT64 is not viable on its own, recommending the dual-stack approach. IPv4+4[1] has many of the design goals of EnIP. It does not have as much addressing flexibility. IPv4+4 requires a router function, where EnIP does not.

2.3 IPv6

IPv6 had several iterations. Unfortunately, IPv6 was not compatible with IPv4 from the start. Complicating matters, IPv4 became commercially successful. The IPv4 to IPv6 transition could not be disruptive like that from NCP to TCP/IP.

IPv6 deployment issues are addressed in the following places: There are IPv6 deployment problems,[12], peering issues (e.g. thousands of voluntary peering agreements major ISP must make[13], security problems[14], and transition challenges[15]).Finally, IPv6 is somewhat complex. The Broadband Forum published

TR-124 in May, 2010, describing the functional requirements for Broadband residential gateways. It lists twenty two IETF RFCs as IPv6 requirements[16].

3 On the Use of IP Options

IP options seemed an obvious choice for creating increased addressing for EnIP. However, there were issues. Hidell et. al. discuss the introduction of fast path and slow path to routers.[17]. With fast path, the line cards use a copy of the routing table to make local forwarding decisions to other line cards based on table lookups. The fast path does not usually include the ability to process packets containing IP options[2]. As a result, packets containing IP options are forwarded to the slow path CPU for processing and forwarding to the correct line card.

A few packets traversing the slow path should not cause saturation of a router's CPU. However, if EnIP were to become popular, a few test packets might cause high slow path CPU utilization, although EnIP was designed to be ignored by core and edge routers. The IP option used for EnIP has a fixed length of 12 bytes and the first byte is always 0x9a. Presently, some forwarding routers ignore options, others detect if IP options are present and send packets to the slow path. For EnIP packets, the following simple pseudo code would need to be implemented for fast path:

```
if (IgnoreOptions)
   FastPathOperations();
else if (IP_Option_Present AND Option_Byte1 == 0x9a)
    FastPathOperations() ;
else
    SlowPathOperations() ;
```

According to Hidell et. al. routers have already moved towards an architecture where there is flexibility in the fast path, so this should not be difficult.

It is known that ping packets with IP options are likely to traverse a router's slow path. Rossi and Welzl conducted ping measurements to hosts across the Internet and their results suggested a 7% RTT increase in a 2003 study[18]. Fonseca et. al. studied the survivability of packets containing IP options in an Origin autonomous system (AS), Transit AS, or Destination AS[19]. Results showed packets with IP options that were dropped, depended on the option used. Between 85% and 92% of the drops occurred at an edge AS. They showed that support for IP options in the wide area could be restored, discovering the core of the network drops very few packets with options with the majority of drops occurring in edge AS networks. To determine the cases when packets are dropped would require the questionable task of publicly divulging core router configurations.

EnIP and IPv6 both require upgrades to the fast path implementations of routers. A major advantage of the EnIP approach is the simplicity of the upgrade in comparison to the IPv6 core network upgrade. EnIP's fast path upgrade has the three advantages: No equipment reconfiguration is required (e.g. IPv6 address assignment, BGP configuration). Internet providers can **independently** upgrade their fast paths. Finally, there are no requirements to update the IPv4 routing tables.

4 Introduction to EnIP (EnIP)

EnIP design increases IP address space, allowing up to approximately 2ˆ56 possible addresses in contrast to IPv6's offer of approximately 2ˆ128 addresses. The word approximately is used since both protocols include a few address ranges not available for use. EnIP also offers a very practical path towards wide scale adoption. It is this upgrade path and similarity to the existing IPv4 model that many may find valuable.

EnIP uses IP option 26 to create a twelve byte extension to the IP header. This extension contains four bytes of overhead, and two four byte fields used as additional storage for the EnIP source and destination addresses. EnIP addresses are written as two IPv4 addresses concatenated together.

