Migrating to IPv6 with Address+Port translation

Florent Fourcot and Bertrand Grelot, *Students in Telecom Bretagne, France*

***Abstract*—As the IPv4 address pool is being exhausted, it becomes urgent to find a way to migrate IPv4 network architectures to IPv6. The current report will discuss a strategy known as "Address + Port" translation, from the first studies to the final experiment and the installation of the real system.**

CONTENTS

I. INTRODUCTION

AS everybody might know, the IANA unallocated address pool will soon be exhausted. At the time we write this document, the estimated day of the exhaustion is September 30th, 2011. And as everybody might know, we should soon migrate to IPv6. The issue is to be able to find a migration process to allow the Internet Providers to migrate smoothly. Different strategies are now being studied; we will present a project of Address + Port translation.

It consists in assigning to several users the same IPv4 address, the users being differenciated by the range of ports they use. An IPv6 tunnel is created via the box, allowing the Internet Provider to transfer data to the user in IPv6 transparently.

The issue with this system is to find the proper number of the ports range to be assigned to each user. This document will present the results of a survey done in a students residence (counting about 150 users) and the study of the number of ports used by several web applications or softwares. Then we will present the Address + Port translation system and the results of the experiment.

March 15, 2010

II. STUDY OF THE NUMBER OF PORTS USED

THE A+P project is based on the use of a unique IPv4 address for several machines. The distinction between these different machines is made by ranges of ports. As one can see, the maximum number of devices connected to the A+P system depends on the width of the port range.

A. Survey context

A computer running a Linux-based distribution[1] has been used in the study and has been connected to the Internet *via* a IPv4 & IPv6 link. Every ten seconds, a script did connected to the computer and collected port counts, using `/proc/net/nf_conntrack`. Then, we used a spreadsheet program to build a table (see table I).

From this table, statistics on the use of ports can be calculated (as in table II).

TABLE I
BASIC EXAMPLE OF IPV4 PORTS COUNTING

Date	Ports	DNS	HTTP	HTTPS
1267551614	5	0	1	4
1267551625	14	8	1	4
1267551636	47	40	2	4
1267551647	47	40	2	4

[1]Ubuntu Karmic, on a 2.6.31-19 Linux kernel

TABLE II
BASIC EXAMPLE OF STATISTICS

v4 ports average	20,7	
DNS average	15,6	75,36%
HTTP average	2,1	10,14%
HTTPS average	2,35	11,35%
Other v4	0,65	3,14%

These results let us present statistics per application and establish different profiles of use. Among the applications, we will present facts and figures for:

- base consumption: none program executed on the test system, apart from the internal services (NTP, updates search, etc.);
- Facebook: a web browser[2] on the home page first, and then on a few pages;
- Gmail: same context, but only on the Inbox page;
- Google Maps: first the display of a plan, then a satellite view;
- Google Wave: web surfing on different features of the Wave project;
- Pidgin: IM software used with a few accounts (MSN, Gtalk, XMPP, Yahoo & ICQ);
- Skype: IM software with a voice-based call;
- Torrent: ISO downloading through a BitTorrent client[3];
- Twitter: web surfing on a Twitter page;
- Youtube: web surfing on video pages.

This will be the first part, presenting results per application.

We in Rennes, France, live in a student residence (called "MaisEl", counting about a hundred students). There is another campus in Brest, with a few more students (around 500). Each location contains an Internet gateway we used to collect statistics on ports consumption.

B. Results per application

By default, the test system appears to have a few ports connected to the Internet (fig. 1), such as update services. As shown on the chart, at the end of the test, only one port was occupied, and this base usage is the reference for all the following tests. Each flow represents the IPv4 port consumption of a website (each test lasted between 10 and 30 seconds).

1) Websites: on the local network, a transparent proxy has been setup, letting us study the most visited websites. It appears than 2.0 web is really present, so let's begin with Facebook (fig. 2) and Twitter (website version, fig. 3). On Facebook, the test was to reach the home page and then to open two pages including media content (e.g. a photo album). The number of DNS requests is really impressive (up to 90 requests at a time), and at the end of the test, 46 HTTP ports were opened. Same observations can be done on Twitter (up to 109 DNS requests and 32 HTTP ports). To be up-to-date, such websites reload continually. In the case of Facebook, Firebug shows that a great number of domains are used for media content, each time with a complete URL. That explains the numbre of DNS calls.

[2]Firefox 3.5.7 for Ubuntu Karmic
[3]Transmission 1.75

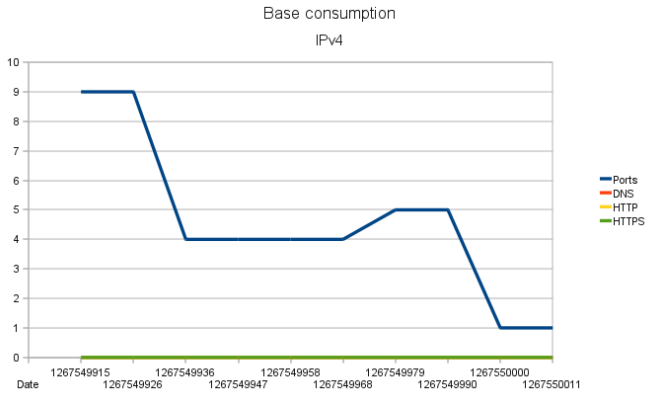

Fig. 1. Base consumption (IPv4)

