Enhanced IP

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

Abstract

This paper introduces an extension to IPv4 called Enhanced IP, or EnIP for short. EnIP offers a solution to IPv4 address depletion. EnIP does not replace IPv4 but builds on top of it, maximizing backward compatibility. EnIP is currently deployed between two nodes at the University of Maryland and the University of Delaware, connected via Internet 2. Each network has an IPv4 gateway with 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.

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.

I. Introduction

Enhanced IP (EnIP), provides a model for increasing the life of IPv4 by integrating EnIP into the current IPv4 infrastructure. One powerful feature of EnIP is that it does not require updates to DHCP, ARP, and existing IPv4 routing protocols. EnIP packets transit the Internet as IPv4 packets carrying additional address bits and state in an IP option, so there is no need for a routing table update as is the case in IPv6. However, EnIP's use of an IP option requires minor additional logic to be added to fast path implementations on routers[1].

II. Outline

In parallel to Enhanced IP research, a survey paper covering many aspects of IPv4, IPv6, and Carrier Grade NAT was created. The survey paper is used to justify research on Enhanced IP. The title of the survey paper is: Survey of Deployed Methods for Increasing Internet Address Space. For those not familiar with the transition to IPv6 and the evolution of IPv4, it is recommended that the survey paper be read prior to this paper.

Section III of this paper describes the design decision to use IP options in EnIP. Section IV introduces the mechanics of EnIP, followed by section V which discusses the integration plan for EnIP. The final section includes closing remarks with emphasis on some areas for further research.

III. On the Use of IP Options

The choice to use IP options in EnIP was challenging. During the early stages of experimentation, IP options seemed an obvious choice for creating extensions to IP. However, there were problems. Hidell et. al. discuss the introduction of **fast path and slow path** to routers.[4]. 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[1]. 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 could become billions overnight and slow path CPU saturation could become a significant issue. The IP option used for EnIP has a fixed length of 12 bytes and the first byte is always 0x9a. Presently, forwarding routers detect if IP options are present and send packets to the slow path. For EnIP packets, it would be necessary for the fast path to detect if IP options are present, then determine if the first byte is 0x9a. If so, the packet is an EnIP packet and could then be simply forwarded based on the destination IP address using the line card's routing table. 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 10% RTT increase in a 2002 study, a 7% RTT increase in a 2003 study[5] and a 26% RTT increase in a 2004 study[6]. Fonseca et. al. studied the survivability of packets containing IP options in an Origin AS, Transit AS, or Destination AS[2]. 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 autonomous system. 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.

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 following advantages:

1) No equipment reconfiguration is required (e.g. IPv6 address assignment, BGP configuration)
2) Providers can **independently** upgrade their fast paths.
3) No updates to the IPv4 routing table are required

IV. Introduction to Enhanced IP

The Enhanced IP (EnIP) design increases IP address space. EnIP allows 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.

A. EnIP Addressing Example

An EnIP host addresses another EnIP host as follows:

65.127.221.2.10.1.1.2

1) In this example, 65.127.221.2 is a public IPv4 address allocated by one of the Internet registries. Under EnIP, this address is called the site address.
2) 10.1.1.2 is a private IPv4 address assigned to a host behind a NAT. In this case, the NAT and the host behind it have been upgraded to include software to handle the EnIP extension to IPv4.
3) On the public IPv4 network, 65.127.221.2 is used for routing purposes. In other words, while the packet traverses the public network, its IPv4 destination is 65.127.221.2. Once the packet reaches 65.127.221.2, 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.

B. IP Header with EnIP Option Header

This is the IPv4 header along with additional space used by IP option 26[7].

Fig. 1 Header

0 1 2 3
0 1 2 3 4 5 6 7 8 90 1 2 3 4 5 6 7 8 90 1 2 3 4 5 6 7 8 9 0 1

<table>
<tr><td><b>Version</b></td><td><b>IHL</b></td><td><b>Type of Service</b></td><td colspan="4"><b>Total Length</b></td></tr>
<tr><td colspan="3"><b>Identification</b></td><td colspan="3"><b>Flags</b></td><td><b>Fragment Offset</b></td></tr>
<tr><td colspan="2"><b>Time to Live</b></td><td><b>Protocol</b></td><td colspan="4"><b>Ipv4 Header Checksum</b></td></tr>
<tr><td colspan="7"><b>Source Address</b></td></tr>
<tr><td colspan="7"><b>Destination Address</b></td></tr>
<tr><td colspan="2"><b>EnIP ID
(0X9a)</b></td><td><b>EnIP
Option Length</b></td><td><b>E
S
P</b></td><td><b>E
D
P</b></td><td colspan="2"><b>(Reserved)</b></td></tr>
<tr><td colspan="7"><b>Enhanced Source Address</b></td></tr>
<tr><td colspan="7"><b>Enhanced Destination Address</b></td></tr>
<tr><td colspan="7"><b>...</b></td></tr>
</table>

