# NAT++: Address Sharing in IPv4

*by Geoff Huston, APNIC*

In this article I examine the topic that was discussed in a session at the 74th meeting of the *Internet Engineering Task Force* (IETF) in March 2009, about *Address Sharing* (the SHARA BOF)[0], and look at the evolution of *Network Address Translation* (NAT) architectures in the face of the forthcoming depletion of the unallocated IPv4 address pool.

Within the next couple of years we will run out of the current supply of IPv4 addresses. As of the time of writing this article, the projected date when the *Internet Assigned Numbers Authority* (IANA) pool will be depleted is August 3, 2011, and the first *Regional Internet Registry* (RIR) will deplete its address pool about March 20, 2012.

Irrespective of the precise date of depletion, the current prediction is that the consumption rate of addresses at the time when the free pool of addresses is exhausted will probably be running at some 220 million addresses per year, indicating a deployment rate of some 170–200 million new services per year using IPv4. The implication is that the Internet will exhaust its address pool while operating its growth engines at full speed.

How quickly will IPv6 come to the rescue? Even the most optimistic forecast of IPv6 uptake for the global Internet is measured in years rather than months following exhaustion, and the more pessimistic forecasts extend into multiple decades.

For one such analysis using mathematical modelling techniques, refer to Jean Camp's work[1]. One of the conclusions from that 2008 study follows: "There is no feasible path which results in less than years of IPv4/IPv6 co-existence. Decades is not unreasonable."

The implication of this conclusion is that we will need to operate a dual-stack Internet for many years to come, and the associated implication is that we will have to make the existing IPv4 Internet span a billion or more new deployed services—and do so with no additional address space.

So how are we going to make the IPv4 address pool stretch across an ever-larger Internet?

Given that the tool chest we have today is the only one available, there appears to be only one answer to this question: Use *Network Address Translators,* or NATs.

For a description of how NATs work and some of the terminology used to describe NAT behavior, refer to the article "Anatomy: A Look Inside Network Address Translators," published in this journal[2].

Today NATs are predominately edge devices that are bundled with DSL modems for residential access, or bundled with routing and security firewall equipment for small to midsize enterprise use as an edge device. The generic model of NAT deployment currently is a small-scale edge device that generally has a single external-side public IP address and an internal-side private IP network address (often network 10). The NAT performs address and port translation to map all currently active sessions from the internal addresses to ports on the public IP address. This NAT deployment assumes that each edge customer has the unique use of a public IP address (refer to Figure 1).

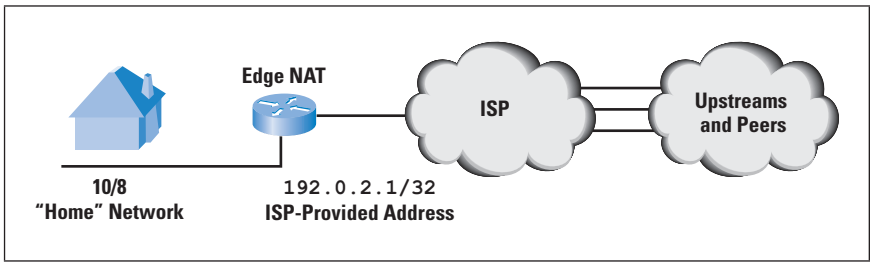



*Figure 1: Conventional NAT Deployment*

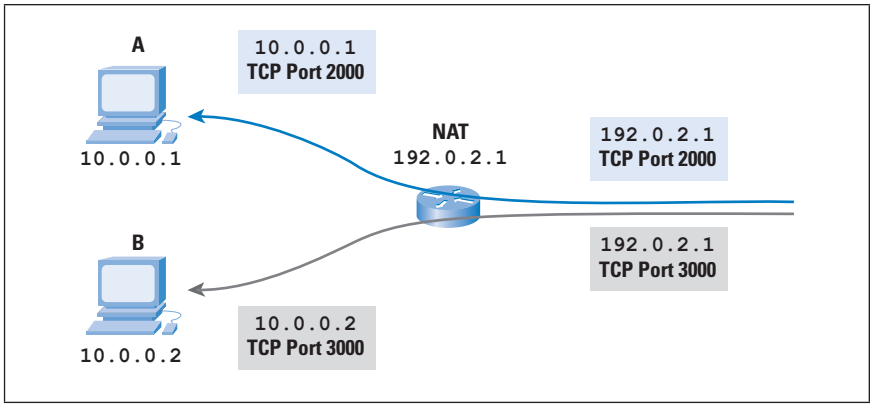
The question provoked by IPv4 address exhaustion is what happens when there are no longer sufficient IPv4 addresses to provide this 1:1 mapping between customers and public IPv4 addresses? In other words, what happens when there are simply not enough IPv4 addresses to allow all customers to have exclusive use of their own unique IPv4 address?

