Impact on End-users by ISP IPv6 Deployment

Tim Niklas Gruel
Ruhr-Universität Bochum
Bochum, Germany
tim.gruel@rub.de

ABSTRACT

This paper presents an overview of different techniques to cope with the IPv4 address exhaustion. The impact on the end-users is on focus. First some early NAT based solutions are introduced. Later dualstack is discussed. At the end IPv6 only solutions, deployed by ISPs today, are presented. It turns out that the end-user, especially the technical experienced one, can experience benefits with IPv6. With IPv6 the internet gets back to its routs. All devices can talk to each other and central server are not mandatory anymore to establish communication.

KEYWORDS

IPv6, Tunneling, Translation, End-user Impact, NAT64, DNS64, 464XLAT, Dualstack

1 INTRODUCTION

In the late 90s it became clear, that the IPv4 address space is not sufficient to meet future demand. The popularity of the Internet especially in the end-user marked was heavily underestimated at the creation of IPv4. There are approximately 4 billion IPv4 addresses. Today there are over 8 billion humans on earth. That means that two humans need to share a single IPv4 address, not considering servers. Today, a typical family has multiple devices per person, including TVs, gaming consoles, smart phones, personal computers and more. This led to the assignment of the last available IPv4 block in the early 2010s. With the rise of smart home technology and IoT devices, the demand for IP addresses per household will not stop to increase. In this paper I will present a nearly complete overview of the different ideas to slow down the inevitable IPv4 exhaustion. Expect for the last solution, none of these is a long term one or even tries to solve the problem itself. ISPs tried to keep cost as low as possible. This led to keeping IPv4 addresses as long as possible.

First, ISPs started to assign only one public IP per customer. The customer had to use NAT44 in order to provide all his end devices with an internet connection. The end-user was the one getting a worse experience. Over time, assigning only one IPv4 per customer was too much for the ISPs. Especially with the rise of mobile internet and an even higher demand for addresses, IPv4 was not sufficient anymore. ISPs tried to provide more IP addresses, by assigning only private IPv4 addresses to customers, leading to NAT444. NAT itself has problems, which will be discussed in this paper. NAT444 makes the problems even worse.

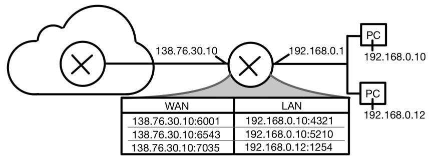
A few years later dualstack was provided by ISPs. Dualstack introduced the next generation of IP addresses, called IPv6 addresses. Compared to IPv4 addresses which are a 32 bit wide, IPv6 addresses are a 128 bit wide. IPv6 completely solves the problem of address exhaustion. Unfortunately IPv6 is not backwards compatible to IPv4. Thus all infrastructure needs to be renewed. The Internet is fully decentralized. It is not possible to perform such a transition on one particular day. Dualstack implements both, IPv4 and IPv6. Devices and servers supporting IPv6 can communicate over IPv6. In the case that one device is not yet capable of IPv6, IPv4 acts as a fallback. This is a perfect transition technology, but it is no solution for the IPv4 exhaustion problem. Each end-user still needs an public or private IPv4. The problem is not solved.

Lately, ISPs started to provide IPv6 only. This is also common in mobile networks. The devices natively speak to servers over IPv6. For legacy IPv4 servers, there are different translation mechanisms. This paper will take a closer look into NAT64 combined with DNS64. Moreover I will present 464XLAT. Both are similar to a NAT44, but translating between IPv6 and IPv4. The interesting thing is that these translations happen at the ISP, so the end-user only uses with IPv6. This has advantages for the end-user. Though, some legacy applications that depend on IPv4 cannot be used. For example, Github is not reachable over Ipv6. The DNS does not return a *AAAA* record.

At the end of this paper, tunneling IPv6 in IPv4 is presented. This happens only at ISP level and indirectly influences the end-user too. Mainly in better path availability, thus seamless internet service. There are various technologies. 6in4 and 6rd will be introduced shortly.

In the last few years, the adoption of IPv6 has increased rapidly. Above 40 percent of all google users access the site over Ipv6 [1]. This statistics really shows the trend of ISPs worldwide, deploying IPv6 only or dualstack. Nearly all mobile clients use IPv6 today and are able to interact with the Internet without notable disadvantages. Cisco predicts that the transition to IPv6 is finished in 2028 [17]. This means that nearly all end-users and servers world-wide communicate over IPv6. That has noticable advantages for the end-user regarding security, because firewalls can be managed easier. In addition it opens up entirely new options for innovative peer to peer technologies. Hosting a Server on a smart phone might sound strange today, but could be a usual thing in 10 years from now. The end-user is the one benefiting the most from unique, world-wide routable IP addresses.

Now I am going to turn back the time a bit and we are focusing on the problem, emerging in the late 90s. It becomes clear that the IPv4 address space is not big enough. ISPs are searching for a solution. The main goal is to assign the same public, globally routable IPv4 address to multiple devices. This is not meant for a long period of time, but much more as a quick fix, until the new standard IPv6 is developed.



WAN	LAN
138.76.30.10:6001	192.168.0.10:4321
138.76.30.10:6543	192.168.0.10:5210
138.76.30.10:7035	192.168.0.12:1254

Figure 1: Example of NAT44 with NAT-Translation table

2 EARLY SOLUTIONS

This section presents three different solution to extend the lifetime of IPv4: NAT44, A+P and NAT444. All these solutions aim to share one single IPv4 address between multiple end devices. It is important to understand theses technologies in order to correctly estimate the impact of IPv6 deployment by ISPs. Evaluating modern solutions alone could lead to the impression, that these solutions are better or worse than they really are. Thus this paper starts with NAT44.

