IPv4+4

Z. Turányi, A. Valkó
COMET Group, Columbia University
2960 Broadway New York, NY 10027
{zoltan,andras}@comet.columbia.edu

Abstract

The IPv4+4 architecture presented in this paper provides an evolutionary approach to the extension of IPv4 address space. It builds on the existence of Network Address Translators (NATs) and private address realms. During and after the transition from IPv4 to IPv4+4 there is no need to change routers, only NATs and end-hosts. End-to-end address transparency is regained incrementally as deployment progresses. The transition process is simple and it provides incentives for networks with both private and public addresses to upgrade and increase transparent reachability. Early implementation experiences are also included.

1 Introduction

One of the greatest challenges facing the current Internet is the exhaustion of the IPv4 address space. In 1994 the IETF Address Lifetime Expectations (ALE) Working Group projected the exhaustion of the IPv4 address space to around 2008 [6]. Since that time, the Internet has experienced the introduction of Network Address Translators (NAT) [17]. NATs have helped reduce the rate of address depletion, enabling continued operation even in regions with shortages in IP addresses.

NATs, however, also contributed to the loss of IP transparency [21]. The negative impact of NATs on the architecture and operation of the Internet is discussed in the literature [23, 25]. In addition to the architectural impact, NATs also have a number of practical drawbacks. For example, NATs prevent IP level end-to-end security, reduce robustness, break a number of application level protocols, complicate the network and inhibit some novel uses of the Internet (e.g., such as peer-to-peer networking).

Despite these disadvantages NATs are becoming more and more popular. Although NATs adversely impact the global Internet, individual networks in many cases are drawn to use them. First of all, if the acquisition of public IP addresses is hard, complicated or expensive, setting up a NAT is easy, quick and cheap. Second, for most end users a "NAT-ed" connection seems good enough. Third, alternative solutions seems too complicated at present.

In 1994 the IETF selected IPv6 [15] as the next generation of the Internet Protocol. At that time, however, NATs were not yet in widespread use. The ngtrans Working Group of IETF [29] developed several transition mechanisms that would allow the temporary co-existence of IPv4 and IPv6 and the communication in mixed environments. However, despite availability of IPv6 and transition procedures, IPv6 has seen little practical deployment. One factor that prevents the rapid deployment of IPv6 is the complexity associated with transition. If one cannot replace all routers and hosts in a site at once, then a (possibly lengthy) period of co-existence follows. During such a period network administrators must manage both IP versions plus a number of transition mechanisms. Individual networks or users may be hesitant to undergo such change preferring instead to live with the limitations of NATs. While transition to IPv6 would benefit the whole Internet at the expense of extra work at individual networks, using NATs benefits the individual networks directly with negative side-effects on the global Internet. This dilemma between NATs and IPv6 is well described in [24].

Transition to IPv6 may last for a long time in the global Internet. In fact, it is possible that it will never be completed with a significant portion of the Internet remaining IPv4 only. Such a situation may occur if only a certain population (e.g., the cellular industry or regions with shortages of IP addresses) feels sufficient incentive to transit to IPv6 with the rest being content with IPv4. This could result in a lengthy partitioning of the Internet, with intensive use of protocol translation and tunneling mechanisms; a result that may be even worse than the present situation with NATs.

In short, it is possible that IPv4 and NATs will remain and become permanent. If so, then a markedly different approach might be necessary to solve address extension and transparency problems. One such alternative solution is provided by *NAT extended architectures* [27]. NAT-extended architectures rely on the existence of multiple ad-

Proceedings of the 10 th IEEE International Conference on Network Protocols (ICNP'02)
1092-1648/02 $17.00 © 2002 IEEE

dress realms and extend the address space by requiring changes only to hosts and NAT devices.

The IPv4+4 architecture [16] presented in this paper is one NAT-extended architecture that was developed independently of other such architectures. It provides a way back to unique, global, network layer host addresses. End-to-end transparency is restored in the Internet to the extent of IPv4+4 deployment. If deployment becomes complete then IP address transparency will also be fully restored. There is no need to change routers. The transition process provides incentives for networks with both private or public addresses to upgrade and increase transparent reachability. During transition, existing IP and NAT mechanisms are used to communicate with and between IPv4 hosts, thus transition never affects existing reachability. In fact, all transition mechanisms are part of either the current practice or the "final architecture" thus there is no need to set up temporary features. The final picture is a homogeneous and transparent Internet that uses 64-bit addresses and a tunneled packet format.

The paper is organized as follows. Section 2 provides a brief overview and discussion of previous address extension proposals related to IPv4+4. Section 3 and Section 4 describe the architecture and operation of IPv4+4, respectively. The transition process is outlined in Section 5 and further extension of the address space is explored in Section 6. A brief comparision to IPv6 is presented in Section 7. Section 8 reports on early implementation results. Finally, Section 9 presents our concluding remarks.

2 Related work

There has been a considerable amount of work on IPng documented in the literature. Much of this work, however, predates the design and deployment of NATs.

The Simple Internet Protocol Plus (SIPP) [9], that later became IPv6 included a built in address extension mechanism. In SIPP host addresses were originally 64-bits long, but an additional "cluster-addressing" mechanism was added. Cluster addresses are 64-bit unicast addresses referring to a set of nodes behind boundary routers. When complemented with a host 64-bit address they enable the extension of address space quite similar to IPv4+4. The cluster address selects the boundary router and the host address selects the host. Other uses of cluster addressing include mobility and provider selection. Later the IPv6 address space was extended to 128 bits and cluster addressing was omitted and merged into source routing.