This question has only two possible answers. One is for no one to use IPv4 addresses at all, on the basis that the entire Internet has migrated to use IPv6. But this answer appears to be an uncomfortable number of decades away, so we need to examine the other answer: If there are not enough addresses to go around, then we will have to *share* them.

But isn't sharing IP addresses impossible in the Internet architecture? The IP address in a packet header determines the destination of the packet. If two or more endpoints share the same address, then how will the network figure out which packets go to which endpoint? It is here that NATs and the transport layer protocols, the *Transmission Control Protocol* (TCP) and the *User Datagram Protocol* (UDP), come together. The approach is to use the *port address* in the TCP and UDP header as the distinguishing element.

For example, in Figure 2, incoming TCP packets with TCP port address 2000 may need to be directed to endpoint A, while incoming TCP packets with TCP port address 3000 need to be directed to endpoint B. The incoming TCP packets with a port address of 2000 are translated to have the private IP address of endpoint A, and incoming TCP packets with a port address of 3000 are translated to have the private address of endpoint B.

*Figure 2: Address Sharing with NATs*




As long as you restrict yourself to applications that use TCP or UDP, you don't rely on receiving *Internet Control Message Protocol* (ICMP) packets, and you don't use applications that contain IP addresses in their payload, then you might expect this arrangement to function.

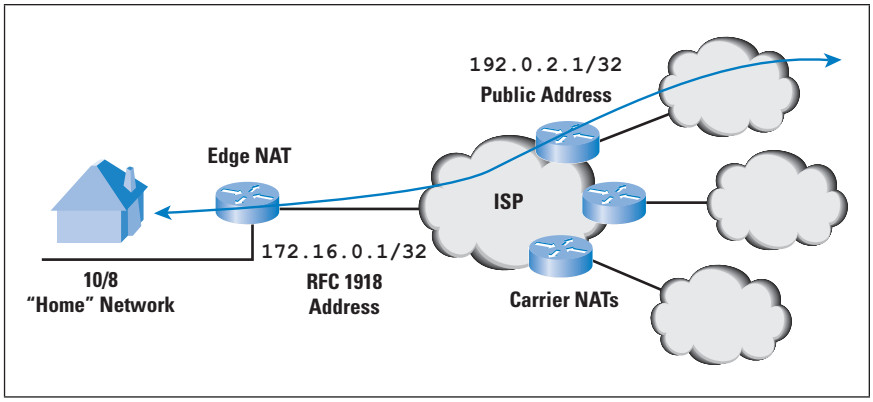
ICMP is a problem because the ICMP packet does not contain a TCP or UDP transport layer. All that a NAT sees in the ICMP packet is its own external address as the destination IP address. To successfully deliver an ICMP packet through a NAT, the NAT needs to perform a more complex function that uses the ICMP-encapsulated IP header to select the original outbound combined IP + TCP header or IP + UDP header in the ICMP payload. The source IP address and transport protocol port address in the ICMP payload are then used to perform a lookup into the NAT binding table and then perform two mappings: one on the ICMP header to map the destination IP address to the internal IP address, and the second on the payload header where the source IP address and port number are changed to the interior-side values, and the checksums altered as appropriate. Now in most cases ICMP really is not critical, and a conservative NAT implementation may elect to avoid all that packet inspection and simply discard all incoming ICMP messages, but one message that is important is the ICMP *packet-too-large-and-fragmentation-disabled* message used in IPv4 *Path MTU Discovery*[3].

Sharing IP addresses is fine in theory, but how can we achieve it in practice? How can many customers, already using NATs, share a single public IP address?

### Carrier-Grade NATs

One possible response is to add a further NAT into the path. In theory the *Internet Service Provider* (ISP) could add NATs on all upstream and peer connections, and perform an additional NAT operation as traffic enters and leaves the ISP's network. Variations of this approach are possible, placing the ISP NATs at customer aggregation points within the ISP's network, but the principle of operation of the ISP NAT is much the same.

Figure 3: Carrier NATs




The edge NATs translate between private address pools at each customer's site and an external address provided by the ISP, so nothing has changed there. The change in this model is that the ISP places a further NAT in the path within the ISP network, so that a set of customers is then sitting behind a larger NAT inside the ISP's network, as shown in Figure 3.