4.1 EnIP Addressing Example

An EnIP host addresses another EnIP host as follows:

65.127.121.2.10.1.1.2

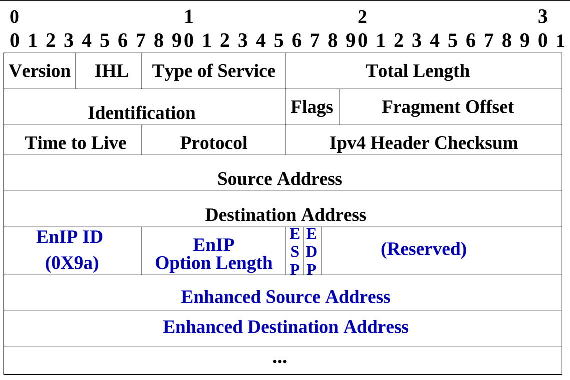
In this example, 65.127.121.2 is a public IPv4 address allocated by one of the Internet registries. Under EnIP, this address is called the *site address*. 10.1.1.2 is called the the *host address* that is a private IPv4 address assigned to a host behind a lightweight EnIP-enabled NAT, not a NAPT. It is the actual private IPv4 address that identifies the host in the network behind that NAT.

On the public IPv4 network, 65.127.121.2 is used for routing purposes. In other words, while the packet traverses the public network, its IPv4 destination is 65.127.121.2. Once the packet reaches 65.127.121.2, which is a NAT upgraded with EnIP software, the IPv4 destination address is changed to 10.1.1.2, which is a value stored in the additional 12 bytes of space afforded by IP option 26. The byte layout used for IP option 26 is defined in Figure 1.

4.2 IP Header with EnIP Option Header

This is the IPv4 header along with additional space used by IP option 26[5].

Figure 1 Header



The EnIP ID field contains the value 0x9a, broken down further using the binary value:

1 00 11010

The first bit, the copy bit, is a 1 to enable the copy. The next two digits are 00 for the control Option Class. The next five digits are 11010, or 26 in base 10. 26 is the IP option value used in the EnIP experiments.

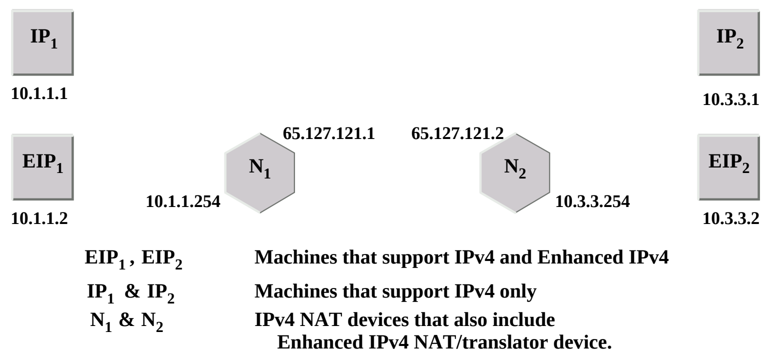
If an EnIP packet traverses a router and must be fragmented because of a link with a smaller MTU, the copy bit ensures that fragments include the 12 byte IP option header in all fragmented packets.

The second byte after 0x9a is the Option Length. This value is always 12. ESP and EDP are one bit fields used to indicate whether EnIP Source Address and EnIP Destination Address are in use. The Reserved field is unused, at present, and always set to zero.

The following scenarios describe the operation of EnIP, but first describe several IPv4 scenarios that exist today as part of a build up to help the network community understand the changes required to implement EnIP.

Reference Figure 2 in the scenarios described below:

Figure 2 IPv4 and EnIP co-existence


IP_1
10.1.1.1
IP_2
10.3.3.1
EIP_1
10.1.1.2
N_1
65.127.121.1
10.1.1.254
65.127.121.2
N_2
10.3.3.254
EIP_2
10.3.3.2
EIP_1, EIP_2 Machines that support IPv4 and Enhanced IPv4
IP_1 & IP_2 Machines that support IPv4 only
N_1 & N_2 IPv4 NAT devices that also include Enhanced IPv4 NAT/translator device.