Mike O'Dell proposed the separation of identifiers from locators in his 8+8 proposal, later known as GSE [14]. In the 8+8 scheme half of the 128-bit IPv6 address is used as locator (termed "routing goop") and the other half as host identifier. End systems are not aware of their full routing goop, only the part that describes their location inside their site. Site border routers place the missing part of the routing goop into the source address of outbound packets. Although there are some similarities between 4+4 and 8+8, a number of fundamental differences exist. First, in 4+4 no part of the address is used as a location independent host identifier. Second, border routers of 4+4 do not rewrite addresses[1]. Third, IPv4+4 hosts are aware of their full address. Finally, the purpose of the two proposals is entirely different. While 8+8 introduces new address semantics to achieve a number of goals, the purpose of IPv4+4 is address space extension with minimal changes to address semantics. An analysis of 8+8 can be found in [18].

An alternative approach proposed for IPng is Nimrod [12], which also separates host identifiers from routing locators. Routing locators are hierarchically organized based on provider-customer relationships to allow natural prefix aggregation. The PIP proposal [7] is similar in separating identifiers and locators. The locator used by PIP is a list of values that can be thought of as locally significant addresses at a given level of the topology. The list effectively specifies a kind of source route. Hosts may learn parts of it from the configuration, incoming packets and the directory. Later, PIP has been merged into SIPP. The idea of using a list of fields for addressing has already been considered during the design of IP [1]. The primary reason was for address extensions, but the finally proposed 32-bit address seemed large enough. See [8] for a detailed discussion on addressing.

Contrary to IPng proposals, a number of other ideas build on the re-use of the existing 32-bit address space. The separation of private and public address space was first proposed in [4] then in [10]. Address translation and NATs emerged as a way to connect private networks to the Internet.

Realm Specific IP [28] starts from private address realms and NATs. It provides an explicit way for hosts in private address realms to obtain a public address when there is a need to contact a peer in the public address realm. Such hosts would then tunnel their traffic destined to the public peer through the private realm to the NAT. The NAT, in turn, de-capsulates such traffic and forwards it to the public Internet. The drawback of Realm Specific IP is that while it does restore network layer transparency, it does not extend the address space, and as such can only be considered a temporary fix.

Robert Hinden proposed a "medium term" routing and addressing scheme [11] that also uses tunneling as its main tool. Border routers of Autonomous Systems (AS) encapsulate egress traffic and put the source and destination AS numbers into the outer IP header. A block of IP addresses would be set aside to identify ASes in such headers. These

[1] Although realm gateways associated with IPv4+4 rearrange addresses when forwarding packets, no new addressing information is added to the packet.

addresses are injected into the interior routing of transit ASes so only border routers need to recognize them as being special. When the packet reaches its target AS, the ingress border router de-capsulates the traffic and forwards it into the AS. If IP addresses are not globally unique [11] suggests the use of an (AS-address, host address) pair as an extended address adding that "this could be the basis for the long term solution." The idea is very similar to IPv4+4. The use of tunneling as a tool to solve the NAT problem is also mentioned briefly in [21, sec. 5.2.1.3] as something that "has never been fully developed, although is fully compatible with end-to-end addressing." The IETF also investigated the use of IP options for address extension, suggested by Brian Carpenter [5]. The idea was quite similar to IPv4+4, but due to using IP options it required changes to ARP, DNS, SNMP and routers within a site. This idea has been abandoned and never implemented.

The ideas discussed above point toward NAT-extended architectures. The recent IP Next Layer (IPNL) [27] proposal is one such architecture. IPNL uses existing IP address realms as "links" to an overlay (or next layer) protocol. The new protocol is tunneled in IP packets and is below the transport layer. Enhanced NAT boxes, called *nl-routers*, route packets realm by realm toward the destination.

IPNL introduces *realm numbers* to identify different realms the same *frontdoor* — an nl-router that connects private realms to the public Internet. The public IPv4 address of the frontdoor concatenated with the realm number identifies the private realm. The (frontdoor address, realm number, host address) triplet, called IPNL address, fully identifies a host in a private realm. However, IPNL addresses are not used as long term host identifiers, only as locators that can change frequently. One reason for change might be a switch to a new frontdoor (e.g., when the old one fails). IPNL uses fully qualified domain names (FQDNs) as host identifiers. Nl-routers are able to route packets based on both FQDNs and IPNL addresses. During initial packet exchanges peers use FQDNs in packet headers to address each other. At the end of the exchange they learn both their own and each-other's IPNL address and use that address for subsequent communication. If a connection toward two peers exits through different frontdoors then a host may learn two different addresses for itself. Hosts are not aware of all their possible addresses. This is not necessary, as the FQDN can be used in packets to identify the host. If the IPNL address changes during a session, the peers automatically detect it and switch to using the new address.

To facilitate such routing a new routing protocol is installed among nl-routers behind the same frontdoor that distributes DNS and realm number information among nl-routers. Nl-routers are also aware of the FQDNs and host addresses of all hosts in the realms they are attached to. This enables them to resolve FQDNs to IPv4 addresses in incoming FQDN-addressed packets. Nl-routers are also capable of performing realm number translation to allow two realms behind the same frontdoor to use the same realm number.

While we believe that IPNL is much easier to deploy than IPv6, IPNL is a fairly complex architecture that represents a significant departure from the current routing and addressing paradigm. Hosts are no longer aware of their full network layer address, only their name. Host identifiers are DNS names once and for all. Moreover, the routing infrastructure becomes irrevocably intermingled with the DNS. Besides architectural impact this may raise some performance issues too. Nl-routers need to manage DNS related routing information and a per-host database. Per-packet processing is complicated for some packets. The packet header is also quite large, 44 bytes for intra-realm packets and 60 bytes for global packets, plus the size of FQDNs if used. Finally, a distinction between the current public address realm and the private address realms are maintained in the architecture. It is not clear if, or how, this distinction can be removed and the Internet made homogeneous again.