2.1 NAT44

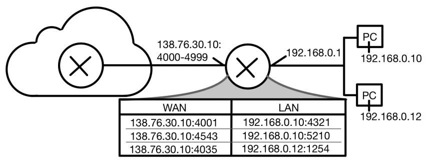
NAT44 is also known as *CPE NAT*. The word NAT stands for *Network Address Translation*. *NAT44* means that one IPv4 address is translated to multiple IPv4 addresses. CPE means *Customer Premises Equipment*. The whole NAT procedure is up to the end-user. The ISP only provides one single IPv4 address. Typically one public, globally routable IP is translated to multiple private IP addresses. There are three private IPv4 Blocks which are the following:

(1) 10.0.0.0/8
(2) 172.16.0.0/12
(3) 192.168.0.0/16

Everybody can use the IPv4 addresses in these blocks. Private IPv4 addresses will never be routed in the internet. Most end-users use the third option for their home network, but only with a /24 network. The third block remains 0 all the time.Thus they can use 254 different devices which is sufficient for most users. All these 254 devices share one public IPv4. All of them share the $2^{16} = 65536$ ports of the public IPv4 too. From the outside you cannot decide whether only one device or multiple devices are behind the NAT. In the example Figure 1 is a NAT-Translation table. As you can see, the PC with IP *192.168.0.10* cannot use all ports, because the PC with the IP *192.168.0.12* uses the Port *7035*. All ports are shared. This can lead to problems as we discuss later. NAT44 was the first solution to the IPv4 exhaustion problem. It is an easy fix, because the ISP has nothing to change, except now only delivering one single IP per customer. The end-user needs a NAT compatible router.

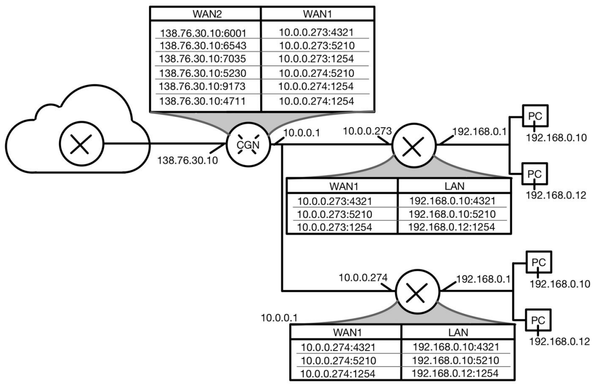
2.2 A+P

A+P means *Address Plus Port*. Over the time NAT44 was not a sufficient solution anymore. The demand for IPv4 rose and ISP were not able to provide one single public, globally routable IPv4 per customer. The idea is still similar to NAT44, but with *A+P* each costumer gets one IPv4 and a designated port range. With that system it is possible to share one IP to theoretically *65536* different customers [14]. In the example Figure 2 you can see that all internal



WAN	LAN
138.76.30.10:4001	192.168.0.10:4321
138.76.30.10:4543	192.168.0.10:5210
138.76.30.10:4035	192.168.0.12:1254

Figure 2: Example of A+P with NAT-Translation table



WAN2	WAN1
138.76.30.10:6001	10.0.0.273:4321
138.76.30.10:6543	10.0.0.273:5210
138.76.30.10:7035	10.0.0.273:1254
138.76.30.10:5230	10.0.0.274:5210
138.76.30.10:9173	10.0.0.274:1254
138.76.30.10:4711	10.0.0.274:1254

WAN1	LAN
10.0.0.273:4321	192.168.0.10:4321
10.0.0.273:5210	192.168.0.10:5210
10.0.0.273:1254	192.168.0.12:1254

WAN1	LAN
10.0.0.274:4321	192.168.0.10:4321
10.0.0.274:5210	192.168.0.10:5210
10.0.0.274:1254	192.168.0.12:1254

Figure 3: Example of NAT444 with translation tables

ports are mapped to port values between *4000* and *4999*. This is because the ISP assigned this port range to that particular customer. All internal devices combined can open *1000* ports. The biggest advantage of *A+P* is also the biggest downside. On the one hand, it is possible to supply more customers with public, globally routable IPv4 address, but on the other hand all these customers have a tiny available port range. Assuming the extreme case of *65536* different customers, each customer would only get one single port. Even when ports are dynamically assigned, the user experience at peak internet usage time would be unacceptable. Ports are important to distingush between different processes on a host. Each process that communicates with the network opens at least one socket and gets a port from the operating system. These ports are unsed to correctly deliver TCP or UDP packets. For example, a mail-programm opens a port to fetch the newest mails and a webbrowser opens ports to download different websites simultaniously. That is why each host needs multiple ports. Having multiple hosts in the network, which consume multiple ports each, can extend the provided port range. In case a customer is not supplied with sufficient ports by the ISP, the end-devices do not work as intended.

2.3 NAT444

NAT444 is also known as CGN. CGN means *Carrier Grade Network Address Translation*. With NAT44 one public, globally routable IPv4 address is translated to multiple private IPv4 addresses at ISP site. Each customer gets one private IPv4 address. Similar to NAT44, the customer than translates that private IPv4 into multiple private IPv4 address for all home devices. That means there is a double NAT. IP addresses get translated twice [20]. This leads to even more problems than NAT44 or A+P [14]. In example Figure 3 two end-users are connected to the CGN. The CGN has the public, globally routable IPv4 *138.76.30.10*. One customer gets assigned the private IPv4 *10.0.0.273* and the other one the private IPv4 *10.0.0.274*. Both operate a typical *192.168.0.0/16* Network. As you can see, both customers have a personal computer with the private IPv4 *192.168.0.10*. Yet the system works. After the end-user NAT translation, these personal computers have the IPv4 *10.0.0.273* and *10.0.0.274* with the corresponding ports. Then the CGN translates these private IPv4 addresses into the only public, globally routable IPv4 *138.76.30.10*. The CGN translates IP and Port and maps them to the corresponding customer. In a real world application, the CGN would control multiple public, gloablly routable IPv4 addresses and split them between all customers. Example Figure 3 only shows a cutout of the whole CGN[16].

END-USER IMPACT

NAT44, A+P and NAT444 cause similar problems and have a huge impact on the end-user.

The first problem is regarding location services. Typically it is possible to determine a rough geo-location regarding one single public IPv4. Especially with NAT444 one IPv4 is shared by multiple users. Thus it is not really possible to accurately guess the location. Some people might argue that this is a advantage too, because it increases privacy[15].