4.3 NAT Explanation using Figure 2

This section shows that when two nodes are not both EnIP nodes, they use typical port-restricted cone NAT ($NAPT$) procedures to communicate.

When IP1 (10.1.1.1), which is a host with an IPv4 stack, wants to reach 65.127.121.2 it uses NAPT (as on Linux iptables) to masquerade as the public IP address 65.127.121.1. Suppose IP1 (10.1.1.1) wants to reach tcp port 80 on IP2 (10.3.3.1), the packets originating from 10.1.1.1 are NAT'd (NAPT) by N1 to use a source IP address of 65.127.121.1. When these packets arrive at N2 (65.127.121.2), it is necessary to have a NAT (NAPT) port forwarding rule setup on N2 to map tcp port 80 on 65.127.121.2 to forward packets to the internal host IP2 (10.3.3.1). This example nicely illustrates the power of NAT (NAPT) but also highlights its weaknesses to enable end to end host connectivity.

EIP1 is a host with an IPv4 software stack as well as EnIP extensions to IPv4. Suppose EIP1 (10.1.1.2) wants to reach N2 (65.127.121.2). In this case, the destination IP address EIP1 contacts is IPv4 address (65.127.121.2). Because of this, the NAT device (N1) uses IPv4 NAT (NAPT) to translate the source of the packets to come from 65.127.121.1. EIP1 behaves as though it is an IPv4 host as there is no need to use EnIP.

Suppose EIP1 (10.1.1.2) wants to reach IP2 (10.3.3.1) on tcp port 80. To reach IP2, it is necessary to talk to N2 on address 65.127.121.2. This is also a function that can be satisfied by IPv4. So when the packets from EIP1 reach N1, they are translated (NAPT) to appear as though they are coming from N1's external IPv4 address of 65.127.121.1. When the packets from 65.127.121.1 reach 65.127.121.2, 65.127.121.2 must have a port forwarding entry for port 80 setup to send the packets to IP2 (10.3.3.1).

It is important to note that thus far we have not demonstrated any usage of EnIP features.

4.4 EnIP Explanation using Figure 2

1. Suppose EIP1 wants to send packets to EIP2. In this instance both hosts are running Enhanced IPv4 stacks and it is assumed that N1 and N2 support EnIP. Suppose EIP1 knows the address of EIP2 is 65.127.121.2.10.3.3.2 (more on DNS later). EIP1 knows its internal IP address of 10.1.1.2 but is not aware of the external address of N1 (65.127.121.1). Thus, initially the following is done:

   EIP1 sets the source IPv4 address to 10.1.1.2; the EnIP ID field is set to 0x9a. It zeroes the ESP bit in the EnIP header. The EnIP Source Address in the EnIP header is set to all ones, or 255.255.255.255, since an EnIP source address is not currently present. The most significant 32 bits of the EnIP address is set by storing 65.127.121.2 in the IPv4 destination field. The least significant 32 bits of the EnIP address is set by storing 10.3.3.2 in the EnIP Destination Address field. Finally, EIP1 sets the EDP bit to 1.

2. When the packet arrives via IPv4 routing to N1, N1 does the following: It examines the packet and determines it has the EnIP options present (0x9a). N1 writes the EnIP source address by reading 10.1.1.2 from the IPv4 source address field and placing this value in the EnIP Source Address field. This field no longer contains 255.255.255.255. It sets the ESP bit to 1 and places 65.127.121.1 as the IPv4 source address. N1 recomputes the IP checksum of the packet since it has changed. If the packet carries TCP or UDP, it recomputes these checksums as they have also changed.

3. Upon arrival at N2 (65.127.121.2), N2 does the following: It recognizes the EnIP packet (0x9a), reads the EnIP Destination Address of 10.3.3.2 and places this value into the IP header's destination address, so that the IP destination address is now 10.3.3.2. N2 zeroes the EDP bit and the EnIP Destination Address to zero. It recomputes the IP checksum. If the packet carries TCP or UDP, recomputes these checksums as they have changed as a result of a change to the IP destination address. Finally, N2 sends the packet to EIP2.