Finally, TRIAD [26] is also a new Internet architecture that builds on the existence of IPv4 address realms and uses names as identifiers. Its primary focus is content distribution. TRIAD introduces a *content layer* and *Content Routers* (CR). CRs hold DNS information and forward combined name lookups/connection setup requests toward destinations. The result of this lookup is transport connection information and a list of relay identifiers that specify a path through several consecutive address realms. A shim header is added to IP packets that contains the list and enable *relays* at the border of address realms to forward packets. IP addresses become locally significant with names representing globally unique identifiers. In comparision, IPv4+4 proposes simple changes to the network layer focusing entirely on address extension, while retaining existing semantics as far as possible.

3 The 4+4 architecture

The basic idea of IPv4+4 is as follows. The primary goal of address extension is to provide nodes currently in private address realms with an end-to-end address. The extended address is formed by concatenating two 32-bit IPv4 addresses: a public and a private one. The public address selects the address realm, while the private address selects the node inside the realm. In fact, the public part is the address of the NAT connecting the realm to the public Internet. Nodes in the public Internet use their existing address as the public part and 0.0.0.0 as the private part.

IPv4+4 packets are minimally encapsulated IP packets [13]. They contain two 32-bit fields for each of the source and destination address. The two parts of the extended address are placed into the two 32-bit address fields. The fields

are managed such that the outer header always contains addresses that are understood by the IPv4 routers in the realm the packet is traveling in. That is, in a private realm the private half of the extended address is visible in the outer header, while in the public Internet the public half is there. This ensures that routers can forward the packet toward the destination without understanding 4+4.

The following sections describe addressing and the header format, while routing is discussed in Section 4.

3.1 Network model

We define an address realm as a collection of networks using addresses from the same address block, while using one address only once. That is, an IP address unambigously identifies an interface within an address realm. We further differentiate the one and only public address realm that uses the public IPv4 address space and several private address realms, each of which may re-use the private address space designated by [10]. Both the public and private realms contain usual TCP/IP networks with routers and hosts. Today, private realms connect to the public realm via NATs.

Figure 1 shows the scenario assumed by IPv4+4. Grey networks belong to the public address realm and each white network represents a separate private realm. For example, networks 1 and 3 represent realm ***A*** and realm ***B***, respectively, and both can re-use the entire private address space.

Upgraded NATs, called *realm gateways*, are integral part of the 4+4 architecture. In addition to the address translation function[2], realm gateways perform a few simple operations on IPv4+4 packets. Realm gateways also act as legacy routers that forward IPv4 packets with public destination addresses. Realms may be interconnected using arbitrary topology and an arbitrary number of realm gateways. The only requirement is that each interface of realm gateways must either belong to a private realm or the public realm to separate the address spaces. Note that the public address realm need not be continuous.

In Figure 1 private realm ***A*** and ***B*** are connected to the public network 2 via realm gateways with public addresses **A** and **B**, respectively[3]. The figure also shows four hosts, two in the public realm (nodes **C** and **D**) and two in private address realms (nodes **X** and **Y**).

IPv4 addresses in the public realm are termed as *level 1* addresses (addresses **A**, **B**, **C** and **D** in Figure 1), while addresses in private realms are termed as *level 2* addresses (**X** and **Y**). In the remainder of this paper bold capitals are used to denote 32-bit IPv4 addresses and the nodes having those addresses. The term *router* always denotes a legacy IPv4 router that is unaware of IPv4+4.

[2]The use of the address translation function will diminish as transition progresses and it can be fully removed if deployment is complete.

[3]These addresses are separate from the address pool assigned for the address translation function.

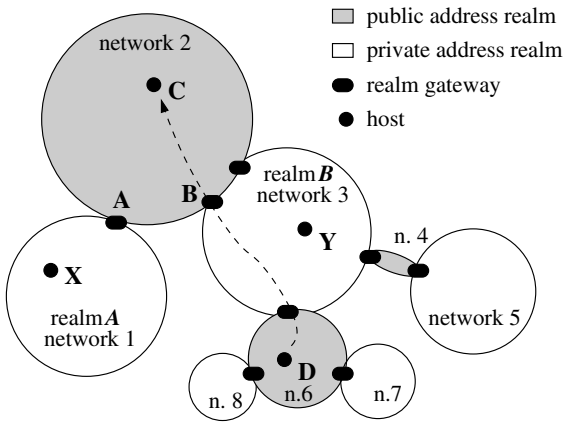

Figure 1. Example networks with arbitrarily interconnected realms.

The routers inside private address realms are configured with the routing information about both private and public addresses; that is, they know how to route toward both level 1 and level 2 destinations. For example, assume that node **D** in network 6 posts a packet to node **C** in network 2. The packet uses public source and destination addresses and will be delivered unaltered to the destination through the routers and realm gateways of network 3.

The routers in the public address realm, on the other hand, are configured to route toward public addresses only. This means that realm gateways must filter out routing information of private addresses when talking to routers in the public address realms. Note that realm boundaries do not have to coincide with autonomous system boundaries.

3.2 Addressing

The IPv4+4 address of a node inside a private realm is a concatenation of two IPv4 addresses: the (public) address of a realm gateway and one of the node's own (private) addresses. We denote this as **A.X**, where **A** represents the realm gateway and **X** represents the node's own address. The first and second parts of the address are termed *level 1* and *level 2* parts, respectively. IPv4+4 nodes may have multiple addresses if the host has multiple interfaces or the realm is 'multihomed' (i.e., there are multiple realm gateways). Any (level 1 part, level 2 part) combination constitutes a valid address of a node. Multiple addresses of the same node are treated the same way as in IPv4, that is, nodes accept packets on all of their addresses and may use any of their addresses as source address in outgoing packets. Transport layer semantics are unchanged, sockets are bound to a tuple of two IPv4+4 addresses and two port numbers.