A serious problem is regarding spam. IPv4 addresses which are distributing spam, typically can be blacklisted. With a shared IPv4 address between multiple end-users this can lead to blocking of innocent end-users who just were unlucky to share a IP with an adversary. In case a server only supports IPv4 and a end-user is behind a NAT444, the end-user has a worse experience if the server is using IP addresses for repuation. If a mallicious user misbehaves, an innocent user using the same public IPv4 can get blocked or rate-limited [15].

Another problem is regarding peer-to-peer applications. To establish and maintain a peer-to-peer network it is necessary to ping a peer directly. This happens with a public IPv4 and a well known port. With NAT that is generally not possible. All devices share one IPv4 with all ports[15].

Similar to peer-to-peer it is not possible to host own servers behind a NAT. Web servers for example establish *https* connection over the well-known port *443*. With NAT44 it is theoretically possible to forward a port to one device. Thus it is possible to operate one server of a kind per port. For example one web server and one ftp server. With A+P or NAT444 this option is completely gone and it is not possible to host servers. That impacts end-users massively. The main idea of the internet is a decentralised network. Everyone should be able to set up servers and start communications without a central instance. NAT disabled this main idea of the internet[15].

Modern web applications nowadays usually require multiple ports to operate. With NAT these ports are limited and could run short. This is a serious problem, because as it is not possible to create more IPv4 addresses, it is not possible to create more Ports[15]. *A+P* makes this problem even worse, because the total available ports are even lower. As soon as all available ports are used, new processes cannot esatblish a internet connection

NAT444 could result in much fragmentation. Packets must travel from the end-device to the home router and from the home-router to the CGN. That are two fixed routes. Assuming that the first route can transport bigger packets, many packets need to be fragmented at the home router. This causes network overhead and performance degradation.

Apart from the obvious disadvantages, some protocols like DNS are less secure with NAT. Because of the *DNS poisoning attack*, a client starts new DNS requests with a random port. Assuming that an attacker cannot guess the port, the DNS server answers to that port. If an attacker can guess the port and answer to it first, the attacker can infiltrate wrong DNS entries. This is a serious security risk. With the use of NAT44 all devices share the available ports. The problem gets even worse with A+P or NAT444. If an attacker knows the assigned port range, the DNS cache poisoning attack is effective. IPSec needed to be upgraded to work with NAT44 and currently does not work with NAT444[15].

A general problem with NAT is, that it is a single point of failure. If a personal NAT router or a CGN router is not working anymore, all clients cannot access the internet. A NAT router is a lucrative attack target[15].

There are transport layer protocols that do not use ports at all. This is the case when multiplexing of the protcol is not needed. An example is the *Interior gateway protocol* which is used to exchange routing information. Such protocols do not work with *NAT*, because one IP address which should belong to one node is used by multiple nodes. This breaks the protocol.

In a nutshell, NAT has problems. It was meant as a quick fix until the next generation of IP is finalized. Unfortunately, NAT outlived it supposed lifetime by years. This is why modern solutions are required to get away from NAT and return to the original open and accessable internet.

3 MODERN SOLUTIONS

It became clear that IPv4 is not sufficient anymore. This is why IPv6 was standardized. IPv6 addresses are *128* bit wide. With IPv6 it is possible to assign $2^{128} = 3.4 * 10^{38}$ addresses. This number of addresses will suffice in the foreseable future. IPv6 solves the IPv4 exhaustion problem. Unfortunately IPv6 is not backwards compatible to IPv4. The first solution was to implement IPv6 alongside IPv4. This gives older applications the possibility to continue using IPv4 until they and the servers are upgraded. Assigning both an IPv4 address and IPv6 addresses is called *dualstack*. The two similar technologies *dualstack* and *dualstack lite* are introduced now. They are meant to be transition technologies from the world of IPv4 only to IPv6 only.

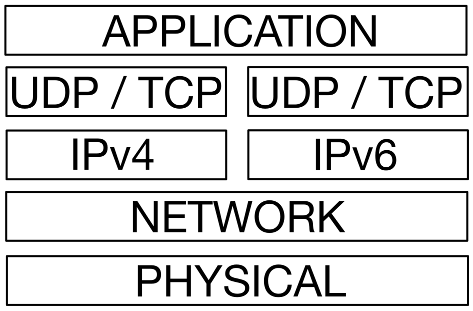


Figure 4: Dualstack IP layer implementation

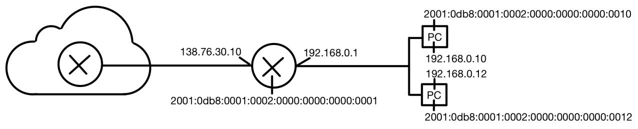
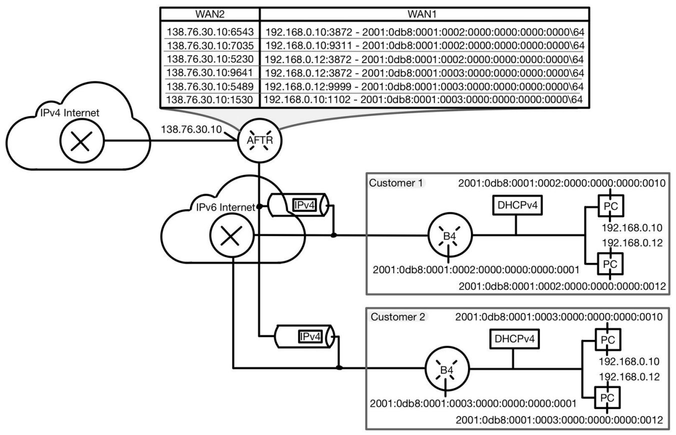

138.76.30.10
192.168.0.1
2001:0db8:0001:0002:0000:0000:0000:0001
2001:0db8:0001:0002:0000:0000:0000:0010
PC
192.168.0.10
192.168.0.12
PC
2001:0db8:0001:0002:0000:0000:0000:0012