4. When EIP2 receives the packet it computes the source address of the packet by concatenating the IPv4 source field (65.127.121.1) with the EnIP source field (10.1.1.2) to get 65.127.121.1.10.1.1.2. The IPv4 destination address is 10.3.3.2.

5. To construct a packet from EIP2 to EIP1, EIP2 does the following: It sets the Option ID field to 0x9a. It takes the IPv4 source address field

from the incoming packet, 65.127.121.1, and sets it as the IPv4 destination field. Then it places the EnIP source address (10.1.1.2) in the EnIP Destination Address field. EIP2 sets the EDP field to 1 and the IPv4 source address field to 10.3.3.2. It sets the EnIP source address to all ones (255.255.255.255), setting the ESP bit to 0.

6. When the packet arrives at N2, the following is done by N2: It places 10.3.3.2 in the EnIP source address field. and the ESP bit to 1. N2 places 65.127.121.2 in the IPv4 source address field and recomputes the IP checksum. If the packet carries TCP or UDP, it recomputes these checksums as well. Lastly, it sends the packet to N1 (65.127.121.1).

7. When the packet arrives at N1, it does the following: N1 reads 10.1.1.2 from the EnIP destination address field, placing this value in the IPv4 destination address field. It zeroes the EDP bit and the EnIP destination address field. N1 recomputes the IP checksum and if the protocol is TCP or UDP, recomputes these checksums as well. N1 sends the packet to EIP1.

4.5 Upgrading Servers to Support EnIP

An existing server deployed using an IPv4 address can be upgraded with EnIP kernel modification to support EnIP. Suppose the server is a TCP echo server [20]. The echo program creates a listening socket and then reads data from it. Data read from the socket is written back to the same socket. Without being upgraded, it should be possible for the server to read EnIP packets from the socket but interpreted as a legacy IPv4 packet. However, the server cannot write EnIP packets back to the socket correctly. The packets would only be sent to the source IPv4 address and not back to the EnIP address which includes the source IPv4 address and the EnIP Source Address. Once the server has the EnIP upgrades, it will be possible for it to receive packets and send echo responses back to the full EnIP address that originated the packet. Care must be taken here to ensure that the EnIP Source Address is an allowed private address.

Suppose the echo server logs the source IP address of each data packet received using the *getpeername* function. The length of the address structure returned is currently either the size of a *struct sockaddr_in* (16 bytes) for IPv4 or the size of *struct sockaddr_in6* (28 bytes) for IPv6. The ALPHA implementation of EnIP returns a new structure called *struct sockaddr_ein*(26 bytes). If it is desired to print out the EnIP addresses correctly, it is necessary to use the length 26 to detect a *struct sockaddr_ein*. Inside this struct are two values: *sin_addr1* and *sin_addr2*. These represent the IPv4 source address and the EnIP Source Address. An implementation of *getpeername* could provide a compatibility mode to treat EnIP addresses as IPv4 addresses. Once the echo client software is upgraded, the *getpeername* implementation could return struct *sockaddr_ein*.

4.6 DNS Operation

RFC 2928[21] sets aside the experimental IPv6 prefix 2001:0101. EnIP lookups use AAAA records that begin with the experimental prefix. This prefix uses 32 bits of the 128 bit AAAA record, leaving 96 bits for EnIP to use, of which 64 bits are used. If 65.127.121.2.10.1.1.2 was the EnIP address, the EnIP AAAA record would be 2001:0101:417f:dd02:0a01:0102::0. This use of the AAAA record we call AA record. Thus, **EnIP does not add a new record type which would cause DNS server software upgrades on a massive scale**. Instead, EnIP uses a synthetic record retrofitting the 64 bit EnIP address into the existing AAAA record. It is imagined in the future, that DNS software could be upgraded to hide these details from the user. For example, the user might enter:

```
65.127.121.1.10.1.1.2       AA       eip1.example.com
65.127.121.2.10.3.3.2       AA       eip2.example.com
```