The DNS is used to store and retrieve IPv4+4 addresses conceptually the same way as with IPv4. There are two alternative implementations. First, a new record type can be defined. Second, two type A records can be used to store the level 1 and level 2 parts of the 4+4 address. The two records are accessible through a prepended domain name, similar to SRV records [20]. For example, the level 1 and 2 parts of the IPv4+4 address of the machine `foo.bar.edu` are stored under `_l1.foo.bar.edu` and `_l2.foo.bar.edu`. The benefit of the latter alternative is that it requires no modification to DNS servers. Our implementation (see later) uses this alternative.

A host may have multiple of both address parts. This is similar to a host having multiple IPv4 addresses today. The level 2 address of a host inside a private realm is also advertised as a legacy IPv4 address to allow IPv4 communication inside the realm. Likewise, the level 1 address of a host in the public Internet is also available as an IPv4 address. Reverse DNS can be provided by prepending the reverse of the level 2 address part to the usual d.c.b.a.in-addr.arpa DNS name.

3.3 Header syntax

IPv4+4 packets are IPv4 encapsulated IPv4 packets. The header syntax is similar to minimal IP-in-IP encapsulation [13], that is, the inner header contains only addresses and a checksum. Encapsulation allows the use of a larger address space and is also a key element for backwards compatibility. The IPv4+4 header is shown in Table 1.

Ver.	Hlen	DS byte	Total Length	
Identification			Flag	Fragment offset
TTL		Protocol 1	Header Checksum 1	
Source Address 1				
Destination Address 1				
Source Address 2				
Destination Address 2				
Protocol 2	SPos	DPos	Header Checksum 2	

Table 1. The IPv4+4 header

The first three rows of fields in the IPv4+4 header are interpreted in the same manner as the IPv4 header, with the exception that the `Protocol 1` field is set to a value that specifies IPv4+4 encapsulation. The `Source Address 1` and `2` fields collectively contain the full IPv4+4 address of the source node, while the `Destination Address 1` and `2` fields contain the full destination address.

The `SPos` and `DPos` fields indicate how the source and destination IPv4+4 addresses are partitioned to the two 32-bit fields. A value of 0 means that the address is *unswapped*, that is, the level 1 address part is in the outer header and the level 2 is in the inner header. A value of 1 means that the two address parts are exchanged and the address is *swapped*. Later in the paper more `SPos` and `DPos` values will be introduced.

The `Protocol 2` field indicates the protocol above IP (e.g., TCP), while the `Header Checksum 2` covers the end-to-end information in the IPv4+4 header[4].

Similar to IPv6, only end hosts may fragment IP packets. This is accomplished by setting the `Don't Fragment` bit in the IPv4 header and using path MTU discovery [3]. The fragmentation related fields of the IPv4 header are used exactly as in IPv4. All fragments contain the inner header as well. Reassembly is performed at the final destination only.

We note that a system using extension headers (similar to IPv6) can be defined for IPv4+4. This would allow the reuse of several mechanisms defined for IPv6. Fragmentation can also be made part of the extension header mechanism leaving the second row of the header mostly unused.

4 Routing

One of the benefits of IPv4+4 is that IPv4 only nodes can essentially communicate as today. They use IPv4 packets, the existing IPv4 routers and if they are in different address realms then they use NATs. Therefore, no new transition mechanisms are needed to provide service to old hosts. Four different scenarios are possible with regards to the relative location of two IPv4 only nodes.

If the two nodes are located in the same private realm, the private IPv4 addresses and the IPv4 protocol are used. If both nodes reside in the public address realm, then they can use their public addresses and the IPv4 protocol to communicate, even if they are separated by one or more private realms. This is possible because the routers inside the private realms have public address routing information and are capable of forwarding packets with public addresses. If one of the IPv4 nodes is in a private realm and the other is in the public Internet, then address translation is performed as today using a NAT. In this case the known problems of NATs apply. These limitations can be resolved by upgrading the nodes to IPv4+4. Finally, if both nodes are in different private address realms then it is impossible for them to communicate unless they upgrade to IPv4+4. In what follows, we describe, how two IPv4+4 nodes in different address realms communicate with each other.

4.1 Routing between two private realms

Assume that a node **X** wishes to send a packet to node **Y** as illustrated in Figure 2 (solid arrows). Assume further,

[4] This includes the addresses, the `Protocol 2` field and the payload length, which is the total length minus the IP header length.

that both nodes are IPv4+4 aware, that is, their operating system is prepared to send and receive IPv4+4 packets. First, node **X** checks if any of the level 1 address parts returned by DNS for node **Y** match any of its own level 1 address parts. If that is the case then the two nodes are in the same address realm and use IPv4 packets to communicate. If this is not the case then **X** selects one level 1 and level 2 part from the set of address parts of node **Y** to form the destination address (e.g., **B.Y**). A source address is also selected (e.g., **A.X**).

Next, the source node creates a 4+4 header and fills in the source and destination address fields as follows. The level 1 part of the source address is placed in `Source Address 2` and the level 2 part in `Source Address 1`, The level 1 and level 2 parts of the destination address are placed in `Destination Address 1` and `2`, respectively. In other words the source address in this packet is swapped, while the destination address is unswapped. This packet header is denoted symbolically as

$$\begin{matrix}\mathbf{X} & \\ \mathbf{A} & \end{matrix} \rightarrow \begin{matrix}\mathbf{B} \\ \mathbf{Y}\end{matrix}$$

where the upper row represents the addresses in the outer IPv4 header (source address is **X**, destination address is **B**) and the lower row represents the addresses in the inner header (source address is **A**, destination address is **Y**). The full 4+4 source address **A.X** is in the left column and is swapped. The full 4+4 destination address **B.Y** is in the right column and is unswapped. The IPv4 routers in realm ***A*** only see $\mathbf{X} \rightarrow \mathbf{B}$.