Figure 5: Example of a Dualstack network



WAN2	WAN1
138.76.30.10:6543	192.168.0.10:3872 - 2001:0db8:0001:0002:0000:0000:0000:0000\64
138.76.30.10:7035	192.168.0.10:9311 - 2001:0db8:0001:0002:0000:0000:0000:0000\64
138.76.30.10:5230	192.168.0.12:3872 - 2001:0db8:0001:0002:0000:0000:0000:0000\64
138.76.30.10:9641	192.168.0.12:3872 - 2001:0db8:0001:0003:0000:0000:0000:0000\64
138.76.30.10:5489	192.168.0.12:9999 - 2001:0db8:0001:0003:0000:0000:0000:0000\64
138.76.30.10:1530	192.168.0.10:1102 - 2001:0db8:0001:0003:0000:0000:0000:0000\64

IPv4 Internet
138.76.30.10
AFTR
IPv6 Internet
IPv4
Customer 1
2001:0db8:0001:0002:0000:0000:0000:0010
DHCPv4
PC
B4
192.168.0.10
192.168.0.12
PC
2001:0db8:0001:0002:0000:0000:0000:0001
2001:0db8:0001:0002:0000:0000:0000:0012
IPv4
Customer 2
2001:0db8:0001:0003:0000:0000:0000:0010
DHCPv4
PC
B4
192.168.0.10
192.168.0.12
PC
2001:0db8:0001:0003:0000:0000:0000:0001
2001:0db8:0001:0003:0000:0000:0000:0012

Figure 6: Example of a dualstack Lite network

3.1 DUALSTACK

With *dualstack* the end-user gets a public, globally routable IPv4 address and a block IPv6 addresses from the ISP. With the IPv4 address, the end-user performs NAT44. This was discussed in the previous chapter. In addition to that private IPv4 address, all devices also get assigned a global, publicly routable IPv6 address [14]. A personal computer now has two implemented IP layers as in Figure 4. Old application that only support IPv4 work seamlessly without any modification required. With IPv6 an ISP typically get a */32* network containing $2^{128-32} = 2^{96}$ IPv6 addresses. The end-user then gets a */64* network. IPv6 addresses are noted as 8 two byte blocks divides by a colon. The example in Figure 5 shows a basic end-user *dualstack* network containing one router and two end-devices. The ISP got the *2001:0db8:0000:0000:0000:0000:0000:0000/32* block assigned. The router implements NAT44 and got the block *2001:0db8:0001:0002:0000:0000:0000:0000/64* assigned by the ISP. The router can now distribute the last 4 two byte blocks to the devices. In this example the two personal computers got the public, globally routable IPv6 *2001:0db8:0001:0002:0000:0000:0000:0010* and *2001:0db8:0001:0002:0000:0000:0000:0012*. In addition to that the router assigns the private IPv4 addresses *192.168.0.10* and *192.168.0.12* to the personal computers. An application on one personal computer can now use the private IPv4 address with all the disadvantages discussed earlier or use the IPv6 address.

3.2 DUALSTACK LITE

Dualstack Lite also called *DS Lite* is similar to *dualstack* from a end-device point of view. An end-device gets a public, globally routable IPv6 address and a private IPv4 address. The end-device needs an *dualstack* implementation as shown in Figure 4. In case the device wants to communicate with an IPv4 legacy server, the device has to use the IPv4 stack and the private IPv4. In all other cases the public, globally routable IPv6 is preferred. There is no translation between IPv4 and IPv6 or the other way around. All in all *dualstack* and *dualstack lite* does not differ for the end-device.

Compared to *dualstack*, the network traffic between the router of the customer and the ISP is IPv6 only. IPv4 packets are tunneled in IPv6. The IPv4 packet is part of the data field of the IPv6 packet. DNS is done over IPv6 only. The *B4* element will perform DNS for all clients in the network. Thus DNS is never routed to *AFTR*.

Figure 6 is an example of a basic *dualstack lite* setup. The customers router has a *B4* element. This *B4* element directly connects to the ISPs *AFTR*. The *B4* and *AFTR* create an IPv6 tunnel for IPv4 packets. NAT44 is only applied at ISP level. Though, each customer hosts his own DHCPv4 Server to distribute IPv4 addresses in the home network. The IPv4 packets are packed into the IPv6 data field and directly send over IPv6 to the *AFTR*. The *AFTR* receives the IPv6 packets and extracts the IPv4 packets. Now the AFTR does NAT44. In addition to the original private IPv4 address and the port, the NAT table also saves the assigned IPv6 block for the customer. Because of that, the end-user can theoretically use an arbitrary private IPv4 block[8]. Figure 6 shows that *customer 1* and *customer 2* both use the private IPv4 block *192.168.0.0/16*. There are even two personal computer with both the private IPv4 *192.168.0.12* in combination with the port *3872*. This is no problem, because the NAT table also stores the customer assigned IPv6 block. *Dualstack Lite* allows nearly endless scaling. It is not necessary to assign one public IPv4 per costumer. Over the time, the workload for the NAT table will decline because more and more clients will use the superior public, globally routable IPv6 address. There are only two cases, when IPv4 is needed. Either a legacy IPv4 application or device on the client side, or a device that wants to reach a server that is only recable over IPv4.

Dualstack lite also works with IPv4 only and IPv6 only devices. These only get assigned the corresponding IP address and work as intended. *Dualstack lite* only uses NAT44 similar to *dualstack*. That is why both are superior to *NAT444*, while providing a usable solution even with IPv4 exhaustion[18].

END-USER IMPACT

Dualstack and *dualstack lite* provide the first real solution to the IPv4 exhaustion problem. They use IPv6 addresses. IPv6 addresses enable the end-user for the first time to get public, globally routable IP addresses for all their devices. This has advantages but disadvantages too.

The introduction of IPv6 partially solves the NAT44 and NAT444 problem. Protocols like IPSec or peer-to-peer protocols function out of the box, without any server, protocol extension or additional setup. IPv6 does not provide any NAT functionality and will never need it.

Because each device has a unique IPv6 address, all ports are available for that device. Modern and future port intensive applications work without a problem.