4.7 Optional DNS Upgrades

Suppose EIP1 must speak to another EnIP host behind N1. Call this host EIP3. EIP3 has an address of 10.1.1.3 and like EIP1 has an external source address of 65.127.121.1. An important question to consider is can EIP1 talk to EIP3 using EnIP addressing without relaying all packets via N1? Suppose the enterprise uses the domain name example.com. On the authoritative name server for example.com, the enterprise maintains a list of IP networks controlled by the enterprise. Suppose 65.127.121.1 is the only entry in the list. DNS resolvers often look up AAAA records followed by A records if the AAAA lookup does not succeed. If a DNS query from 65.127.121.1 arrives at the authoritative server requesting a AAAA record for EIP3.example.com it would be possible to send back DNS answer for no such record. When the request for the A record for EIP3.example.com arrives at the name server, the authoritative server responds with the least significant 32 bits of the EnIP address. When other IP addresses query the authoritative name server asking for the AAAA record of EIP3.example.com, they will receive the EnIP address encoded as an IPv6 address.

4.8 Information Security

IDS/Firewalls Operation: Although changes are needed in firewall software for the entire EnIP address, network security devices will easily filter the site addresses as they are IPv4 addresses. This could be utilized to divide host groups behind appropriate EnIP NATs for various purposes. One group is servers, another for video and voice communications, another for research, which then would use existing security devices to provide as much freedom or restrictions as desired by the IT staff. In addition, SSL and SSH runs over EnIP without modification.

Preventing EnIP NAT or hosts from forwarding illegal packets: EnIP-capable NAT devices swap the EnIP destination address into the IPv4 header's destination address. Once the swap occurs the packet is routed on to the address stored in the EnIP destination field. This is the desired behavior when the destination address is for a network directly connected to the NAT. This is not the desired behavior if the NAT is forwarding on to a network that is not directly connected. EnIP NAT devices MUST only perform the swap and forward operation if the EnIP destination address is for an RFC 1918 private address of a network directly connected to the EnIP NAT. To perform this swap otherwise, would mean packets sent to EnIP NATs could be used to relay packets towards unwitting victims. If not protected against, this could lead to these packets being used in denial of service attacks or other malicious acts.

IPSEC: NAT breaks IPSEC in the full tunnel mode AH+ESP scenario. EnIP breaks this configuration also. It is important to acknowledge this as a design limitation. More work would be required to modify IPSEC to work with EnIP.

5 Tests and Comparisons of EnIP and Typical IPv4

This section describes the two-node deployment over the Internet 2 network and the tests made to compare typical IPv4 connections with the enhanced IPv4 connection.

5.1 EnIP Deployment

To test the ability of EnIP to run on a network, a node at the University of Maryland was setup running the Linux 2.6.38 kernel with patches to include the EnIP alpha code. Another similar node was setup at the University of Delaware. Each network has an IPv4 gateway with the hosts behind the gateway. Packets between the two networks are routed over IPv4 but using addresses of the following form: 128.175.150.106.10.1.1.1. The first four bytes is the gateway address and the second four bytes, carried in an IP option, is the address of a host behind the gateway.

The following applications were set up to see if they would run without modification: *http*, *samba*, and *ssh*. Each of the tested nodes had the daemons running to support the applications. A DNS entry was made for the machines on a DNS sever for each side. A browser was brought up on each machine and when the URL was typed in, the web page was displayed. A *ssh* was then made in both directions. Then, *samba* file transfers were made in both directions. Wireshark captures verified the EnIP addresses were used and not the IPv4 address. These applications worked between the two EnIP networks *without modification of the software or routers in the path.* A traceroute revealed there were 7 router hops between the two networks. These routers allowed packets with IP options to pass.

5.2 Test Setup

The test used four *Virtualbox* (version 4.1.2) Gentoo Linux virtual machines (VMs) running Gentoo 2.6.38 kernels. One test was performed on unmodified kernels. The other tests used kernels modified to include EnIP alpha code. These kernels include approximately 700 lines of additional kernel code to process 64-bit EnIP connections.