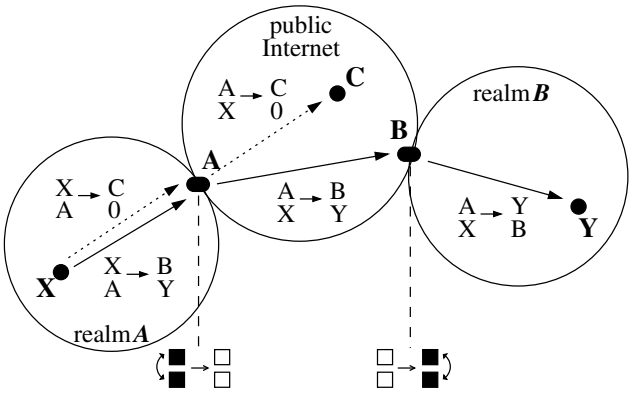

Figure 2. Routing between two IPv4+4 nodes in different realms.

As a result, the routers will forward the packet toward node **B**. (See Figure 2.) When it reaches the border of realm ***A***, the realm gateway exchanges the content of the fields `Source Address 1` and `2` making the source address unswapped. (This is depicted visually at the bottom of Figure 2.) Then the packet is forwarded into the level 1 (public) Internet. At this point the addresses in the packet are as shown in Figure 2. The IPv4 routers see only $\mathbf{A} \rightarrow \mathbf{B}$ and continue forwarding the packet toward node **B**. We note that in the case of a 'multihoming' realm, it may be a realm gateway other than **A** that executes the change.

When node **B** receives the packet and sees that it is an IPv4+4 packet, then it swaps `Destination Address 1` and `2`. The outer header contains $\mathbf{A} \rightarrow \mathbf{Y}$ allowing the routers inside realm ***B*** to forward the packet to node **Y**.

When node **Y** receives the packet, it recognizes itself in the full destination address. If it is required to send a response, it finds the complete IPv4+4 address of the sender in the packet. The return packet is routed across realm boundaries in the same way as the forwarded packet. Realm gateways **B** and **A** will swap source and destination addresses, respectively. The addressing fields of the packet will be

$$\begin{matrix}\mathbf{Y} & \\ \mathbf{B} & \end{matrix} \rightarrow \begin{matrix}\mathbf{A} \\ \mathbf{X}\end{matrix}, \quad \begin{matrix}\mathbf{B} & \\ \mathbf{Y} & \end{matrix} \rightarrow \begin{matrix}\mathbf{A} \\ \mathbf{X}\end{matrix}, \quad \begin{matrix}\mathbf{B} & \\ \mathbf{Y} & \end{matrix} \rightarrow \begin{matrix}\mathbf{X} \\ \mathbf{A}\end{matrix}$$

as the packet travels through realm ***B***, the level 1 realm and realm ***A***, respectively.

The above swapping procedure ensures that private IPv4 addresses are never used in the outer header outside the private realm they belong to. Therefore, IPv4 routers of both private and public realms see only such addresses on which they have routing information.

When realm gateways swap source or destination addresses, they also set the `SPos` or `DPos` fields appropriately. In addition, they decrement the `TTL` field and recalculate `Header Checksum 1`.

4.2 Routing between public and private realms

In what follows, we describe routing between a node in the public Internet and a node in a private address realm (e.g., node **C** and node **X** as illustrated in Figure 2).

If both nodes are IPv4+4 capable, then they can use the full IPv4+4 header to communicate. The IPv4+4 address of node **C** is **C.0**. (Here **0** refers to the all-zero IP address 0.0.0.0). The realm gateway performs exactly the same address swapping as described in the previous section. The addresses in a packet sent from **X** to **C** are shown in Figure 2 beside the dotted arrows. On the way back, the fields are the same with the source and destination exchanged.

If any of the nodes is not IPv4+4 capable, then IPv4 and traditional address translation is used. Observe that here an already existing mechanism (i.e., address translation) is used as a transition tool to enable communication between upgraded and non-upgraded hosts in different realms.

4.3 ICMP messages

ICMP messages provide important error feedback, control and debugging functions that are an integral part of the Internet Protocol suite. ICMP messages are used in IPv4+4 conceptually the same way as in IPv4. The current definition of the ICMP protocol is used unchanged (although it is possible to restructure the ICMP protocol as was done in the case of IPv6).

ICMP messages generated by IPv4+4 end-hosts, such as Echo or Port Unreachable are addressed and routed just like any other IPv4+4 packet. End-hosts include the full IPv4+4 header plus 8 bytes of the original packet. This allows for the inclusion of protocol and port numbers.

Some ICMP messages generated by routers in response to packets not addressed to them, however, require special attention from realm gateways and IPv4+4 hosts[5]. Since routers along the way may be IPv4 only routers, the ICMP messages may not be sent to the original source, but to the outer IPv4 source address of the packet. The following sections discuss this possibility more in detail.

Assume that node **A.X** sent a packet (packet p) to node **B.Y** but the packet can not be delivered and router **R** generates an ICMP message in response. If **R** is in realm ***A*** then the outer source address field of p contains the level 2 address of the source node, that is **X**. In this case the ICMP message will reach the source node without special treatment. The source node is able to recognize that the ICMP message was sent in response to an IPv4+4 packet by looking at the ICMP payload.