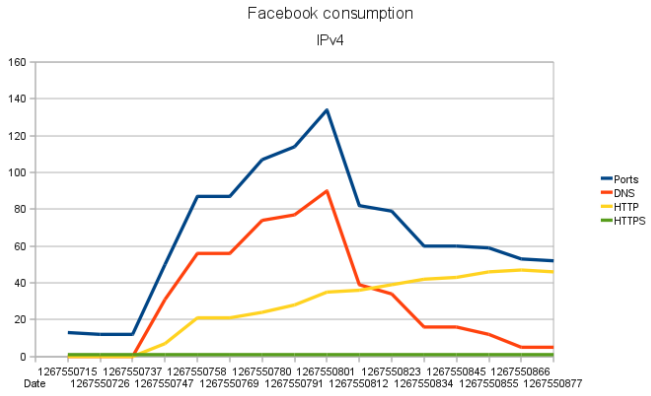

Fig. 2. Facebook consumption (IPv4)

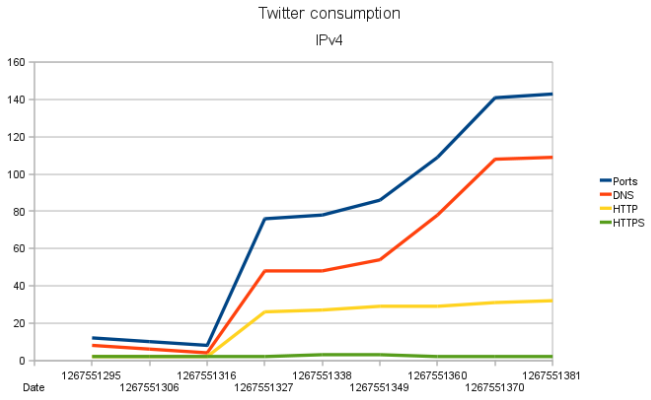

Fig. 3. Twitter consumption (IPv4)

2) Hybrid IPv4/IPv6 websites: Google & consorts provide IPv6 websites, so it can be interesting to look at the distribution between IPv4 & IPv6. Tests have been made on

Gmail (fig. 4, 5, 6) and Google Maps (fig. 7, 8, 9). The same behaviour is observed on other Google sites (like Wave or Youtube).
In both cases, more than $2/3$ of the traffic is IPv6. It is to be noted that IPv4 is mainly used for DNS requests while IPv6 transports contents. With Google Maps, a huge number of connections is opened simultaneously. In this case, reducing the length of the ports range would have implications on the download rate.

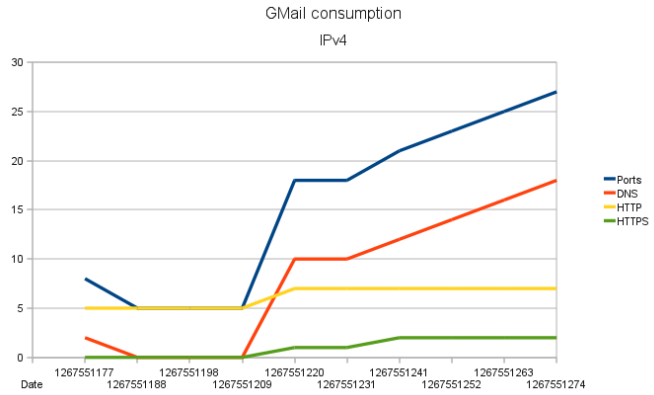

Fig. 4. Gmail consumption (IPv4)

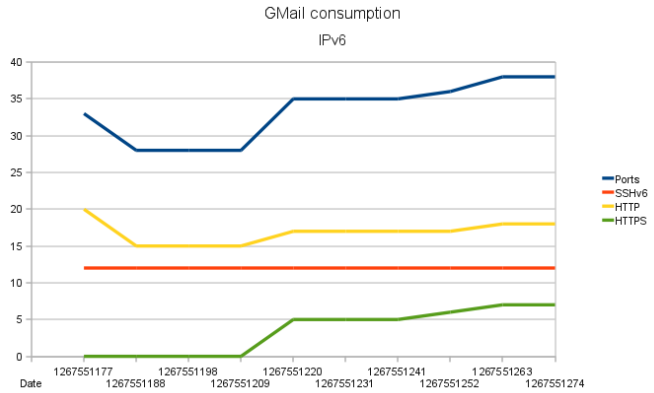

Fig. 5. Gmail consumption (IPv6)

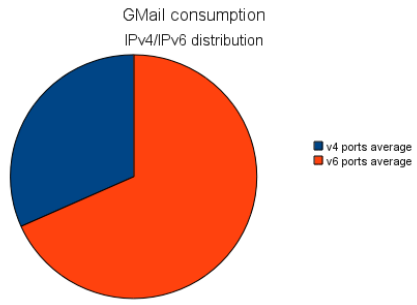

Fig. 6. Gmail IPv4/IPv6 distribution

3) Instant messaging: instant messaging softwares packages are also used on the local network. Pidgin has been used to test a set of protocols:

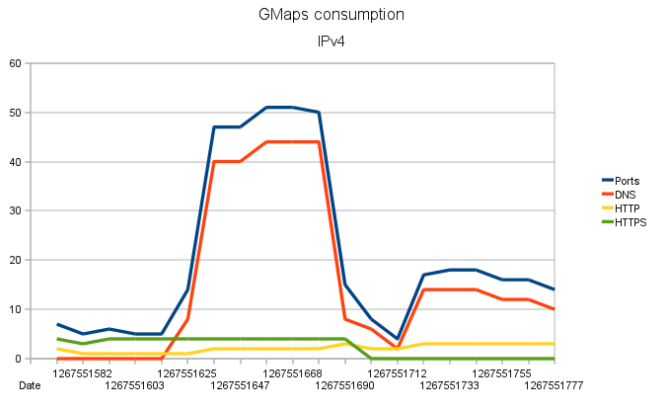

Fig. 7. Google Maps consumption (IPv4)