A Macbook Pro with 8 Gigabytes of RAM and an Intel i7 processor running at 2.7 GHz hosted the VMs. The VMs tested the protocol communications between the four nodes in Figure 2: EIP1, N1, N2, and EIP2.

Two main experiments were done. In the first experiment, IPv4 was used with port forwarding using secure copy (*scp*) and *curl* (used for http) from EIP1 to EIP2. In the second experiment, a modified kernel with EnIP was loaded and *scp* and *curl* used to transfer files between EIP1 and EIP2.

The *Unix time* command measured time to transmit 3 different file sizes using *curl* (to test http) and *scp* from EIP1 to EIP2. The file sizes were 64 Kilobytes, 1 Megabyte, and 50 Megabytes.

5.3 Test Results

The results are shown in Table 1. The twofold speedup when using EnIP is not surprising. EnIP is a stateless NAT that requires minimal processing to rewrite a packet. The original IPv4 NAT (NAPT), however, is stateful. It has processing penalties for the table management to maintain the states for all of the connections in addition to rewriting a packet.

Type of Transfer	**Amount of Transfer**	**EnIP time** (secs)	**IPv4 time** (secs)	**Speed up**
scp copy	**64 KB**	0.159 ± 0.030	0.262 ± 0.040	1.7
scp copy	**1 MB**	0.693 ± 0.084	1.080 ± 0.093	1.6
scp copy	**50 MB**	27.531 ± 1.603	43.844 ± 1.045	1.6
curl copy	**64 KB**	0.114 ± 0.024	0.169 ± 0.035	1.5
curl copy	**1 MB**	0.350 ± 0.108	0.825 ± 0.101	2.4
curl copy	**50 MB**	10.968 ± 2.917	24.786 ± 0.919	2.3

Table 1: Test Results of EnIP and IPv4 Data Transfers

The successful transmissions between the University of Maryland and the University of Delaware, along with this initial test result demonstrates the viability of EnIP in environments requiring increased network addresses.

6 Conclusion

This paper presents a brief survey of layer 3 network protocols. It describes the operation of IPv4 using IP options, called Enhanced IP or EnIP. EnIP provides a solution to the IPv4 address depletion problem that minimizes impact on core, border, and edge routers. The design criteria was that EnIP can extend IPv4 instead of replacing it to reduce impact on routers and information security mechanisms already in place. Operation of two EnIP nodes at the University of Maryland and the University of Delaware, running *http*, *samba*, and *ssh* without modification of the software or routers in the path demonstrates the feasibility of EnIP on a small but realistic scale.

Thousands of *scp* and *curl* transfers were conducted using four virtual machines using payload sizes of 64 Kilobytes, 1 Megabyte, and 50 Megabytes. The tests showed the expected performance improvement in transmission times between the the old $NAPT$ currently used, and the new EnIP NAT. It provided roughly a factor of 2 improvement.

There are additional research opportunities. This includes evaluation of EnIP by other parties, development of EnIP upgrades for a large variety of operating systems, as well as evaluation of the performance and security of existing EnIP implementations. Development of EnIP NAT in ASIC and FPGA form factors can be done to determine the limits of usable EnIP addresses behind an EnIP NAT. Research on the feasibility of using EnIP to do dynamic traffic shaping, quality of service optimization, and real-time network forensics at the EnIP NAT by coordinating with software defined networking (SDN) controllers are some other topics to be explored. The alpha implementation should be improved so it could be included as an option in the main Linux kernel tree. EnIP should be tested over the commercial Internet.

References

[1] Z. Turanyi and A. Valko, "IPv4+4," Proceedings of the 10th IEEE International Conference on Network Protocols, 2002.

[2] J. Aweya, "IP Router Architecture: An Overview," tech. rep., University of Virginia, 1999.

[3] J. Postel, "NCP/TCP transition plan." RFC 801, Nov. 1981.