If the router **R** is in the public address realm, then the ICMP message will be sent to the realm gateway **A**. This realm gateway determines that the packet included in the ICMP message is an IPv4+4 packet and converts the IP header of the ICMP message to IPv4+4. The destination address will be **A.X** (swapped) copied from packet p included in the ICMP message. The source address will be **R.0**. This allows the original sender to identify the router that generated the ICMP message.

If the router is in realm ***B*** (or any private address realm different from ***A***) then the ICMP message will be routed toward the realm gateway **A**. However, because the ICMP packet is an IPv4 packet containing a private source address, it needs address translation and will be captured by the realm gateway at the realm border (**B** in our case). Recognizing the ICMP message as a response to an IPv4+4 packet, the realm gateway converts the ICMP message header into IPv4+4, with destination address **A.X** (unswapped) and source address **B.R** (unswapped) and forwards it into the public address realm. This packet is then routed to node **A.X** as a regular IPv4+4 packet.

Another problem with router generated ICMP messages is that only the IPv4 header of the original packet (including the IP options) plus 8 bytes of payload are included [2]. In case of an IPv4+4 original packet, this excludes the transport protocol and port information. As a consequence, the source host cannot identify the transport connection for which it received an ICMP message. However, this is not as serious as it first seems, as most ICMP messages generated by routers correspond to all transport sessions with the destination host. If the route is congested or the peer is unreachable, then it affects all such transport sessions. In this case the signal may be delivered to all relevant transport identities. Another approach to help this situation may be to cache the `Identification` fields of recently emitted packets together with the protocol and port information. As the `Identification` field is returned in the ICMP message, the source node will be able to identify the port and protocol information. The best solution is to configure IPv4 routers to return an additional 12 bytes of payload in ICMP messages, if possible.

[5]These ICMP messages include Host and Network Unreachable, Time Exceeded, Parameter Problem and Source Quench messages. Other messages, such as Redirect or Router Discovery messages are always sent inside subnets and thus are not affected by IPv4+4.

5 Transition to IPv4+4

Transition to IPv4+4 represents a straightforward, stepwise upgrade of NATs, hosts and optionally routers.

To upgrade a private access realm at least one of its NATs must be upgraded first to act as a realm gateway. The new functions are (1) the ability to swap addresses in IPv4+4 headers; (2) the conversion of ICMPv4 message headers; and (3) the participation in routing and filtering of private addresses. The latter function is already part of many NATs today. If a realm is concerned about the realm gateway as a single point of failure, it shall set up two realm gateways with the same address. Since realm gateways hold no per-flow state, if one fails, the other can take over. If the realm wish to do robust multihoming to multiple ISPs, the same realm gateway address have to be advertised to each ISP. Although such a construct is known to increase core routing table sizes, lacking alternatives it is commonly used today.

After the private realm has one realm gateway, hosts inside the realm can start upgrading. To upgrade a host, its operating system must be augmented with the ability to send and receive IPv4+4 packets. Auxiliary protocols, such as DHCP, ARP, RARP, router discovery, etc., need not be modified. Similar to IPv6, some applications also need to be upgraded at least to handle larger addresses. Bump-in-the stack address translation [19] developed for IPv6 might be used allowing applications that do not carry IP addresses in payloads to run unchanged.

The DNS itself need not be modified if 4+4 addresses are stored as two type A records (see Section 3.2). The address

of the upgraded hosts, however, needs to be included in the DNS zone files and made available to the outside world.

Optionally, routers can also be upgraded at least in certain regions to understand the 64-bit 4+4 addresses. This requires modifications to routing protocols and the forwarding engine.

5.1 Transition incentives

Transition will probably be started by networks that have insufficient amount of IPv4 addresses. These may be existing networks using NATs or new networks that find the acquisition of many IP addresses too costly. This is part of the incentives, as transition is directly motivated by the problem the new architecture aims to solve, i.e., address depletion.

Upgraded hosts immediately gain access to all other IPv4+4 nodes globally regardless of location. Upgraded hosts use IPv4 inside a realm and IPv4+NATs outside a realm to communicate with non-upgraded hosts, just as they did before the upgrade. This means that hosts can be upgraded one-at-a-time without impacting other hosts.

As the number of upgraded hosts increases in private realms, hosts in the public address realm also have growing incentive to upgrade to IPv4+4. They can do so at any time individually. As a result they gain access to hosts behind NATs, e.g., to run peer-to-peer applications. At this point IP transparency between such hosts is accomplished. End-to-end transparency is restored in the Internet to the extent of IPv4+4 deployment. When (if ever) deployment becomes complete then IP address transparency will also be fully restored. Note that this is accomplished without replacing or even reconfiguring routers. Backbone operators, for example, may remain completely unaffected[6].

As the number of hosts reachable via IPv4+4 increases, organizations that had no NATs before may see the benefits of setting up their own address realms. By doing so, they have the opportunity to install new equipment without obtaining more public IP addresses. The existing nodes need not be renumbered; public addresses may remain to be used inside the realm even for communicating with the outside world. In addition, the routing information of the realm may be hidden from the outside world. The address translation function of realm gateways is required only for communicating with IPv4 only hosts. As transition progresses it will be invoked less frequently and can be completely removed once the majority of the nodes have transitioned. This way IPv4+4 provides a way out of NATs.

If routers are also upgraded in an area, the special role of the realm gateways diminishes and they simply reduce to ordinary routers. In addition, the borders of address realms "dissolve" and routers in a private realm become no different from routers in the public address realm. In fact, the notion of address realm also disappears. IPv4+4 becomes "classless" by softening up the strict distinction between level 1 and 2 parts of the address. If the transition goes on in this direction then eventually we get back to a totally transparent Internet where all routers are equal and where the address space is 64-bits. We note, however, that there may be no need ever to upgrade routers, if the operator community decides so. Also, selective regions of the network may decide to upgrade independently.