The EnIP ID field contains the value 0X9a. This field can be broken down further by converting the hexadecimal value 0x9a to binary:

1 00 11010

1) The first bit is a 1. This represents the copy bit.
2) The next two digits are 00. This represents the **control** Option Class.
3) The next five digits are 11010, or 26 in base 10. This represents the new IP option value.

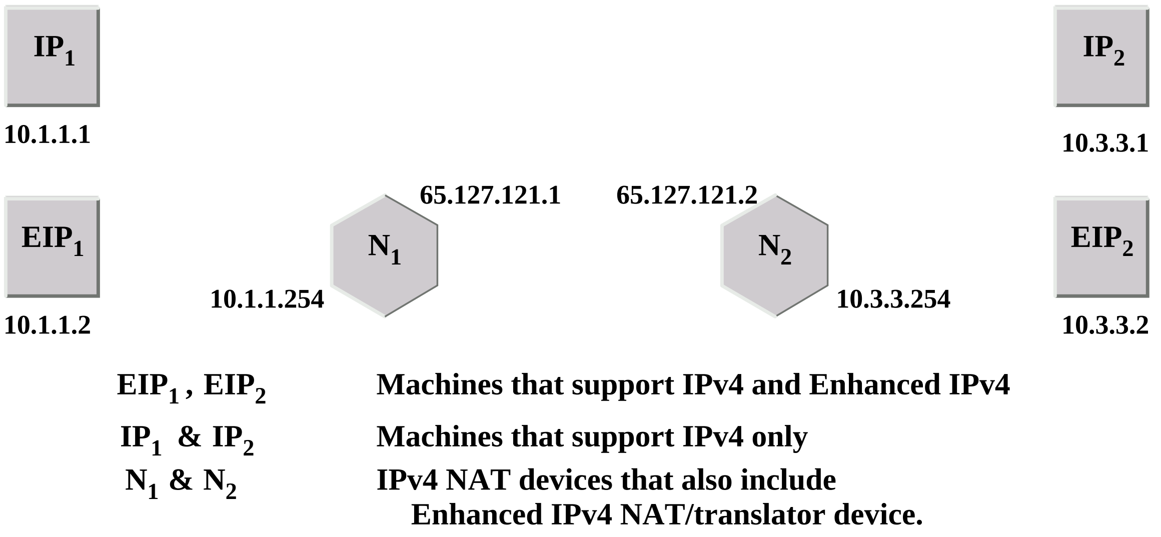
If an EnIP packet traverses a router and must be fragmented because of a link with a smaller MTU, the copy bit ensures that fragments include the 12 byte IP option header in each of the fragmented packets.

In EnIP, 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 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 Internet community understand the changes required to implement EnIP.

Reference Figure 2 in the scenarios described below:

Fig. 2 IPv4 and Enhanced IPv4 co-existence



IP_1
10.1.1.1
IP_2
10.3.3.1
EIP_1
10.1.1.2
65.127.121.1
N_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.