This scenario implies that the external address that the ISP provides to the customer is actually yet another private address, and the ISP's NAT performs yet another transform to a public address in this second NAT. In theory this NAT is just a larger version of an existing NAT with larger NAT binding space, higher packet-processing throughputs, and a comprehensive specification of NAT binding behavior. In practice it may be a little more complicated because at the network edge the packet rates are well within the processing capability of commodity processors, whereas in the core of the network there is an expectation of higher levels of robust performance from such units. Because it is intended that such a NAT handle thousands of customers and large numbers of simultaneous data flows and peak packet rates, it requires a performance level well beyond what is seen at the customer edge and, accordingly, such a NAT has been termed a *Carrier-Grade NAT* (CGN), or a *Large-Scale NAT* (LSN).

From the inside of the two NATs, not much has changed with the addition of the CGN in terms of application behavior. It still requires an outbound packet to trigger a binding that allows a return packet through to the internal destination, so nothing has changed there. Other aspects of NAT behavior, notably the NAT binding lifetime and the form of *Cone Behavior* for UDP, take on the more restrictive of the two NATs in sequence. The binding times are potentially problematic in that the two NATs are not synchronized in terms of binding behavior. If the CGN has a shorter binding time, it is possible for the CGN to misdirect packets and cause application-level problems. However, this situation is not overly different from a single-level NAT environment where aggressively short NAT binding times also run the risk of causing application-level problems when the NAT drops the binding for an active session that has been quiet for an extended period of time.

However, one major assumption is broken in this structure, namely that an IP address is associated with a single customer. In this model a single public IP address may be used simultaneously by many customers at once, albeit on different port numbers. This scenario has obvious implications in terms of some current practices in filters, firewalls, "black" and "white" lists, and some forms of application-level security and credentials where the application makes an inference about the identity and associate level of trust in the remote party based on the remote party's IP address.

This approach is not without its potential operational problems as well. For the ISP, service resiliency becomes a critical concern in so far as moving traffic from one NAT-connected external service to another will cause all the current sessions to be dropped, unless the internal ISP network architecture uses a transit access network between the CGNs and the external transit providers. Another concern is one of resource management in the face of potentially hostile applications. For example, an end host infected with a virus may generate a large amount of probe packets to a large range of addresses. In the case of a single edge NAT, the large volumes of bindings generated by this behavior become a local resource management problem because the customer's network is the only affected site. In the case where a CGN is deployed, the same behavior starts to consume binding space on the CGN and, potentially, can starve the CGN of external address bindings. If this problem is seen to be significant, the CGN would need to have some form of external address rationing per internal client in order to ensure that the entire external address pool is not consumed by a single errant customer application. This "rationing" would have the unwanted effect of forcing the ISP to deny access to its customers.

The other concern here is one of scalability. Although the greatest leverage of the CGN in terms of efficiency of usage of external addresses occurs when the greatest numbers of internal edge-NAT-translated clients are connected, there are some real limitations in terms of NAT performance and address availability when an ISP wants to apply this approach to networks where the customer population is in the millions or larger. In this case the ISP is required to use an IPv4 private address pool to number every client. But if all customers use network 10 as their "internal" network, then what address pool can the ISP use for its private address space? One of the few answers that come to mind is to deliberately partition the network into numerous discrete networks, each of which can be privately numbered from the smaller private address pool of `172.16.0.0/12`, allowing for some 600,000 or so customers per network partition, and then use a transit network to "glue" together the partitioned elements, as shown in Figure 4.