One benefit of the above transition process is that the upgrade of hosts and routers are de-coupled and may happen at different pace — or, in case of routers may not happen at all. The most complex transition tool is the NAT itself. This provides communication between hosts if no native IPv4 or IPv4+4 path is available. There is no transition mechanism to allow communication between certain hosts (e.g., IPv4 only hosts in different address realms). However, such possible lack of reachability would not discourage starting the transition, as it is already in place today. In contrast, it provides an incentive to transition as transition provides the missing reachability.

6 Further extension of the address space

Currently, approximately a 24-bit address space is designated as private [10]. Since private realms may use only such addresses, the "effective size" of the extended address space is very roughly $32 + 24 = 56$ bits only. Furthermore, this address space is hard to utilize efficiently, as the vast majority of the current public addresses do not have a corresponding private address realm. Also, it is naive to expect private address realms to be as large as to efficiently utilize the entire private address space. Thus we conclude that an effective new address space might be even smaller than 56-bit. This poses questions regarding the expected lifetime of IPv4+4 addresses.

To add more addresses another level, level 0 can be introduced. This means a new network, in which the current Internet is merely a "private address realm" connected via one or more *level 0 realm gateways* (like node **M** in Figure 3). The realm gateways connecting level 1 and level 2 become *level 1 realm gateways*. The nodes in the current Internet then would have a number of level 0 address parts and the packet format would be double encapsulation[7].

The level 0 address space has to be disjoint from the addresses used in both level 1 and level 2. In addition, routers at level 1 and level 2 should have routing information on

[6] Of course, if a router has not been upgraded, its control plane cannot be reached from outside the address realm for example for management purposes. But since routers are usually managed from within the same domain this problem may not be serious. In addition, a control software upgrade of the router solves this problem.

[7] If this feature is desirable, then IPv4+4 hosts should be upgraded with preparedness to this scenario from day one.

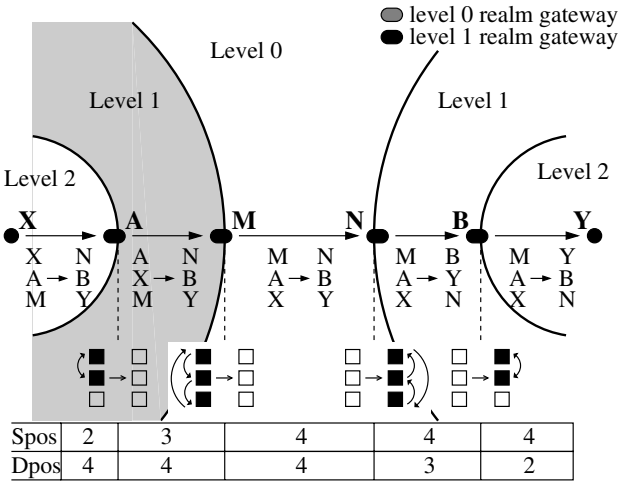

Figure 3. Further extending the address space.

the level 0 addresses. This allows such routers to route toward level 0 realm gateways. The level 1 realm gateways need to perform the same address swap operations as described in Section 4.1, while level 0 realm gateways need to cycle through the three address parts. The realm gateway operations are illustrated in Figure 3. The state of the addresses are reflected in the `SPos` and `DPos` fields using the values 2, 3 and 4, as shown in Figure 3.

The resulting address space is quite sizable, although not as large as the 128-bit IPv6 address space[8]. If, for example, one class A address block is used as the level 0 address space, then the effective size of the total address space is approximately $24 + 32 + 24 = 80$ bits.

The level 0 extension may likely be introduced by the efforts of network operators. While level 2 realms would most likely be introduced independently by individual sites or organizations, the level 0 infrastructure can be added only as a global effort that requires more co-ordination.

Finally, we note that another direction of address space extension is possible, namely by introducing level 3 inside private address realms, instead of level 0. However, for us it seems that the private address space available is sufficiently large for most private address realms to make such nested realms unnecessary.

7 Comparison to IPv6

The major benefits of IPv4+4 over IPv6 are its incentives, backwards compatibility, and the ease of transition. Due to the backward compatibility of the packet header and addressing, the transition can be gradually started, gradually blending to the new architecture. There is no need for temporary transition mechanisms (such as tunneling, tunnel brokers, 6to4, 6over4 or DSTM; see [29]), all new mechanisms are final. There is no need for a new addressing plan, dual routing, new network management tools, new routing protocols or new routers. The transition requires little new software and minimal changes to a running network. Full backward compatibility is maintained even when all hosts have already transitioned. IPv4+4 and IPv4 only hosts and networks can co-exit without new overhead. On the negative side, the IPv4+4 architecture is far from being as clean as IPv6. Many of its features (e.g., the header format) include a compromise for backward compatibility. The extended address space is substantially smaller than that of IPv6. If operators and users choose to undergo the IPv6 transition, IPv4+4 is not needed. However, if IPv4 and NATs prevail, IPv4+4 provides a plausible solution to the known problems discussed in this paper.

[8] We note, however, that 64 bits of the IPv6 address is used solely to address hosts within a subnet. In IPv4 (and 4+4) much less bits are used for that purpose.