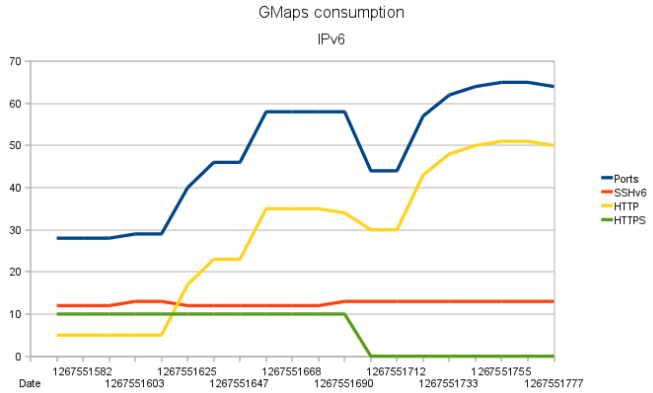

Fig. 8. Google Maps consumption (IPv6)

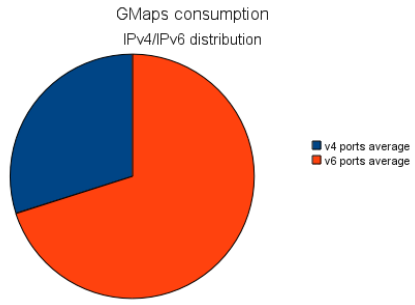

Fig. 9. Google Maps IPv4/IPv6 distribution

- MSN (text-only)
- XMPP (Gtalk, text-only)
- XMPP (external server, text-only)
- ICQ (text-only)

Each service uses a different port (even if MSN is designed to be able to work through the port 80). In this test (fig. 10), all the services logged on at the same time, and a text conversation has been initiated (on the XMPP-GTalk account). One can observe up to 30 DNS requests at a time and a great amount of ports used at a point in time (this is the time when the software gets the contact list, with metadata and optional pictures).

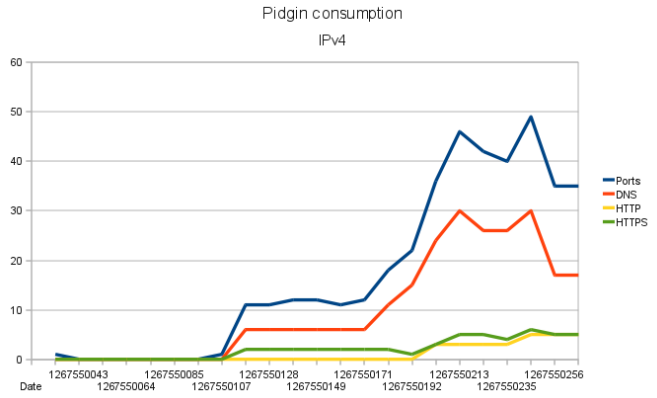

Fig. 10. Pidgin consumption (IPv4)

Skype (fig. 11) is a peer-to-peer VoIP client developed in 2003. It is often used by people to contact family or friends because the communication is free. In the p2p Skype system, we can find three types of nodes [7]:

- end hosts
- super nodes
- login server

Each host is connected to a super node and, at the start of a session, to the login server. The connection test has been made on a non port-restricted NAT (login + voice call). Different transfers are made by the Skype client:

- startup and search of a new version
- first-time login
- bootstrap super nodes
- call establishment
- ...

On the chart, we can see that very few DNS requests are made by the Skype client (fewer than 10), and a few HTTP/HTTPS are used (fewer than 5 each – HTTP is used for instance during the search of a new version or during the login process).

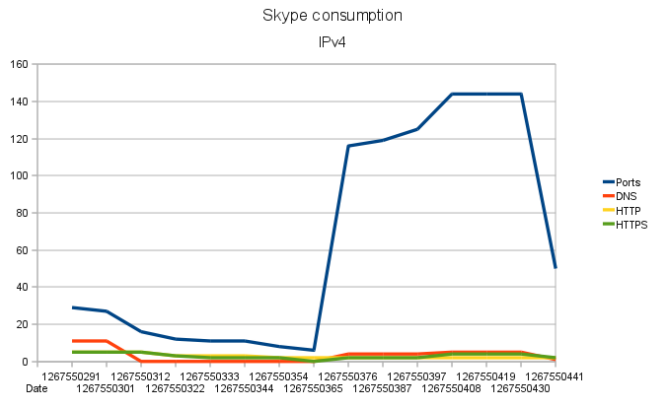

Fig. 11. Skype consumption (IPv4)

4) Peer-to-peer file transfer: the last test is a BitTorrent file transfer. It is well known that the BitTorrent protocol is very greedy as shown on fig. 12. The file download has two stages:

- download + file sharing (upload)
- upload only

This is the only test which lasted more than 30 seconds (41 minutes and 20 seconds). More than 400 ports were opened as a time. After a study of the addresses used, we can note that very few hosts are used more than one time (fewer than 10, using between 2 and 4 ports at the same time).

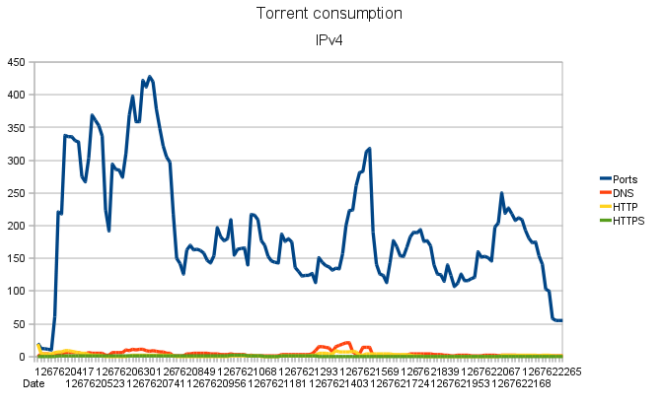

Fig. 12. Torrent consumption (IPv4)

C. Gaming: the PS3 case

The previous results do not cover the whole studied population. The case of gamers remains. The fig. 13 to 17 represent different use cases of Sony's Playstation 3, which has been designed to provide network games and Internet surfing through a WiFi or an Ethernet connection.