[4] J. Postel, "TCP/IP Digest, 11 Nov 1981, Volume 1 : Issue 6 (unpublished)." `http://www.livinginternet.com/i/ii_tcpip.htm`.

[5] J. Postel, "Internet Protocol." RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.

[6] T. E. Paul F. Tsuchiya, "Extending the IP internet through address reuse," *ACM SIGCOMM Computer Communication Review*, vol. 23, pp. 16–33, Jan 1993.

[7] J. Yamaguchi, Y. Shirasaki, *et al.*, "Nat444 addressing models," Request for Comments, draft-shirasaki-nat444-isp-shared-addr-08, Internet Engineering Task Force, 2012.

[8] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, “Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion.” RFC 6333 (Proposed Standard), Aug. 2011.

[9] M. Bagnulo, P. Matthews, and I. van Beijnum, “Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers.” RFC 6146 (Proposed Standard), Apr. 2011.

[10] F. Baker, X. Li, C. Bao, and K. Yin, “Framework for IPv4/IPv6 Translation.” RFC 6144 (Informational), Apr. 2011.

[11] J. Arkko and A. Keranen, “Experiences from an IPv6-Only Network.” RFC 6586 (Informational), Apr. 2012.

[12] B. Carpenter, “Advisory Guidelines for 6to4 Deployment.” RFC 6343 (Informational), Aug. 2011.

[13] G. Doring, “IPv6 Peering Thoughts - IXPs DOs and DONTs.” `http://www.space.net/~gert/RIPE/EPF4-IPv6-panel.pdf`, September 2009. 4th European Peering Forum, Kopenhagen.

[14] G. V. Neville-Neil, “Security, safety, and the acceptance of ipv6.” `http://conferences.sigcomm.org/sigcomm/2007/ipv6/1569042755.pdf`, 2007.

[15] P. Savola and C. Patel, “Security Considerations for 6to4.” RFC 3964 (Informational), Dec. 2004.

[16] “Functional Requirements for Broadband Residential Gateway devices,” tech. rep., Editors: Barbara Stark (AT&T) and Ole Tran (Cisco), Broadband Forum, May 2010 Issue: 2. `http://www.broadband-forum.org/technical/download/TR-124_Issue-2.pdf`.

[17] M. Hidell, P. Sjodin, and O. Hagsand, “Router Architectures: Tutorial at Networking 2004.” `http://web.ict.kth.se/~mahidell/pubs/networking04_tutorial_final.pdf`.

[18] M. Rossi and M. Welzl, “On the Impact of IP Option Processing,” October 2003. Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 15.

[19] R. Fonseca *et al.*, “IP options are not an option,” tech. rep., Electrical Engineering and Computer Sciences University of California at Berkeley, 2005. Report No. UCB/EECS-2005-24.

[20] J. Postel, “Echo Protocol.” RFC 862 (Standard), May 1983.

[21] R. Hinden, S. Deering, R. Fink, and T. Hain, “Initial IPv6 Sub-TLA ID Assignments.” RFC 2928 (Informational), Sept. 2000.

7 Biographies

William J. Chimiak is a senior member of the IEEE and the IEEE Region 3 Outstanding Engineer of the year for 2000. He is currently a network research engineer at the LTS in College Park, Maryland. He is researching the IP multimedia subsystem and the 3GPP extended packet core. He holds a PhD in Electrical Engineering from the North Carolina State University. His interests include computer networking, medical informatics, and audiovisual production. Contact him at w.chimiak@ieee.org

Stephen Janansky recently graduated with a degree in Computer Engineering from the University of Delaware, where he conducted research on many topics, such as: analog circuit design, embedded systems, hardware security, and network security. Stephen has at one time or another worked in every layer of the OSI model. His interests include music, Linux, networking, amateur radio, and audio engineering. sjanansky@gmail.com

Samuel T. Patton is a research staff member at LTS in College Park, Maryland. He graduated in 2001 from Illinois State University with a bachelor's degree in Computer Science. His interests include computer networks, programming, and attempting to invent things. sam@ltsnet.net