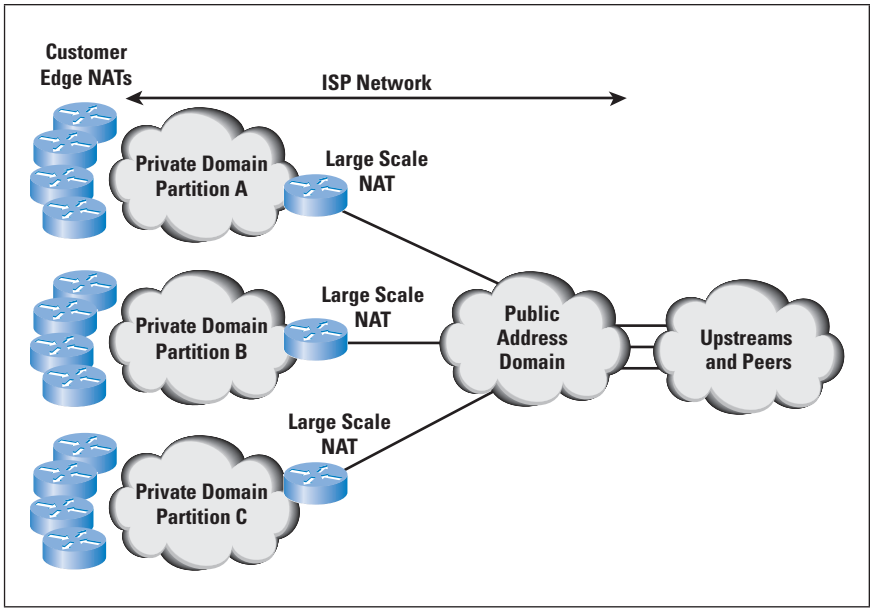



*Figure 4: Multiple Carrier NAT Deployment Using Network Partitioning*

The advantage of the CGN approach is that for the customer nothing changes. Customers do not need to upgrade their NAT equipment or change them in any way, and for many service providers this motivation is probably sufficient to choose this path. The disadvantages of this approach lie in the scaling properties when looking at very large deployments, and the problems of application-level translation, where the NAT attempts to be "helpful" by performing deep packet inspection and rewriting what it thinks are IP addresses found in packet payloads. Having one NAT do this rewriting is bad enough, but loading them up in sequence is a recipe for trouble!

Are there alternatives?

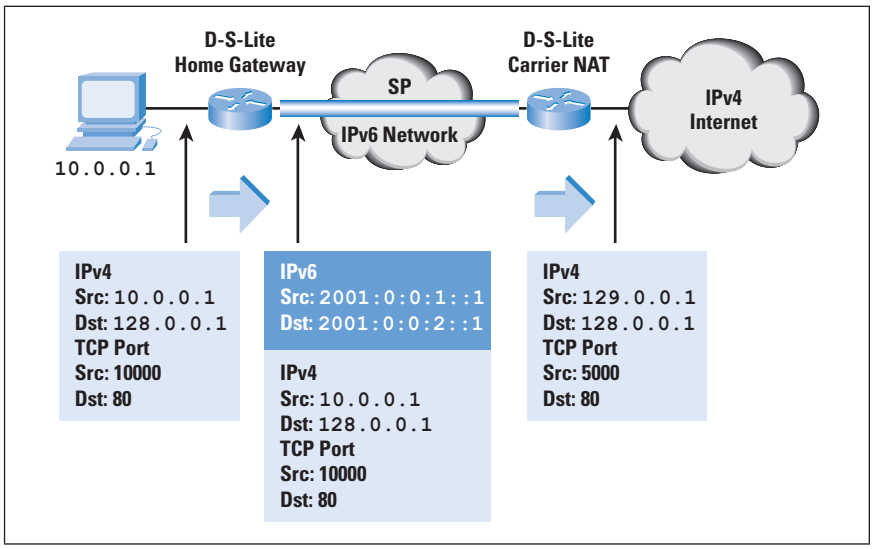
### Dual-Stack Lite and Carrier-Grade NATs

One rather elegant alternative is described by Alain Durand and others in an Internet Draft "Dual-stack lite broadband deployments post IPv4 exhaustion"[4]. The assumption behind this approach is that the ISP's network infrastructure needs to support IPv6 running in native mode in any case, so is there a way in which the ISP can continue to support IPv4 customers without running IPv4 internally?

Here the customer NAT is effectively replaced by a tunnel ingress/egress function in the *Dual-Stack Lite Home Gateway*. Outgoing IPv4 packets are not translated, but are encapsulated in an IPv6 packet header, where the IPv6 packet header contains a source address of the carrier side of the home gateway unit and a destination address of the ISP's gateway unit. From the ISP's perspective, each customer is no longer uniquely addressed with an IPv4 address, but instead is addressed with a unique IPv6 address. The customer's interface to the ISP network, the Home Gateway, is configured with this IPv6 address as the customer end of the IPv4-in-IPv6 tunnel, where the other end of the tunnel is the IPv6 address of the ISP's Dual-Stack Lite Gateway unit.

The service provider's Dual-Stack Lite gateway unit performs the IPv6 tunnel termination and a NAT translation using an extended local binding table. The "interior" NAT address is now a 4-tuple of the IPv4 source address, protocol ID, and port, plus the IPv6 address of the home gateway unit, while the external address remains the triplet of the public IPv4 address, protocol ID, and port. In this way the NAT binding table contains a mapping between interior "addresses" that consist of IPv4 address and port plus a tunnel identifier and public IPv4 exterior addresses. This way the NAT can handle a multitude of network 10 addresses, because the addresses can be distinguished by different tunnel identifiers. The resultant output packet following the stripping of the IPv6 encapsulation and the application of the NAT function is an IPv4 packet with public source and destination addresses. Incoming IPv4 packets are similarly transformed, where the IPv4 packet header is used to perform a lookup in the Dual-Stack Lite gateway unit, and the resultant 4-tuple is used to create the NAT-translated IPv4 packet header plus the destination address of the IPv6 encapsulation header (refer to Figure 5).