IPv6 enables new multicast and broadcast features. Stricly speaking, there is no broadcast in IPv6. Only multicast is supported. IPv6 has a more advanced multicast compared to IPv6. *Source-specific mulicast* or short *SSM* allows only one specific source to send data. This reduces traffic. For example, an ISP could stream a TV channel to all customers via a single multicast, yet customers are not able to send messages to all others using that multicast group. *Multicast Listener Discovery version 2* or short *MLDv2* is used to dynamically manage multicast groups. User's end-devices can dynamically join and leave different groups. This is also possible with *Internet Group Management Protocol version 3* or short *IGMPv3* and IPv4 which does exaclty the same [5]. However, *IGMPv3* does not support *SSM*, but only *Multicast Group Filtering* that allows end-devices to list clients in which they are not interested .The total number of multicast groups in IPv6 is bigger. In addition *Multicast Address Scopes* are supported [12]. In total IPv6 has more advanced multicast features.

IPv6 enables end-user accessable end-to-end connectivity, because no *NAT* is used. A direct connection between two individuals can be established without a central server. This opens up better gaming multiplayer experience, because everybody can play with anyone else. Gamers do not have to care about different NAT types anymore. This lead to many connection problems with IPv4 and *NAT* with for example gaming consoles [2].

With direct end-to-end TLS communication, a trusted channel between two parties can be established with use of client-side TLS certificates. This opens up new innovation possibilities for messager-apps and much more. Until now many services work with a central server. Assuming Alice and Bob want to communicate with eachother. Alice send a message to a server C. This message is end-to-end encryted using TLS. Bob connects to C and gets the message for the server C. In case server C is mallicious, C could perform *Person-In-The-Middle* attacks. If Alice and Bob create a TLS tunnel directly with each other, they can validate their identity. *Person-In-The-Middle* attacks are not possible anymore. IPv6 enables the direct communication that is needed for that

The introduction of IPv6 to end-users with *dualstack* and *dualstack lite* has some downsides too. The main problem is regarding privacy. This is discussed later on in this paper. *Dualstack* and *dualstack lite* have a security problem that is not related with IPv6 directly. Most devices accept IPv4 and IPv6 traffic. This leads to complicated firewall rules. System administrators and especially end-users can make mistakes setting up the system. This can lead to exposed clients which are reachable over the internet even though they should remain private. That is a security risk.

All in all, *dualstack* and especially *dualstack lite* are the first good solutions for the IPv4 exhaustion problem. Apart from a more complicated and complex setup, the end-user can use new features and capabilities to explore the internet. *Dualstack* is the first step in the direction to return the internet to an open and accessible global network where everyone can host servers and access content.

4 LATEST SOLUTION: IPV6 ONLY

Unfortunately *dualstack* still requires the assignment of one or multiple IPv4 addresses per user. ISPs, especially mobile internet providers, run out of IPv4 addresses and want to transition to the new technology IPv6. This is mainly to reduce complexity and cut costs. That is why ISPs start to roll out IPv6 only connections. The end-user usually gets one */64* IPv6 block.

If the end-user only accesses IPv6 content no problems appear. In case a end-device wants to access an IPv4 only server, network-based translation is required. Another problem occurs regarding IPv4 legacy applications that run on IPv6 only end-devices. These IPv4 requests must be translated too. This happens on the end-device itself with host-based translation.

4.1 NETWORK-BASED TRANSLATION: NAT64/DNS64

NAT64 in combination with *DNS64* is a way for ISPs to deploy IPv6 only to their customers. *NAT64* only works if all end devices support IPv6 natively. This is the case for nearly all devices. Every device from the end-user gets a public, globally routable IPv6 address. If a device wants to connect to a website, the URL is translation to an IP address with the use of the *Domain Name System* or short *DNS*. *DNS* works a little bit different in this setup. The default *DNS* server is not a normal DNS server, but a special *DNS64* server. The *DNS64* server works as the following.

First the *DNS64* server forwards the request to a normal *DNS* server. This normal *DNS* server can be reached either over IPv4 or IPv6. In example Figure 7 the *DNS* server is reached over IPv4. If the *DNS* server delivers a AAAA record for the requested domain, the end-user client communicates normally over IPv6. If the *DNS* server only delivers an A record for the domain, the *DNS64* manipulates the DNS record and delivers an AAAA record with a special IPv6 address to the client. The special address points to the ISP's *NAT64* endpoint. The address of the *NAT64* server consists of two parts. The first part is a well-known standardized part to reach the *NAT64*.

0064:ff9b::/96

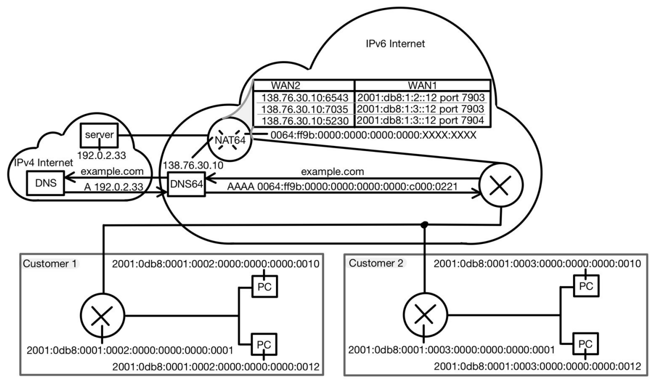

IPv6 Internet
WAN2
WAN1
138.76.30.10:6543
2001:db8:1:2::12 port 7903
138.76.30.10:7035
2001:db8:1:3::12 port 7903
138.76.30.10:5230
2001:db8:1:3::12 port 7904
NAT64
0064:ff9b:0000:0000:0000:0000:XXXX:XXXX
server
192.0.2.33
IPv4 Internet
138.76.30.10
example.com
DNS
A 192.0.2.33
DNS64
example.com
AAAA 0064:ff9b:0000:0000:0000:0000:c000:0221
Customer 1
2001:0db8:0001:0002:0000:0000:0000:0010
PC
PC
2001:0db8:0001:0002:0000:0000:0000:0001
2001:0db8:0001:0002:0000:0000:0000:0012
Customer 2
2001:0db8:0001:0003:0000:0000:0000:0010
PC
PC
2001:0db8:0001:0003:0000:0000:0000:0001
2001:0db8:0001:0003:0000:0000:0000:0012