C. NAT Explanation using Figure 2

1) When IP1 (10.1.1.1), which is a host with an IPv4 stack, wants to reach 65.127.221.2 it uses NAT to masquerade as the public IP address 65.127.221.1. (Port-restricted cone NAT as on Linux iptables).
2) 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 by N1 to use a source IP address of 65.127.221.1. When these packets arrive at N2 (65.127.221.2), it is necessary to have a NAT port forwarding rule setup on N2 to map tcp port 80 on 65.127.221.2 to forward packets to the internal host IP2 (10.3.3.1). This example nicely illustrates the power of NAT but also highlights the weaknesses of NAT to enable end to end host connectivity.
3) EIP1 is a host with an IPv4 software stack as well as Enhanced IP extensions to IPv4. EIP stands for ”Enhanced IPv4”. Suppose EIP1 (10.1.1.2) wants to reach N2 (65.127.221.2). In this case, the destination IP address EIP1 will talk to is an IPv4 address (65.127.221.2). Because of this, the NAT device (N1) uses IPv4 NAT to translate the source of the packets to come from 65.127.221.1. In this case, EIP1 behaves as though it is an IPv4 host as there is no need to use EnIP.
4) Suppose EIP1 (10.1.1.2) wants to reach IP2 (10.3.3.1) on tcp port 80. In order to reach IP2, it is necessary to talk to N2 on address 65.127.221.2. Since this is also a function that can be satisfied by IPv4, when the packets from EIP1 reach N1 they are translated to appear as though they are coming from N1's external IPv4 address of 65.127.221.1. When the packets from 65.127.221.1 reach 65.127.221.2, 65.127.221.2 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 Enhanced IPv4 features.

D. 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 65.127.221.2.10.3.3.2 (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 (65.127.221.1). Thus, initially the following is done:
   - The source IPv4 address is set to the address 10.1.1.2.
   - The EnIP ID field is set to 0X9a.
   - The ESP bit in the EnIP header is set to zero.
   - The Enhanced IP Source Address in the EnIP header is set to all ones, or 255.255.255.255, since an Enhanced IP source address is not currently present.
   - The most significant 32 bits of the EnIP address is set by storing 65.127.221.2 in the IPv4 destination field
   - The least significant 32 bits of the EnIP address is set by storing 10.3.3.2 in the Enhanced IP Destination Address field.
   - The EDP bit is set to 1.
2) When the packet arrives via IPv4 routing to N1, N1 does the following:
   - Examines the packet and determines it has the Enhanced IP options present (0x9a).

- Writes the EnIP source address by reading 10.1.1.2 from the IPv4 source address field and placing this value in the Enhanced IP Source Address field. This field no longer contains 255.255.255.255.
- Sets the ESP bit to 1.
- Places 65.127.221.1 as the IPv4 source address.
- Recomputes the IP checksum of the packet since it has changed. If the packet carries TCP or UDP, recomputes these checksums as they have also changed.

3) Upon arrival at N2 (65.127.221.2), N2 does the following:
   - Recognizes the Enhanced IP 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.
   - Sets the EDP bit to 0
   - Sets EnIP Destination Address to zero.
   - 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.
   - N2 sends the packet to EIP2.
4) When EIP2 receives the packet it does the following:
   - Computes the source address of the packet by concatenating the IPv4 source field (65.127.221.1) with the EnIP source field (10.1.1.2) to get 65.127.221.1.10.1.1.2.
   - The IPv4 destination address is 10.3.3.2.
5) To construct a packet from EIP2 to EIP1, EIP2 does the following:
   - Sets the Option ID field to 0x9a.
   - Takes the IPv4 source address field from the incoming packet, 65.127.221.1, and sets it as the IPv4 destination field.
   - Places the EnIP source address (10.1.1.2) in the EnIP Destination Address field.
   - Set the EDP field to 1.
   - Sets the IPv4 source address field to 10.3.3.2.
   - 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:
   - Places 10.3.3.2 in the EnIP source address field.
   - Set the ESP bit to 1.
   - Place 65.127.221.2 in the IPv4 source address field.
   - Recomputes the IP checksum and if the packet carries TCP or UDP, recompute these checksums as well.
   - Send the packet to N1 (65.127.221.1).
7) When the packet arrives at N1, it does the following:
   - Read 10.1.1.2 from the EnIP destination address field, and place this value in the IPv4 destination address field.
   - Set EDP to 0.
   - Place a value of 0 in the EnIP destination address field.
   - Recompute the IP checksum and if the protocol is TCP or UDP, recompute these checksums as well.
   - Send the packet to EIP1..

E. Upgrading Servers to Support EnIP

An existing server deployed using an IPv4 address can be easily upgraded to support EnIP connections. Suppose the server is a simple TCP echo server[8]. The echo program creates a listening socket and then reads data from it. Any data read from the socket is also written back to the same socket. Without being upgraded, it should be possible for the server to read EnIP packets from the socket. However, it will not be possible for the server to 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 one of the allowed RFC 1918 addresses.