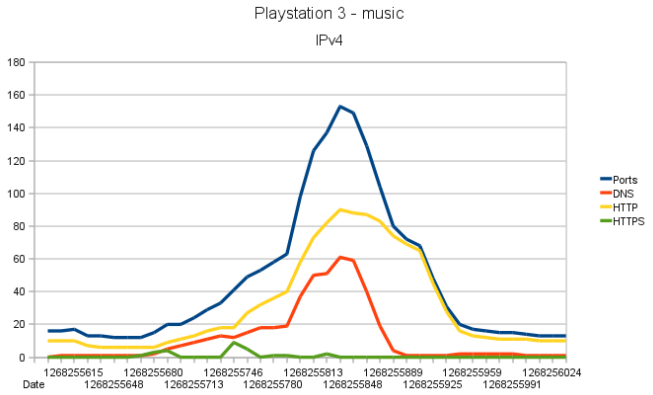

Fig. 13. Playstation 3 - music streaming (IPv4)

Sony's Playstation 3 provides a multimedia streaming application – *Vidzone* – (fig. 13). This is transparent for the user, but *Vidzone* uses HTTP to download data. During the navigation stage through the menus, Vidzone has the same consumption as a small website with not so much ad serving. The biggest consumption is seen when the music is downloaded: it is probably load-balanced on a lot of web servers. When the music title is completely downloaded, the application does not more do anything.

WipEout HD (fig. 14) is a futurist racing game. The test was a multiplayer racing game on the Internet. The start of

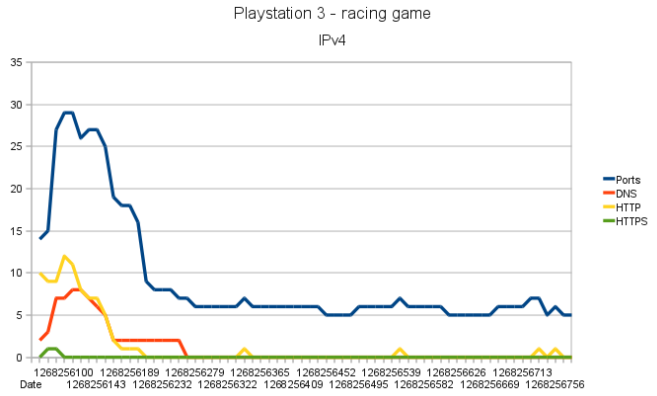

Fig. 14. Playstation3 - racing game (IPv4)

this chart is the search of a race. When the race starts, the game uses only 6 ports. This is really not much: while playing, a gamer uses fewer ports than a typical web surfer. Then the connection is very stable, as long as the player does not search for a game nor a team of players.

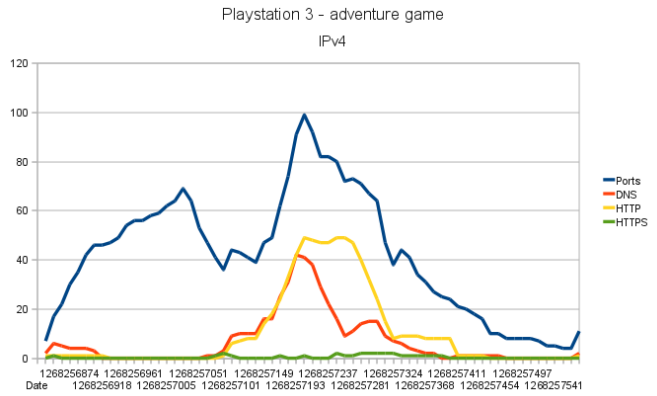

Fig. 15. Playstation3 - adventure game (IPv4)

LittleBigPlanet is an online but solo player. The connectivity is very important because the community builds all the maps of the game. The connection begins with the navigation through the menus, with a lot of parallel connections – probably on several servers (there are no DNS requests, of course the server's IP address can be hard-coded). When the player chooses a map, there is a download stage through HTTP protocol. Then the game does not consume any ports, all data are cached in the local Playstation memory.

To test online shooting games, *Call of Duty* seemed to be a good test. But after the connection stage, the game did not work because it refused the no-fragment flag on IPv4 datagrams, causing the connection to be stalled. It is not a protocol issue, it comes from the PRR which does not support fragmentation yet. So the test was made with *Killzone 2* (fig. 16) instead. The connection begins on the server, and the search for a game. The connection becomes stable when the game itself starts. Like *WipeOut*, the game is very stable, regarding the port consumption.