Figure 7: Example of a NAT64/DNS64 network

The second part encodes the original requested 32-bit IPv4 address. Both parts are combined as described in RFC6052 [7]. In the example Figure 7, the server *example.com* only accepts IPv4 traffic. Thus the *DNS* server only return an A record with the public IPv4 address *192.0.2.33*. The *DNS64* manipulates that record and returns an AAAA record with the IPv6 address *0064:ff9b:0000:0000:0000:0000:c000:021*. The first part is the standardized prefix. The second parts looks a little bit different than the IPv4 address. Each of the four IPv4 address parts, that is represented by a number between 0 and 255, is represented by two hex digits now. The first block *192* is represented as a *c0* in hex. The second block *0* is represented as a *00* in hex. The third part *2* as a *02* in hex and the last part *33* as a *21* in hex[7].

A client device that wants to access an IPv4 server sends IPv4 packets according to the *DNS64* AAAA record. The ISP's *NAT64* server gets these IPv6 packets and translates them into IPv4. A NAT translation is performed. The *NAT64* server maps its public IPv4 address and a certain port to the IPv6 address plus port. This is similar to a *NAT44* translation. The IPv4 only server does not notice that the connection is between an IPv6 and IPv4 device.

It is only possible to establish an connection from an IPv6 device to and IPv4 device. This direction works, because the IPv6 address is four times longer than the IPv4 address. The IPv4 and the *NAT64* server can be encoded into one address as decrebed above. It is not possible to encode an IPv6 address into IPv4. The only possibility to start an IPv4 to IPv6 conenction is over static routes. The *NAT64* would have to assign a public IPv4 to an IPv6 exclusively. This eliminates the benefit of IPv6 and is not possible on a scale due to IPv4 exhaustion[9].

4.2 NETWORK-BASED TRANSLATION: 464XLAT

464XLAT as described in the RFC [11] is a combination of stateful and stateless translation. The ISP only provides an IPv6 connection to the customer. All ISP internal networks can be IPv6 only. Each end-device can get an private IPv4 and a public IPv6 address. It is possible that some devices only get IPv4 addresses and others only get IPv6 addresses. There are two translators involved.

The first translator is at customer-side. It is called *CLAT*. The *CLAT* performs a stateless translation, which translates a private IPv4 address into a public IPv6 address. This is a bijective one to one mapping and therefore stateless. The *CLAT* also performs routing, DNS and acts as a DHCP server. It is a usual home router.

The second translator is at ISP side. It is called *PLAT*. The *PLAT* is stateful and translates global IPv6 addresses to global IPv4 addresses. It works like a NAT64 as described above.

464XLAT only works with a typical client-server model where the server has a public, globally routable IPv4 address. Inbound IPv4 connections to the customer are not possible. There are three different ways of communication.

The first option for a device is to use it's public, globally routable IPv6 address to directly communicate with other IPv6 capable devices. No translation is required. All IPv6 features work as intended, because IPv6 is directly used.

The second option is to connect from an IPv6 end-device to an IPv4 server. This is done with a stateful translation, performed by the *PLAT* at the provider. This is equivalent to Figure 7. A *NAT64* is used.

The third option is to connect from a IPv4 end-device to a IPv4 server. This uses both, the client side *CLAT* and the ISP side *PLAT* and is the actual *464XLAT*. First the private IPv4 address of a end-device is uniquely mapped to a Public IPv6 address. This is done similar to *DNS64* but with an /64 block. The exact mapping is defined in RFC6052 [7]. The most important property is, that it is a bijective and stateless. The *PLAT* of the ISP is reached over IPv6. Then the *PLAT* translates the IPv6 address with *NAT64* to an public IPv4 address. The client using a private IPv4 address now communicates over IPv6 with another IPv4 only server. A end-user can use their home setup with private IPv4 addresses as usual. *464XLAT* translates everything to IPv6 and back to IPv4.This is desirable for the provider, because the ISP can deploy IPv6 in their entire network and only needs a *PLAT* at the edge to a connecting IPv4 network.

It is important to note that it is not possible to reach an IPv6 client from an IPv4 end-device. This is an expected behaviour and not a problem. Most IPv6 servers are dualstack and also support IPv4.

END-USER IMPACT: NETWORK-BASED TRANSLASTION

NAT64 in combination with *DNS64* or *464XLAT* has advantages for the ISPs. It enables them to completely switch to IPv6 on their internal network and still provide legacy IPv4 support for their costumers. The end-user experiences some advantage and some disadvantages with a network based translation.

The end-user profits from native IPv6 support. Once the transition is completed, neither the ISP nor the end-user needs to do

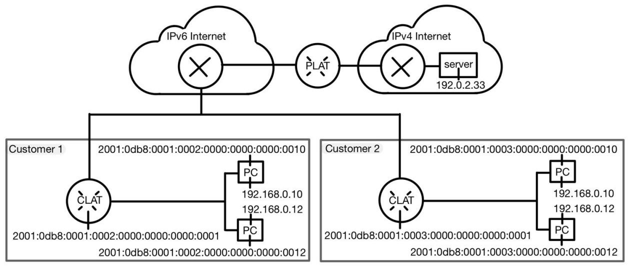

IPv6 Internet
PLAT
IPv4 Internet
server
192.0.2.33
Customer 1
2001:0db8:0001:0002:0000:0000:0000:0010
PC
CLAT
192.168.0.10
192.168.0.12
PC
2001:0db8:0001:0002:0000:0000:0000:0001
2001:0db8:0001:0002:0000:0000:0000:0012
Customer 2
2001:0db8:0001:0003:0000:0000:0000:0010
PC
CLAT
192.168.0.10
192.168.0.12
PC
2001:0db8:0001:0003:0000:0000:0000:0001
2001:0db8:0001:0003:0000:0000:0000:0012

Figure 8: Example of a 464XLAT network

much. The *DNS64* and *NAT64* servers can be deactivated or kept online for a small number of still present IPv4 servers. The end-users can keep using their network equipment. If *464XLAT* is used, the user could do an update and only use IPv6 in the future, eliminating all private IPv4 address in the network. With *NAT64*, IPv6 is already used exclusively. This simplifies the setup. It increases security too, because only one set of firewall rules needs to be set. A simpler setup is more error-resistant and thus more secure. Being future-proof is an advantage for the end-user too.