*Figure 5: Dual-Stack Lite*




The advantage of this approach is that now only a single NAT is needed in the end-to-end path because the functions of the customer NAT are now subsumed by the carrier NAT. This scenario has some advantages in terms of those messy "value-added" NAT functions that attempt to perform deep packet inspection and rewrite IP addresses found in data payloads. There is also no need to provide each customer with a unique IPv4 address, public or private, so the scaling limitations of the dual-NAT approach are also eliminated. The disadvantages of this approach lie in the need to use a different *Customer Premises Equipment* (CPE) device, or at least one that is reprogrammed. The device now requires an external IPv6 interface and at a minimum an IPv4 or IPv6 tunnel gateway function. The device can also include a NAT if desired, but it is not required in terms of the basic Dual-Stack Lite architecture.

This approach pushes the translation into the middle of the network, where the greatest benefit can be derived from port multiplexing, but it also creates a critical hotspot for the service itself. If the carrier NAT fails in any way, the entire customer base is disrupted. It seems somewhat counter intuitive to create a resilient network with stateless switching environments and then place a critical stateful unit in the middle! So is there an approach that can push this translation back to the edges while avoiding a second NAT in the carrier's network?

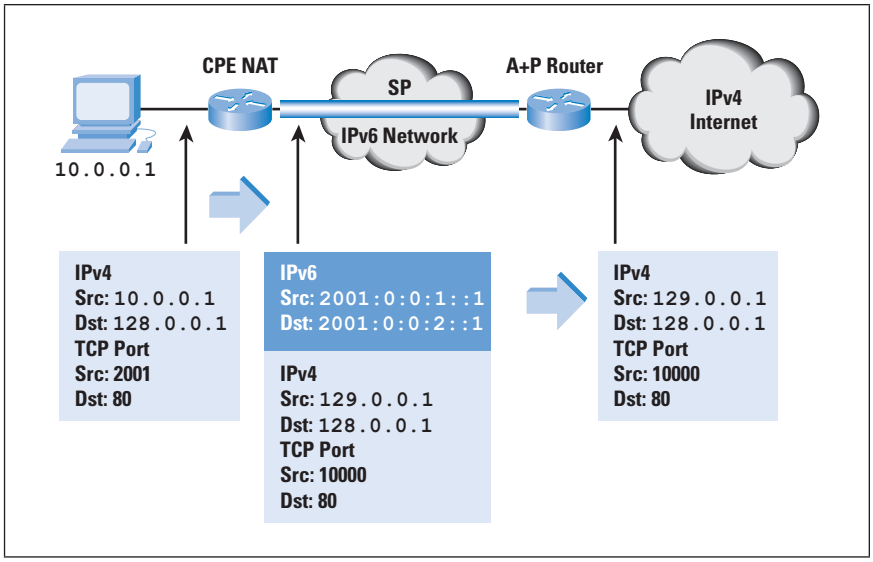
### The Address Plus Port Approach

The observation here is that CPE NATs currently map connections into the 16-bit port field of the single external address. If the CPE NAT could be coerced into performing this mapping into 15 bits of the port field, then the external address could be shared between two edge CPE devices, with the leading bit of the port field denoting which CPE device. Obviously, moving the bit marker across the port field would allow more CPE devices to share the one address, but it would reduce the number of available ports for each CPE device in the process.

The theory is again quite simple. The CPE NAT is dynamically configured with an external address, as happens today, and a port range, which is the additional constraint. The CPE NAT performs the same function as before, but it is now limited in terms of the external ports it can use in its NAT bindings to those that lie within the provided port range, because some other CPE may be concurrently using the same external IP address with a different port range.