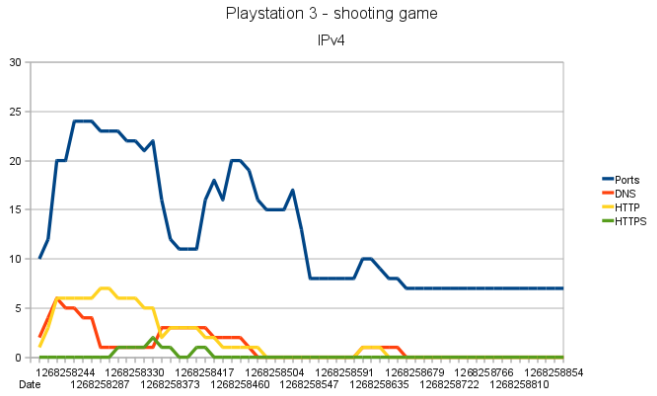

Fig. 16. Playstation3 - shooting game (IPv4)

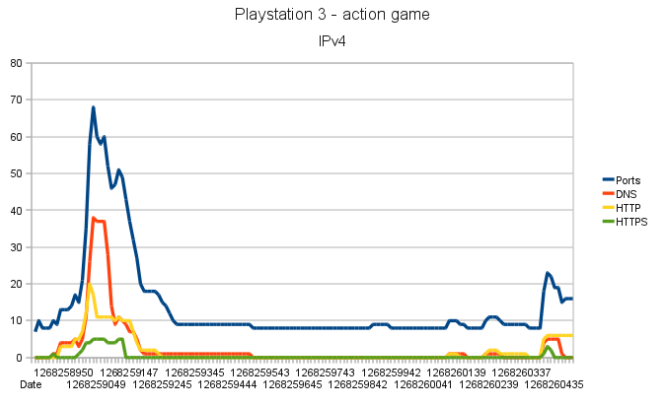

Fig. 17. Playstation3 - action game (IPv4)

Uncharted 2: Among Thieves (fig. 17) is an action & shooting game. This is one of the most technically advanced games produced by Sony. Like *WipeOut HD*, it starts with a search for a multiplayer group of players. The real game does not use a lot of ports.

Actually, gaming does not use a lot of ports. It is a very quiet and stable activity and an IPv4 address can be shared whitout any issue. The search of multiplayer groups uses more ports, we can always note the same behaviour, but it remains very reasonable.

The network configuration of Sony's PS3 is very easy and is built to work with all kinds of networks (NATed, not NATed, with UpNp support or not). A shared IP address should not be an issue. We observe in multimedia applications and in *LittleBIgPlanet* the same trend: the HTTP protocol is everywhere when we need to download from a server. It is easy to be intercepted with a proxy server to reduce the consumption of the client.

D. Impact of IPv6 connectivity on a wider population

1) Context of the study: we examinated the global behaviour of a group of users in a students residence. We have analysed two campuses, one with about 100 computers connected (Rennes) during the day, and the other with 400

computers (Brest). We had more freedom to change everything for the study in Rennes (e.g. the network core or the Internet gateway) and our results are more relevant in there.

The connectivity is limited on the two campuses. IPv6 and IPv4 are allowed, UDP is dropped. Users are not allowed to send any DNS request outside, they have to use the internal resolver. Every user is authorized to download 1,5GB per day, not more. All users share the same NATed IP address. All TCP ports are allowed and there is no applicative filter. A proxy server can be used for HTTP and HTTPS requests.

2) Results in Brest: we could not administrate the IPv6 router and we could not administrate the proxy server. But we have statistics on the IPv4 Internet gateway. Users do not use more than 6000 connections at a time, it is not much (fig. 18). Despite the proxy, there is a lot of HTTP connections (users have to configure the proxy manually) (fig. 19).

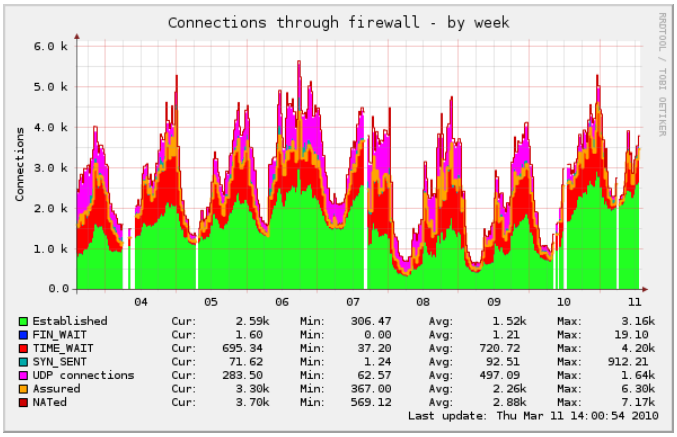

Fig. 18. A week of IPv4 connections in Brest

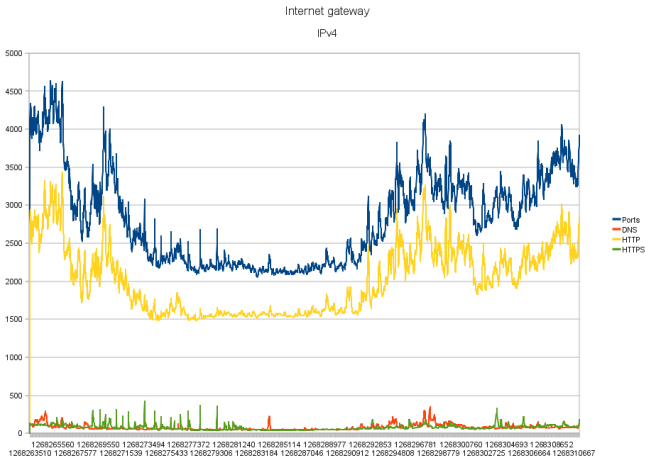

Fig. 19. HTTP the most popular protocol

DNS and UDP connections are not relevant. They represent connections from outside to a DNS master of a zone. Our conclusion is that users do not do a lot of non-HTTP(S) transfers, HTTP(S) represents in average 72% of the whole ports consumption. And a pool of 1000 ports for a computer seem to be very acceptable and confortable for users.

3) Results in Rennes: the results in Rennes are very similar to Brest ones (not surprisingly, this is the same population with the same rules). We have in Rennes all HTTP (with the proxy) connections and we stay with fewer than 4000 ports (fig. 20). HTTP represents in average 69% of the whole IPv4 ports consumption, the optional proxy seems to not be relevant in Brest.