NAT64 has some disadvantages for the end-user. DNSSEC does not work, because *NAT64* manipulates the records. The idea of DNSSEC is to prohibit DNS attacks. An attacker should not be able to exchange a DNS entry with his own mallicious one and poison DNS caches. This is achieved by signing DNS entries. Even if an attacker is able to perform an DNS cache poisonig attack, a victim can verify the signature and prevent the cache poisining. *NAT64* manipulates DNS entries intentionally. The signature would be invalid. DNSSEC is not possible with *NAT64*.Moreover, it is not possible to establish a connection from an IPv4 only device to the home network of the end-user. The end-user can only host servers for IPv6 devices. This is a clear drawback, compared to *dualstack*. With *dualstack* and port-forwarding, at least one server per port can be hosted. This disadvantages is less dramatic, the more the IPv6 transition continues. I assume that most end-user try to access their home devices from a mobile device. Nearly all mobile devices support IPv6 by default.Protocols that embed the IP address at higher levels will probably not work, because a NAT is used for translation. It is desirable to complete the IPv6 transition as fast as possible to finally end the use of NAT. Translation in general also effects the performance of devices. IPv6 only users are negativly effected by translation[19]. Devices with a static IPv6 address can easily be tracked. Normally the device MAC-Address is used as a suffix for the last blocks of the IPv6 address. IPv6 can be a privacy problem. There a ideas to randomise the IPv6 address on the end-device to improve privacy. This is a problem that can be fixed.

4.3 HOST-BASED TRANSLATION: BIS

Bump-In-the-Stack or short *BIS* is a technique that allows IPv4 legacy applications on a *dualstack* device to communicate with an IPv6 host over IPv6. This only works when the legacy application wants to reach a server over a URI. The IPv4 legacy application does a DNS request for an A record. This request is intercepted. A DNS request for an AAAA record is done. The transaltor has an internal table with public IPv6 addresses from the DNS and corresponding dummy IPv4 addresses. The dummy IPv4 is either the real public IPv4 or an synthetic IPv4 that is only valid inside the host.Then a manipulated DNS A record with the dummy IPv4 is created and passed to the IPv4 legacy application. The legacy application and the IPv4 stack create a valid IPv4 packet with the local sender IPv4 and the reciever dummy IPv4. The packet is intercepted between the TCP/IP or UDP/IP module and the network card driver. The translator provides the IPv6 address that correspond to the dummy IPv4 address and creates a valid IPv6 package. The data from the IPv4 packet is inserted into the IPv6 packet. The communication happens over nativ IPv6. The response is converted back to an IPv4 packet using the translation table and passed to the legacy application. This does not work with embedded IPv4 addresses, because no AAAA record can be requested.It is sufficient to have a local IPv4 address[14] [3].

4.4 HOST-BASED TRANSLATION: BIA

Bump-In-the-API or short *BIA* is a similar technique and a improvement of *BIS*. It also only works on dualstack end-devices. The socket call is intercepted. Instead of an IPv4 socket call, an IPv6 socket is invoked. Thus it is not necessary to translate the IP header. This also does not work with hard-coded IPv4 addresses. A local IPv4 is sufficient[14] [4].

4.5 HOST-BASED TRANSLATION: BIH

Bump-In-the-Host or short *BIH* is a successor to *BIS* and *BIA*. With *BIH* it is possible to support an IPv4 only legacy application on an IPv6 only device. Thus it is possible to run these application when an IPv6 only connection is provided by the ISP. This also requires a name based connection and not a embedded IPv4 address. There are two versions of *BIH*, one version using an improved version of *BIS* and the other version using an improved version of *BIA*. The main change is that the end-device assigns itself a private IPv4 that is used for *BIS* or *BIA*. Thus a real private IPv4 and a Dualstack network is not needed.[14][10].

END-USER IMPACT: HOST-BASED TRANSLATION

Host-based translation is the last puzzle peace for an IPv6 only system. Network-based translation enables users to deploy an IPv6 only network at their home. This requires all end-devices to support IPv6. In addition to that, all applications running on these end-devices need to support IPv6 too. Most applications probably automatically support IPv6 because they are programmed in a higher level programming language. Specific legacy low-level IPv4 only applications might not support IPv6 out of the box. Host-based translation enables end-users to continue using these applications. Multicast and embedded addresses are not supported. The end-user could experience rare situations where programs will not work. Host-based translation in general only has positive effects for the end-user, because it enables nearly all IPv4 only applications to run.

5 ISP TUNNELS

This last section deals with ISP tunnels. Especially at the beginning of IPv6 deployment, IPv6 path stability was worse than IPv4 path stability[13]. That is because of the way ISPs transport IPv6 traffic. This has an impact on end-users. ISPs could build an IPv4 only and an IPv6 only infrastructure. To make the transition faster, ISPs tunnel IPv6 traffic over existing IPv4 paths. There are various protocols to achieve that. It is also possible to tunnel IPv4 over IPv6. Existing IPv4 dominance does not hinder IPv6 only investments because of that.

5.1 6in4 TUNNELING

6in4 is described in the corresponding RFC[6]. The basic concept is fairly simple. The IPv6 header, the transport layer header and the data of the IPv6 packet are inserted into the data field of the IPv4 packet. Both ends of the IPv4 tunnel need public, globally routable IPv4 addresses. At the tunnel entry, the IPv6 package is encapsuled into the IPv4 package. At the tunnel exit, the IPv6 package is extracted from the IPv4 data field. The tunnel counts as a single hop regarding IPv6, regardless of the actual IPv4 hops.