8 Implementation

We have implemented an early version of IPv4+4 under the Linux 2.4.18 operating system in the form of a kernel module [30]. It contains roughly 2000 lines of C code that includes both the end-host and the realm gateway functionality. The end-host functions are achieved by a transparent protocol translation mechanism similar to [19]. IPv4+4 addresses are mapped to 32-bit *peer identifiers* that are of local significance only and are taken from a yet unused block of the IPv4 address space. The module transparently translates between IPv4 packets with peer identifiers and IPv4+4 packets. Applications are only presented with peer identifiers. An API is available for new applications to manipulate and query mappings. Mapping between peer ids and 4+4 addresses are established by incoming IPv4+4 packets, DNS queries and API calls. DNS replies are also captured and the applications are presented by a peer id only, instead of the IPv4+4 address. ICMP related functions, described in Section 4.3 are also implemented. For more details on the implementation, please refer to [30].

Our experience is, that all applications, that do not carry IP addresses in the payload, work unmodified. This includes the web, ssh, scp, telnet, ping (although they display the peer id instead of the 4+4 addresses). We have also performed some preliminary performance measurements and found that the overhead of address swaps in realm gateways is about 1.8 microseconds compared to the IPv4 case[9]. In short, we found that by loading a relatively small kernel module, most functions of IPv4+4 can be made to work

[9] Our measurements were taken on a 1GHz Pentium III with 256K memory.

with little overhead. Detailed discussion of implementation strategies, experiences and measurements will be the topic of a forthcoming report [30].

9 Conclusion

In this paper, we defined IPv4+4, a new address extension architecture for the Internet. IPv4+4 relies on existing private address realms and NATs and represents an evolutionary approach to address extension. Encapsulation is used as the main tool to maintain backward compatibility with existing routers that need not be modified. IPv4+4 is simple, affects only the network layer and retains the existing semantics of names and addresses. IPv4+4 provides a lightweight, well defined, incentive-driven transition process and can be incrementally deployed today. Upgraded hosts can immediately gain access to upgraded hosts in all other realms, while existing communication is not influenced in any way. Early implementation experiences are also reported.

Acknowledgments The authors wish to thank the help of Prof. Andrew Campbell in preparing this paper as well as the constructive aid of the anonymous reviewers.

References

[1] J. Postel, "Extensible Field Addressing," *Internet RFC 730,* May 1977.

[2] J. Postel, "Internet Control Message Protocol," *Internet RFC 792,* September 1981.

[3] J. Mogul, S. Deering, "Path MTU discovery," *Internet RFC 1191,* November 1990.

[4] Z. Wang, J. Crowcroft, "A Two-Tier Address Structure for the Internet: A Solution to the Problem of Address Space Exhaustion," *Internet RFC 1335,* May 1992.

[5] Minutes of the Address Extension by IP Option Usage BOF, *proceedings of 29th IETF meeting,* Seattle, April 1994.

[6] Minutes of the Address Lifetime Expectations working group, *proceedings of 29th IETF meeting,* Seattle, April 1994.

[7] P. Francis, "Pip Near-term Architecture," *Internet RFC 1621,* May 1994.

[8] P. Francis "Addressing in Internetwork Protocols," PhD Thesis, *University College London,* available at `www.ingrid.org/francis/thesis.ps.gz`, September 1994.

[9] R. Hinden, "Simple Internet Protocol Plus White Paper," *Internet RFC 1710,* October 1994.

[10] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, E. Lear, "Address Allocation for Private Internets," *Internet RFC 1918,* February 1996.

[11] R. Hinden, "New Scheme for Internet Routing and Addressing (ENCAPS) for IPNG," *Internet RFC 1955,* June 1996.

[12] I. Castineyra, N. Chiappa, M. Steenstrup, "The Nimrod Routing Architecture," *Internet RFC 1992,* August 1996.

[13] C. Perkins, "Minimal Encapsulation within IP," *Internet RFC 2004,* October 1996.

[14] M. O'Dell, "8+8 – An Alternate Addressing Architecture for IPv6," *Internet Draft,* named as draft-odell-8+8-00, Work in progress, November 1996.

[15] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," *Internet RFC 2460,* December 1998.

[16] Z. Turányi, A. Valkó, "4+4: Expanding the Internet Address Space without IPv6," *Ericsson Internal Report,* August 1999.

[17] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," *Internet RFC 2663*, August 1999.

[18] M. Crawford, A. Mankin, T. Narten, J. Stewart, L. Zhang, "Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6," *Internet Draft,* named as draft-ietf-ipngwg-esd-analysis-05, Work in progress, October 1999.

[19] K. Tsuchiya, H. Higuchi, Y. Atarashi, "Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)," *Internet RFC 2767,* February 2000.

[20] A. Gulbrandsen, P. Vixie, L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," *Internet RFC 2782,* February 2000.

[21] B. Carpenter, "Internet Transparency," *Internet RFC 2775,* February 2000.

[22] M. Crawford, "Router Renumbering for IPv6," *Internet RFC 2894,* August 2000.

[23] T. Hain, "Architectural Implications of NAT," *Internet RFC 2993,* November 2000.

[24] G. Huston, "To NAT or IPv6 – That is the question," *Satellite BroadBand magazine,* available at the author's page `http://www.telstra.net/gih`, December 2000.

[25] M. Holdrege, P. Srisuresh, "Protocol Complications with the IP Network Address Translator," *Internet RFC 3027,* January 2001.

[26] M. Gritter, D. R. Cheriton, "An Architecture for Content Routing Support in the Internet," *Usenix Symposium on Internet Technologies and Systems,* `http://gregorio.stanford.edu/triad`, March 2001.

[27] P. Francis, R. Gummadi, "IPNL: A NAT-Extended Internet Architecture," *SIGCOMM'01,* August 2001.

[28] M. Borella, J. Lo, D. Grabelsky, G. Montenegro, "Realm Specific IP: Framework," *Internet RFC 3102,* October 2001.

[29] The IETF Next Generation Transition (ngtrans) working group, `http://www.ietf.org`

[30] The IP4+4 project webpage at `http://ipv44.comet.columbia.edu`