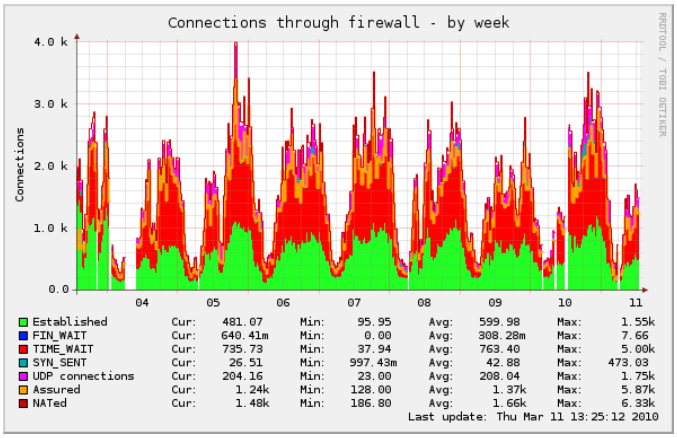

Fig. 20. A week of IPv4 connections in Rennes

4) IPv6 development: at the beginning of the project, the IPv6 use was very low. The consumers were power users who had IPv6 connectivity on their own server.

But we began to receive AAAA records from Google DNS. Like in the IPv4 case, the most important consumption of ports is the HTTP(S) (more than 90% of IPv6 ports are HTTP(S)). And Google (with all services, like Gmail and Youtube[4]) is very much used.

In the MaisEl we forced users to use IPv6 for HTTP with a transparent proxy. When a user asks for an IPv6 compliant website with a IPv4 request, we intercept the request and we transmit it in IPv6 through the proxy. This does not change anything for the users.

In a day, we use less than 200 ports 21. In the same day, the maximal IPv4 consumption is 4000 ports. The IPv6 use is really growing, and Google with very popular website helps that. But it stays very quiet. The IPv6 use is not more than 5% of IPv4, even through we have a LAN with very good IPv6 facilities. Other protocols like games, Instant Messaging and VoIP are not yet significant in IPv6.

III. The A+P system

EVERY major Internet Provider is aware of the exhaustion of public IPv4 address. The thing is, what can be the best solution to guarantee a soft migration to IPv6? Different approaches are suggested, we'll detail the Port Range Architecture, as described a draft by M. Boucadair: *IPv4 Connectivity Access in the Context of IPv4 Address Exhaustion: Port Range based IP Architecture*.

[4]Google services available currently include Google search (including image search, blog search and code search), Alerts, Docs, Finance, Gmail, Health, iGoogle, News, Reader, Picasa, Maps, and YouTube. [14]

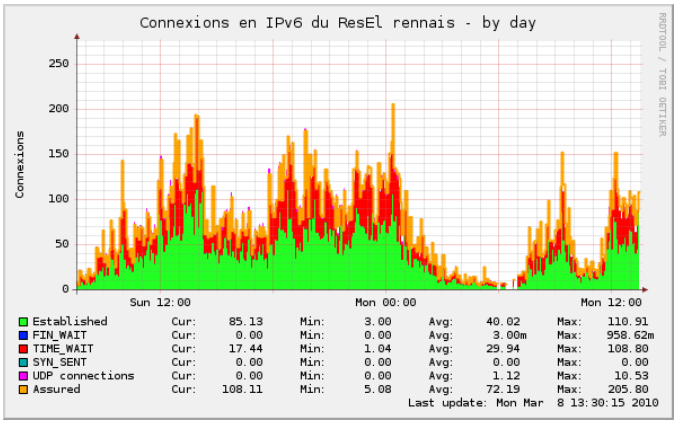

Fig. 21. A day of IPv6 connections in Rennes

A. The IETF model

The Port Range solution consists in assigning the same IP adress to several end-users' devices [4]. Each device is allocated a port range. The inbound communications are managed by a PRR[5], while the outbound communcations are managed by a device which controls the source port number from the port range. A SMAP[6] is able to provide two functions:

- encapsulation of an IPv4 packet on its way to a shared IPv4 address, in an IPv6 packet[7]
- extraction of an IPv4 packet from an incoming v6 packet.

M. Boucadair's draft on Port Range model suggests two implementation modes:

- using a SMAP to route the incoming traffic only
- using SMAPs on both end devices, with no IPv4 capabilities between them.

In the current project, we used the second implementation mode.

The main goal of such a solution is to keep IPv4-based services and add IPv6 services. It must be transparent to the end-users and must be easy to use and to implement (especially with a minor impact on routing and addressing architectures). There are known drawbacks (section III-D) detailed below.

B. Other examples of IPv6 soft migration solutions

In January, 2010, Comcast declared: *"Many experts believe that this transition could be disruptive for Internet users, so the trials we plan to conduct in 2010 will help us identify and solve any areas of difficulty involved in the transition to IPv6. Well also use this trial to determine what approach will be the easiest and most seamless for our customers. Comcast will continue to share what we learn with the Internet community, particularly with the IETF, for the benefit of other users of the Internet."* [6]

Three different main approaches are considered:

- 6RD (similar to 6to4, the difference being that 6RD only tunnels packets on IPv4-only parts of the provider's network)
- native IPv6 (creating a dual-stack architecture)
- encapsulation of IPv4 in IPv6 to get across IPv6-only parts of the provider's network (DS-Lite technique)

A fourth trial concerns only the business class customers.