Suppose the echo server also 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.

F. DNS Operation

RFC 2928 sets aside[3] the experimental IPv6 prefix 2001:0101. EnIP lookups use AAAA records that begin with the experimental prefix 2001:0101. The prefix 2001:0101 uses 32 bits of the 128 bit AAAA record, leaving 96 bits for EnIP to use, of which 64 bits are used. Supposing the EnIP address was 65.127.221.2.10.1.1.2, the EnIP AAAA record would be 2001:0101:417f:dd02:0a01:0102::0. This use of the AAAA record is called a AA record. It is important to note that a new AA record type was not added, rather the AA record is a record created by retrofitting the 64 bit EnIP address into the existing AAAA record. With this approach, it is not necessary to upgrade DNS server software so long as it supports AAAA lookups. It is imagined in the future, that DNS software will be upgraded to hide these details from the user. For example, the user might enter:

```
65.127.221.1.10.1.1.2       AA      eip1.example.com
65.127.221.2.10.3.3.2       AA      eip2.example.com
```

G. Optional DNS Upgrades

Suppose EIP1 must speak to another EIP 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 65.127.221.1. 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 65.127.221.1 is the only entry in the list. DNS resolvers typically look up AAAA records followed by A records if the AAAA lookup does not succeed. If a DNS query from 65.127.221.1 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.

H. Security Issue: 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 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.

I. Security Issue: IPSEC

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

J. Security Issue: Host security, sensible defaults

EnIP hosts behind a permissive EnIP NAT device are more exposed on the Internet. It is recommended that the default configuration of an EnIP NAT not allow any inbound connections to EnIP hosts behind the NAT unless explicitly configured to do so. This is the most sensible configuration for EnIP devices designed for the home user as replacements/upgrades to existing NAT gateways. This configuration will not be possible for the mobile network.

V. EnIP Integration

Currently, mobile devices are the largest growing consumers of IP addresses. EnIP Integration begins by focusing on the steps to enable Enhanced IP for Mobile Devices. Mobile service providers could benefit from the ability to address any phone with a unique IP address. This is not possible with IPv4. This would be possible with Enhanced IP. It is envisioned that Enhanced IP would enable VoIP systems to make calls based on 64-bit EnIP addresses.

a) Phase 1: Deployment to Mobile Devices and Infrastructure: Technologies such as LTE and WiMAX provide voice channels, previously circuit switched, using Internet protocols (mainly Voice over IP or VoIP). This can stress a provider's overloaded NAT devices and usually requires NAT traversal methods such as STUN to enable end-to-end communications.

NATs are typically limited on the number of devices they can support based on the number of active ports used. There are only 65,535 ports available. So at best one IP can only sustain that many connections for devices it serves. A large amount of loaded persistent connections can degrade NAT service. Voice channels are sensitive to these degradations. EnIP resolves these for providers due to it's stateless nature and ability to allow for end-to-end communication. EnIP on mobile devices is the proposed first stage of integration.

EnIP's deployment to mobile devices should be straightforward for all stakeholders. For simplicity, consider the Android Open Source Platform. Given Android's Linux core, EnIP patches written against the Linux Kernel are merged into a release of Android. Device manufacturers provide carriers with their updated Android device firmware release. Service providers utilize their established firmware update mechanism, updating their customers' devices. Legacy NAT traversal techniques are used with old IPv4 devices for which an EnIP update is not available. After a kernel update the new devices are ready to communicate over EnIP.

The next step in phase 1 is for a service provider to upgrade their networks to support EnIP. To do this, providers first apply the EnIP NAT patch to their NAT devices. This involves a small modification of the NAT kernel moduel (nf_nat.ko) and the addition of another (eipnat.ko). We have demonstrated it is possible to perform this update without rebooting a NAT server. Next, providers update their intermediate connections between provider networks to allow for the passing of IP Option 26. In addition to this, router fast paths will need to be updated so that IP Option 26 is no longer passed via the slow path. Once completed, mobile devices can perform device-to-device communication using EnIP. EnIP connections do not deplete the number of connections available on a NAT as is the case with current VOIP architectures that utilize IPv4 and NAT.