5.2 6rd RAPID DEPLOYMENT

6rd Rapid Deployment is an extension to *6in4*. It allows ISPs to limit the 6in4 tunnel to their own network. Thus they can provide access to all nodes in their network and guarantee good service.

END-USER IMPACT

Tunneling IPv6 traffic enables ISPs to rapidly provide IPv6 traffic to all customers without investing heavily in new hardware. This accelerates IPv6 adaption. With IPv6 tunneling more IPv6 paths are virtually created. Existing IPv4 paths can be used for IPv6. The end-user experiences a stable internet connection even when some IPv6 paths are not active due to technical problems. ISP tunnels have a minor effect on the end-user and will not be discussed in more detail in this paper.

6 SUMMARY

With the exhaustion of IPv4 addresses, ISPs are now forced to transition to IPv6 addresses. This has an direct impact on the end user, because both protocols are not compatible.

Before transitioning to IPv6, ISPs invested heavily in NAT systems. This caused customer end-devices to not be routable in the public internet. *NAT44* effected multiple protocols and required a new version of them. NAT traversal can be a security problem. This is why NAT should be abandoned as soon as possible. With *NAT444* even more problems appeared for the end-user.

Dualstack introduced IPv6 addresses to end-users for the first time. This enabled user to host multiple public, globally routable servers from their home. With a further scarcity of IPv4 addresses, *dualstack lite* was introduced. *Dualstack lite* outsources the NAT from the end-user to ISP level. The end-user can only profit from a *dualstack* connection. There are only added features.

Today, ISPs start to provide IPv6 only services. With *NAT64/DNS64* end-users are only provided one IPv6 block. Connections to IPv4 only servers are done over ISP site translation. *464XLAT* performs stateless translation at end-user level and stateful translation at ISP level. Both systems allow the ISP too share precious public IPv4 addresses among multiple customers. In most cases these IPv4 addresses are not needed, because the end-users use a native IPv6 connection to the servers they want to access. Directly compared to *dualstack*, IPv6 only solutions are worse for the end-user. Translation is slower than a direct connection and some features of IPv4 do not work. It is not possible to reach an IPv6 only connected home over IPv4.

With IPv6 addresses for every device, applications running on the users end-device need to support IPv6 too. *Bump-In-the-Host* is a system, to use IPv6 even with legacy application that only support IPv4. Making old applications work with IPv6 has only a positive impact for the end-user

To improve IPv6 availability, ISPs tunnel IPv6 traffic over existing IPv4 infrastructure. This has a positive effect on the path availability and thus a positive effect on the end-user experience. *6in4* is a basic protocol to establish those tunnels. *6rd* is an improved version. The end-user profits from that.

All in all, the deployment of IPv6 by ISPs benefits the end-user, at least in the long term. There might be some devices or applications not work properly at the beginning. Though, once the transition is completed, end-users profit. IPv6 enables all devices to initiate true end-to-end communication. End-users are able to host multiple servers and participate in peer-to-peer networks. In the short term, the end-user can experience some unpleasent things. In case NAT444 is used, innocent user get punished for misbehavior of other user in form of rate-limiting or total blocking. If the ISP only supports IPv6, some applications or services might not work as expected, because of a lack of nativ support. Though the end-user can profit from IPv6 in the short term. For example, online multiplayer games work more seamlessly with IPv6, because no NAT is involved. To answer the title of this paper in one sentence:

The end-user impact may be slighly negativ in the short term, though the end-user will benefit in the long-term.

REFERENCES

[1] Google ipv6 statistics. https://www.google.com/intl/en/ipv6/statistics.html. Accessed: 2022-12-29.
[2] Troubleshoot nat errors and multiplayer game issues. https://support.xbox.com/en-US/help/hardware-network/connect-network/xbox-one-nat-error/. Accessed: 2022-12-29.
[3] Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS). RFC 2767, RFC Editor, February 2000.
[4] Dual Stack Hosts Using "Bump-in-the-API" (BIA). RFC 3338, RFC Editor, October 2002.
[5] Internet Group Management Protocol, Version 3. RFC 3376, RFC Editor, October 2002.
[6] Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213, RFC Editor, October 2005.
[7] IPv6 Addressing of IPv4/IPv6 Translators. RFC 6052, RFC Editor, October 2010.
[8] Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion. RFC 6333, RFC Editor, August 2011.
[9] Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146, RFC Editor, April 2011.
[10] Dual-Stack Hosts Using "Bump-in-the-Host" (BIH). RFC 6535, RFC Editor, February 2012.
[11] 464XLAT: Combination of Stateful and Stateless Translation. RFC 6877, RFC Editor, April 2013.
[12] IPv6 Multicast Address Scopes. RFC 7346, RFC Editor, August 2014.

[13] Forough Golkar, Thomas Dreibholz, and Amund Kvalbein. Measuring and comparing internet path stability in ipv4 and ipv6. In *2014 International Conference and Workshop on the Network of the Future (NOF)*, pages 1–5, 2014.

[14] Ala Hamarsheh and Yazan AbdAlaziz. Transition to ipv6 protocol, where we are? In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6, 2019.

[15] Lawrence E. Hughes. *The Depletion of the IPv4 Address Space*, pages 119–146. Apress, Berkeley, CA, 2022.

[16] Lawrence E. Hughes. *The Future of Messaging with No NAT*, pages 355–377. Apress, Berkeley, CA, 2022.

[17] Lawrence E. Hughes. *IPv6 Deployment Progress*, pages 147–156. Apress, Berkeley, CA, 2022.

[18] Lawrence E. Hughes. *Transition Mechanisms*, pages 285–326. Apress, Berkeley, CA, 2022.

[19] Mehdi Nikkhah, Roch Guerin, and Mehdi Nikkhah. Migrating the internet to ipv6: An exploration of the when and why. *IEEE/ACM Trans. Netw.*, 24(4):2291–2304, aug 2016.

[20] Philipp Richter, Florian Wohlfart, Narseo Vallina-Rodriguez, Mark Allman, Randy Bush, Anja Feldmann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. A multi-perspective analysis of carrier-grade nat deployment. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 215–229, New York, NY, USA, 2016. Association for Computing Machinery.