C. The architecture of the experiment

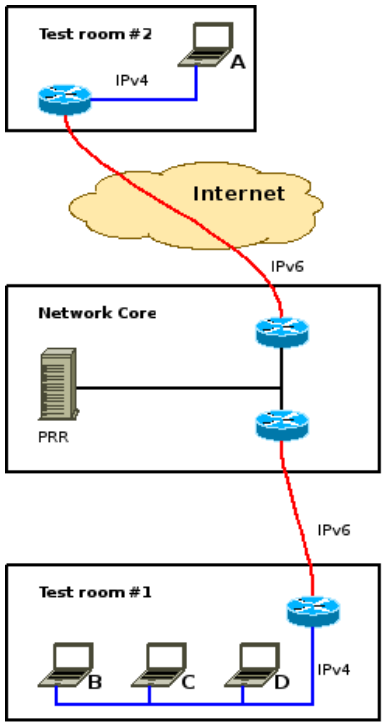

Fig. 22. Global architecture

In the fig. 22, test room #1's router and test room #2's router have IPv6 connectivity with the network core and they share the same IPv4 public address. The network map is listed on table III. The two routers are Linksys WRT54GL (native WiFi access points) modified to provide a 2.6 Linux kernel (with conntrack support). These two routers play the role of the SMAPs. They have been dispatched to two separated sites and are connected to a research lab (with the PRR) through an IPv6 connectivity. Test room #1 is connected with the PRR on the same LAN (two different VLANs with an IPv6 router in the middle), and test room #2 is linked to the PRR through a SixXS tunnel through the Internet.

1) PRR configuration: The PRR server is a C program which receives IPv6 datagrams from SMAP clients. The protocol of IPv6 datagram is a strict IPv4 in IPv6 encapsulation (protocol number 4, cf. RFC2003 [12]), the SMAP does not modify the packet. When the kernel receives the datagram, it is sent to the PRR via *libipq*. This is an old development library in userspace designed for packet processing. The PRR decapsulates the packet and send the IPv4 datagram to the final destination.

[5]Port Range Router
[6]Stateless A+P Mapping Gateway
[7]The IPv6 address is built from a v6 prefix, the IPv4 shared address and the port range used on a device.

TABLE III
NETWORK MAP OF THE EXPERIMENT

PRR • 192.108.119.26 • 2001:660:7301:d002:250:baff:febe:711
Test room #1 router (public) • 192.108.119.27 (tunnel) • 2001:660:7301:4666::1
Test room #1 router (private) • 192.168.1.1
Test room #2 router (public) • 192.108.119.27 (tunnel) • 2001:660:7301:4666::2
Test room #2 router (private) • 192.168.2.1
A • 192.168.2.121
B • 192.168.1.142
C • 192.168.1.143
D • 192.168.1.144

When the kernel receives an IPv4 datagram, this is the same procedure with *libipq*. The PRR reads the IP packet and the TCP/UDP header. If the destination IP is the shared IP, the IP datagram is sent according to the destination port. If the destination port isn't in a Port Range of a client, it is dropped. The IPv4 datagram is encapsulated in an IPv6 packet (with protocol number 4 of IPv6) and sent to the SMAP.

The PRR configuration file is very simple, and provides the list of the SMAPs (IPv6 & their port range).

```
PrrAddrV4 192.108.119. ...
PrrAddrV6  2001:660:7301: ...
IPv4Shared 192.108.119. ...
Cpe1AddrV6 2a01:240:fe55: ...
Cpe1PortRange 10000-11000
Cpe2AddrV6 2001:660:7301: ...
Cpe2PortRange 11001-12000
```

This configuration file is built dynamically[8] by a script on the routers that knows its own IPv6 address and can send it to the PRR. Once the configuration file is up to date, the PRR is restarted.

2) Configuration of the routers: We modified the Linksys configuration to have a *plug-and-play* router. At the beginning, the Linksys sends a Router Solicitation to get an IPv6 prefix and a default route. We configured the tunnel with this information and we send by SSH the new information to the PRR. We have generated a ssh-key to have a completely automatic script[9].

When it does not work (e.g. in an IPv6 network without Router Advertisements), it is possible to configure the Linksys from a client. A SSH server is listening on the IP 192.168.2.1 and port 22. The script is runnable with arguments like the local IPv6 and IPv6 default route.

By default, the MTU is configured to 1400 bytes. It will work on most of the configurations, but not everywhere. It can be changed in the script and in the configuration file `/etc/config/dhcp`.

3) End user configuration: The end user can own two kinds of equipments:

- IPv4-only device, which is entirely connected to the Internet through the SMAP
- IPv6-compatible device, which uses IPv4 through the SMAP and IPv6 directly (the SMAP is IPv6 compatible and represents the next hop).

The end user has nothing to configure, because the router provides a DHCP server (IPv4, DNS, MTU) and sends routing advertisment (IPv6 prefix & routing).

D. Known limits

While deploying the solution, Internet Providers will encounter some issues concerning the structure of the sytem. The first issue we had is that an end user cannot easily run a server with a private address (along with the difficulty to configure port forwarding policies and to enable inbound access: opened ports have to be within the allowed range). The Port Range draft suggests some other issues, like the need to activate a second ALG[10] in the core network (for instance to support SIP protocol). It also points out the fact that there may be interferences between the service and network layers.

A draft has been written on the behaviour of BitTorrent in an IP shared environment, where two limitations have been found. The first limit occurs when two clients sharing the same IP address want to simultaneously retrieve the same file located in a single remote peer. The second one is the case when a client wants a file provided by different seeders who share the same IP. In both cases, the workaround is to configure the clients with the `bt.allow_same_ip == TRUE` parameter. [5]

IV. EXPERIMENT

AN experiment has been led with two SMAPs and one PRR. We have managed to do some tests, the current part will present some of the results we got and expose some of the encountered bugs or limits of the system.