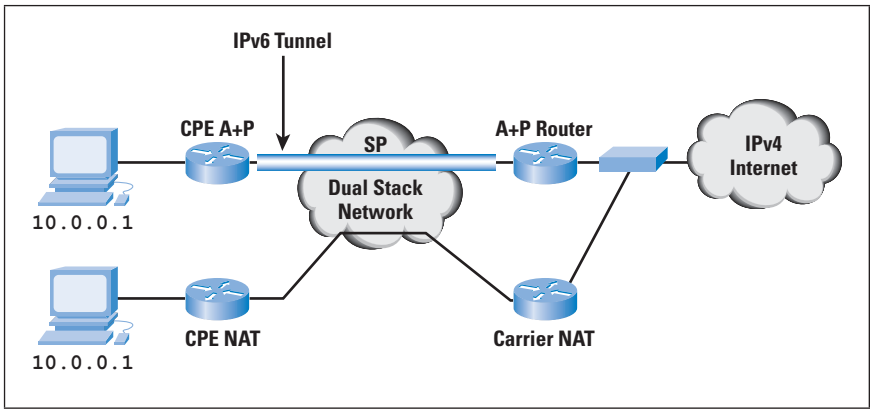
For outgoing packets this limitation implies only a minor change to the network architecture, in that the RADIUS[9] exchange to configure the CPE now must also provide a port range to the CPE device. However, the case of incoming packets is more challenging. Here the ISP must forward the packet based not only on the destination IP address, but also on the port value in the TCP or UDP header. A convenient way to forward the packet is to take the Dual-Stack Lite approach and use an IPv4-in-IPv6 tunnel between the CPE and the external gateway (Figure 6). This gateway, or *Address Plus Port* (A + P) router, needs to be able to associate each address and port range with the IPv6 address of a CPE device, which it can learn dynamically as it decapsulates outgoing packets. Corresponding incoming packets are encapsulated in IPv6 using the IPv6 destination address that it has learned previously. In this manner the NAT function is performed at the edge, much as it is today, and the interior device is a more conventional form of tunnel server.

*Figure 6: Address Plus Port Framework*




This approach relies on every CPE device being able to operate using a restricted port range, to perform IPv4-in-IPv6 tunnel ingress/egress functions, and to act as an IPv6 provisioned endpoint for the ISP network, which is perhaps an unrealistic hope. Further modifications to this model (Figure 7) propose the use of an accompanying CGN operated by the ISP to handle those CPE devices that cannot support these Address Plus Port functions.

*Figure 7: Combined Address Plus Port and Carrier Grade NAT*




If the port range assigned to the CPE is from a contiguous range of port values, then this approach could exacerbate some known problems with infrastructure protocols. There are *Domain Name System* (DNS) problems with guessable responses. The so-called "Kaminsky Attack" on the DNS[5, 6] is one such example where the attack can be deflected, to some extent, by using a randomly selected port number for each DNS query. Restricting the port range could mitigate the efficacy of such measures under certain conditions.

However, despite such concerns, the approach has some positive aspects. Pushing the NAT function to the edge has some considerable advantage over the approach of moving the NAT to the interior of the network.

The packet rates are lower at the edge, allowing for commodity computing to process the NAT functions across the offered packet load without undue stress. The ability for an end-user's application to request a particular NAT binding behavior by speaking directly with the local NAT using the *Internet Gateway Device Protocol,* as part of the *Universal Plug and Play* (UPnP)[7] framework, will still function in an environment of edge NATs operating with restricted port ranges. Aside from the initial provisioning process to equip the CPE NAT with a port range, the CPE, and the edge environment is largely the same as in today's CPE NAT model.

That is not to say that this approach is without its negative aspects, and it is unclear as to whether the perceived benefits of a "local" NAT function outweigh the problems associated with this model of address sharing. The concept of port "rationing" is a very suboptimal means of address sharing, given that after a CPE device has been assigned a port range those port addresses are unusable by any other CPE. The prudent ISP would assign to each CPE device a port address pool equal to some estimate of peak demand, so that, for example, each CPE device would be assigned 1,000 ports, allowing a single external IP address to be shared across only 60 such CPE clients. Neither the Carrier-Grade NAT approach nor the Dual-Stack Lite approach attempts this form of rationed allocation, allowing the port address pool to be treated as a common resource, with far higher levels of usage efficiency through dynamic management of the port pool.

The difference here is that in the dynamically managed approach any client can use the currently unused port addresses, whereas in the rationed approach each client has access to a fixed pool of port addresses that cannot be shared with any other client—even when the client does not need them. The difference here parallels the difference in network efficiency between time-division multiplexed synchronous circuits and asynchronous packets at Layer 2 in the network model. In the Address Plus Port framework the leverage obtained in terms of making efficient use of coopting these additional 16 bits of port address into the role of additional bits of client identifier address space is reduced by the imposition of a fixed boundary between customer and ISP use in the port address plan. The central NAT model of a CGN effectively pools the port address range and facilitates far more efficient sharing of this common port address pool across a larger client base.