It is possible for EnIP to be a successful protocol completely within the confines of mobile networks. It is envisioned that mobile operators will use EnIP to create end-to-end VoIP systems.

b) Phase 2: Deployment to Content Providers and Infrastructure: A move of mobile devices to EnIP initially benefits mobile providers by making simpler VoIP architectures possible. The remaining traffic transitting their NATs is data traffic to and from content providers. In particular, the large content providers produce most of the traffic. The steps for content providers to support EnIP are simple enough that this integration is realistic.

Content providers upgrade in two steps: First, they integrate EnIP into their networks. IP Option 26 must be allowed to traverse their networks and the fast paths of their routers must be upgraded to quickly process EnIP packets. All routers MUST process EnIP packets in the fast path. Network equipment vendors should make these fast path upgrades available. Vendors not providing an upgrade place themselves at a competitive disadvantage. Since many of these vendors have the experience of designing fast path upgrades for IPv6, the minor changes required for EnIP integration should be simple. Similarly, NAT implementations should begin to integrate EnIP. It will also be necessary to upgrade firewall software in some cases so it can forward packets containing IP Option 26. With this completed, traffic can successfully traverse between content providers and mobile customer networks. Since the customers will already be able to communicate over EnIP, and the packets can now flow between provider and customer, it will be time to upgrade the servers which provide the content.

The second step in Phase 2 is to perform an upgrade to the servers and software which provide the content. EnIP patches for the host operating systems need to be deployed. These patches are minimal and comparable to the application of small security patches. In Linux, for example, it will require short patches of approximately 700 lines of code to the system's kernel in order to allow it to properly process EnIP connections. Once this is completed and verified, applications need to be tested. Our initial application tests worked without modification(ssh, samba, apache, firefox). Once applications are validated, content providers switch to the integrated EnIP-IPv4 setup allowing them to serve both legacy IPv4 customers and the newer EnIP customer set.

c) Phase 3: Deployment to Last Mile and Home Users: First, ISPs integrate EnIP into their core and edge devices, as the content providers did. Then they upgrade the Customer Premise Equipment when additional address space is needed. In cases where the users own their CPEs, the vendors provide firmware updates and users upgrade their devices. For those who have ISP provided CPEs, the ISP can patch the devices' NATs to support EnIP. As before, the patches are small so ISPs can upgrade their customer devices without rebooting the devices. At this point, mobile devices will be capable of utilizing their home network for backhaul to Mobile Providers.

With the user's home network capable of passing EnIP, users could start to make use of the protocol on their computer systems as well. Patches to the host OS could be provided through auto-update procedures to all major Operating Systems. At this point, home users could opt-in to using EnIP on their home network, if they wish. Some mobile service providers who also offer home Internet services may be motivated to upgrade some home users CPE devices early so that mobile device voice calls can be offloaded to a wireline connection.

VI. Conclusion

This paper described the operation of an extension to IP called Enhanced IP or EnIP. EnIP provides a solution to the IPv4 address depletion problem and includes an integration plan, much of which is based on the idea that EnIP can extend IPv4 instead of replacing it. The authors believe the integration plan is feasible and based on sound economic principles.

There is work to be done. This includes evaluation of EnIP by other parties, consideration of other protocols, development of EnIP upgrades for a large variety of operating systems, as well as evaluation of the performance and security of existing EnIP implementations. Much further development and analysis is needed. The interested reader might also consider reading the work done in IPv4+4[9], [10], as many of the design goals in this paper are similar to EnIP.

References

[1] James Aweya. Ip router architecture: An overview. Nortel Networks.

[2] Rodrigo Fonseca et. al. Ip options are not an option. Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2005-24.

[3] B. Fenner. Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers. RFC 4727 (Proposed Standard), November 2006.

[4] Olof Hagsand Markus Hidell, Peter Sjodin. Router architectures: Tutorial at networking 2004. 2004.

[5] Michael Welzl Mattia Rossi. On the impact of ip option processing. October 2003. Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 15.

[6] Michael Welzl Mattia Rossi. On the impact of ip option processing - part 2. July 2004. Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 26.

[7] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[8] J. Postel. Echo Protocol. RFC 862 (Standard), May 1983.

[9] Andras Valko Zoltan Turanyi. Ipv4+4. ICNP ’02 Proceedings of the 10th IEEE International Conference on Network Protocols.

[10] Anrew Campbell Zoltan Turanyi, Andras Valko. Design, implementation, and evaluation of ipv4+4.