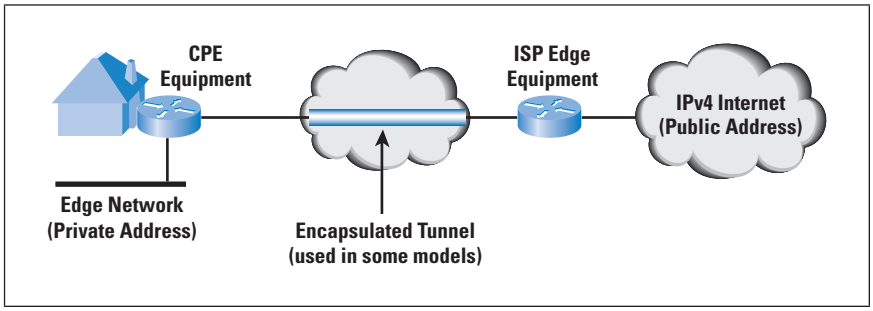
Alain Durand reported to IETF 74 on a data-collection experiment using a *Cable Modem Termination System* (CMTS) with 8,000 subscribers where the peak port consumption level was 40,000 ports, or a maximum average port consumption of 5 ports per subscriber in each direction. As Alain noted, this average value needs to be compared with the hundreds of ports consumed by a single client browsing a Web 2.0 or *Asynchronous Java and XML* (AJAX) site, but he also noted that a central model of port sharing does yield far higher levels of address-sharing efficiency than the Address Plus Port advanced allocation model.[8]

The other consideration here is that this approach constitutes a higher overhead for the ISP, in that the ISP must support both "conventional" CPE and Address Plus Port equipment. In other words, the ISP must deploy a CGN and support customer CPE using a two-level NAT environment in addition to operating the Address Plus Port infrastructure. Unless customers would be willing to pay a significant price premium for such an Address Plus Port service, it is unlikely that this option would be attractive for the ISP as an additional cost after the CGN cost.

### General Considerations with Address Sharing

The basic elements of any such approach to address sharing involve the CPE equipment at the edge, optionally some form of tunneling of traffic between the CPE and the carrier equipment, and carrier-provided equipment at the edge of the carrier's network (refer to Figure 8).

*Figure 8: Generic Architecture for Address Sharing*




A variety of technical solutions here involve these basic building blocks, so it is not true to say that this challenge is technically significant. But few ISPs have decided to proceed with large-scale deployment of any form of address-sharing technology for their IPv4 network infrastructure. So what is the problem here?

I suspect that the real concern is the consideration of the relevant business model that would guide this deployment. Today's Internet is large. It encompasses some 1.7 billion human users, a larger pool of devices, and hundreds of millions of individual points of control. If we want to change this deployed system, we will need copious quantities of money, time, and unity of purpose. So do we have money, time, and unity of purpose?

*Money is missing:* It could be argued that we have left the entire IPv6 transition effort to this late stage because of a lack of money. The main advantage of the Internet was that it was cheap. Packet sharing is intrinsically more efficient than circuit sharing, and the shift in functions of network service management from the network to the customer-owned and -operated endpoints implied further cost savings for the network operator. So the Internet model gained ascendency because for consumers it represented a cost-effective choice. It was cheap.

But what does IPv6 offer consumers? For existing Internet consumers it appears that IPv6 does not offer anything that they don't already have with IPv4—it offers mail, the web, various forms of voice services, and games. So consumers are not exactly motivated to pay more for the same services they already enjoy today.

In addition, it would appear that the ISP must carry this cost without incremental revenue from its customer base. But the ISP industry has managed to shave most of its revenue margins in a highly competitive industry, and at the same time lose control of services, their delivery, and their potentially lucrative revenue margins. Thus the ISP industry has been collectively idle in this area not because it cannot see the problem in terms of the imminent exhaustion of IPv4, but because it has little choice because of financial constraints that have prevented it from making the necessary longer-term investments in IPv6. So if the ISP industry has been unwilling to invest in IPv6 so far, then what incentive is there for it to invest in IPv6 and at the same time also invest in these IPv4 address-sharing measures? Is the lure of new, low-margin customers sufficient incentive to make such investments in this carrier-grade equipment? Or is the business case still insufficiently attractive?

*Time is missing:* The unallocated IPv4 address pool is already visibly waning. Without any form of last-minute rush, the pool will be around for the next 2 years, or until 2012 or so. But with any form of typical last-minute rush, this pool could be depleted in the coming months rather than in the coming years. Can we do what we need to do to get any of these approaches to a state of mass-market deployment in the next few months? All these approaches appear to be at the early stages of a timeline that starts with research and then moves on to development, prototyping, and trials; then to standards activity and industry engagement to orchestrate supply lines for end user equipment, ISP equipment, and definition of operational practices; then to product and service development; and finally, to deployment. For an industry that is the size of the Internet, "technical agility" is now an obsolete historical term. Even with money and unity of purpose this process will take some years, and without money—or even the lure of money—it becomes a far more protracted process, as we have seen already with IPv6 deployment.

And do we have *unity of purpose* here? Do we agree on an approach to address sharing that will allow players to perform their tasks? That will allow consumer product vendors to develop the appropriate product? That will allow application developers to develop applications that will operate successfully in this environment? That will allow the end user platform vendors to incorporate the appropriate functions in the operating system stacks? That will allow ISPs to integrate vendors' productions into their operational environments? Right now it is pretty clear that what we have is a set of ideas, each of which has relative merits and disadvantages, and no real unity of purpose.

It is easy to be pessimistic at this stage, given that the real concerns here appear to be related more to the factors associated with a very large industry attempting to respond to a very challenging change in the environment in which it operates. The question here is not really whether Address Plus Port routing is technically inferior to Dual-Stack Lite, or whether Carrier-Grade NATs are technically better or worse than either of these approaches. The question here is whether this industry as a whole will be able to sustain its momentum and growth across this hiatus. And, from this perspective, I believe that such pessimism about the future of the Internet is unwarranted.

The communications industry has undergone significant technological changes over the years, and this change is one more in the sequence. Some of these transformations have been radical in their effect, such as the introduction of the telephone in the late nineteenth century, whereas others have been more subtle, such as in the introduction of digital technology to telephony in the latter part of twentieth century, replacing the earlier analogue circuit model of telephony carriage. Some changes have been associated with high levels of risk, and we have seen a myriad of smaller, more agile players enter the market to lead the change while the more risk-averse enterprises stand back. On the other hand, other changes require the leverage of economies of scale, and we have seen market consolidation behind a smaller number of highly capitalized players.

My personal opinion is that the Dual-Stack Lite approach is the best one, because it appears to be technically elegant. I suspect, however, that the lowest-common-denominator fall-back position that this somewhat conservative industry will adopt will rely strongly on Carrier-Grade NATs, and the industry is likely to eschew the more complex support mechanisms required by the various permutations of Address Plus Port routing.

### Further Reading

[0] The Address Sharing BOF was held at IETF 74 in March 2009. The presentations and a summary of the session can be found as part of the proceedings of that meeting:
**http://www.ietf.org/proceedings/09mar/shara.html**

[1] **http://www.ripe.net/ripe/meetings/ripe-56/presentations/Camp-IPv6_Economics_Security.pdf**

[2] Geoff Huston, "Anatomy: A Look Inside Network Address Translators," *The Internet Protocol Journal*, Volume 7, No. 3, September 2004.

[3] Jeff Mogul and Steve Deering, "Path MTU Discovery," RFC 1191, November 1990.

[4] **draft-ietf-softwire-dual-stack-lite-00.txt**

[5] **http://www.doxpara.com/DMK_BO2K8.ppt**

[6] **http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html**

[7] http://en.wikipedia.org/wiki/Universal_Plug_and_Play

[8] http://www.ietf.org/proceedings/09mar/slides/shara-8/shara-8.htm

[9] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, June 2000.

[10] Egevang, K., and P. Francis, "The IP Network Address Translator (NAT)," RFC 1631, May 1994.

[11] Srisuresh, P., and D. Gan, "Load Sharing Using IP Network Address Translation (LSNAT)," RFC 2391, August 1998.

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

[13] Tsirtsis, G., and P. Srisuresh, "Network Address Translation—Protocol Translation (NAT-PT)," RFC 2776, February 2000.

[14] Hain, T., "Architectural Implications of NAT," RFC 2993, November 2000.

[15] Srisuresh, P., and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, January 2001.

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

[17] D. Senie, "Network Address Translator (NAT)-Friendly Application Design Guidelines," RFC 3235, January 2002.

[18] Srisuresh, P., J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox Communication Architecture and Framework," RFC 3303, August 2002.

[19] Daigle, L., and IAB, "IAB Considerations for Unilateral Self-Address Fixing (UNSAF) Across Network Address Translation," RFC 3424, November 2002.

[20] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN—Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489, March 2003.

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. The author of numerous Internet-related books, he is currently the Chief Scientist at APNIC. He was a member of the Internet Architecture Board (IAB) from 1999 until 2005, and served on the Board of the Internet Society from 1992 until 2001.
E-mail: gih@apnic.net