[8] cf. Appendix C

[9] cf. Appendix C

[10] Application Level Gateway

A. Context

Tests have been made from January to March, 2010, on the two routers. In test room #1, between one and three computers were connected to the system, and different OSes have been used (Ubuntu karmic, Debian Testing, Windows XP & Windows Seven). In test room#2, a Debian stable and one Sony's Playstation 3 were connected to the router. On these systems, unit tests have been conducted, and after a while, we have had a daily use of the solution, without modifying any of our habits.

B. Results

One of the first tests was to change the length of the ports range. A professional use (web surfing, mail fetching) would require 200 ports. With 10 ports only, it becomes really difficult. 1000 ports (the default value in the experiment) is really comfortable and the architecture is almost transparent for a typical user.

Our most important issue concerned MTU and it was very difficult to deal with it. We did not always have access to the router between the Linksys and the PRR, and `tracepath` did not work fine (we had an asymetric MTU, it was a little bit confusing and hard to find). The IPv4 within IPv6 tunnel has an overhead of IPv6 header (40 bytes). The table IV details the MTU values used during the project. Test Room #1 was directly connected to a native IPv6 network (Renater, French scientific network). Test Room #2 was connected with SixXS tunnel (IPv6 in IPv4). The endpoint of the tunnel did not allow packets bigger than 1280.

TABLE IV
MTU TABLE

	Native MTU IPv6	Path MTU to PRR	MTU IPv4 for clients
Room #1	1500	1460	1420
Room #2	1280	1280	1240

Our goal was to provide IPv4 and IPv6 connectivity to Linksys clients. It was easy in Test Room #1, Router Advertisements were sent on the LAN with Radvd and the Linksys forwarded IPv6 packets to the next hop. But it did not work for Test Room #2 (IPv6 does not allow a MTU smaller than 1280). We cannot configure an interface with two MTUs, one for IPv6 and one for IPv4 (Linksys client has a MTU for IPv6 equal to the native MTU, they do not need to use the tunnel), so we couln't use IPv6 in Test Room #2. This does not impact our tests, Playstation Network does not use IPv6. To configure the MTU value on the client, we used the option number 26 of DHCP protocol (cf. RFC 2132 [9]).

For a gamer, the latency was acceptable. We had an overhead of about 50 ms and it is not enough to be a problem. We were not limited by the bandwidth, the PRR being connected on a 100Mb/s Internet link.

C. Bug report

We found some issues during the tests. They are not critical, but should be fixed in the future. We described more detailed bugs in section IV-D. This enumeration is only for the record:

- we cannot send ICMPv4 from a linksys or from a client.
- FTP does not work from a client.
- some applications like *Call of Duty* do not work (fragmentation issues).
- our Openwrt version on the Linksys has some problems with IPv6 interface setups on boot. Fixed in Kamikaze 8.09.2. Our configuration init scripts are a workaround.

D. Guidelines for the after-project

Our experiments on the SMAPs are a success. But there are quite a few possible improvements. The first one is not directly related to the protocol implementation, but our Linksys does not support WiFi connectivity. An OpenWRT distribution was installed and the driver for our Wireless card did not work. But on March 4th, 2010, the OpenWrt Team has announced a beta of the next major release, codenamed *Backfire*. So the router used in the experiment (Linksys WRT54GL) should support wireless functions while running a 2.6 kernel [8].

The second one is the MTU issue. It was very frustrating during the project to change manually the MTU, the Linksys being not able to discover the path MTU with the PRR. The Linksys and PRR should listen to ICMPv6 to dynamically discover the path MTU of the link.

The third one is not a configuration issue, but depends on the capabilities of the PRR server. We had problems during the test with poorly coded application, like *Call of Duty*. They do not listen to the ICMP message *Need to Frag* and the client cannot receive messages.

We did not solve the FTP connections issues, and it should be interesting to know what the issue exactly is. FTP is still often used for file transfers, and more generally that means that we have a problem somewhere.

It is very important to test the system with more than two clients. But we did not have enough equipment to test it, and the PRR does not support more than two clients as we are writing this document. But it can improve the experiment and maybe we could find some new issues. We think that it can be very interesting to dynamically allocate a range of ports to clients. Another way to test is to build a network of virtual machines. We now know the user experience, and it is more scalable for bigger tests to use virtual servers, and not 100 configured Linksys.

Last but not least, web surfing on the Internet is possible without ICMPv4, but it is quite better with it. The system is stateless so we cannot build a context like a NAT does. But maybe there is another possible solution we did not think about yet.

V. CONCLUSION

KEEPING a IPv4 address plan on the end user's network guarantees that the system will work with non-IPv6-compliant operating systems. The whole system remains transparent for the users, and the Internet provider can migrate freely to a IPv6 network core. When the IPv6 network core is ready, two address plans can be developed, so that IPv4 can

get along with IPv6. As soon as the end user supports IPv6 on every equipment he has, the Internet provider is free to stop the IPv4 tunneling and can guarantee a full IPv6 connectivity. With the A+P solution, the issue of scaling the ports range remains. That would eventually depend on the use of each user and on the software packages used. For instance, in a small business a professional use of the Internet would require fewer than 500 ports per user, which makes it possible to have more than 120 computers on a unique public IPv4 address.
There is no geographical need, so the same public IPv4 address can be used by a computer kilometers away from another one. The solution is quite simple to develop and is affordable for a basic Internet provider as well as a network administrator of a small business who wants to migrate a local network without changing the habits of the employees. So that is an excellent starting point for a IPv6 smooth transition.

APPENDIX A
ASSIGNMENT OF PORTS

By default, a NAT does not change a source port. It is only modified when a conflict occurs :

- same source port;
- same destination address;
- same destination port.

TABLE V
NAT PORT MODIFICATION EXPERIMENT

tcp 6 5 CLOSE src=172.23.42.42 dst=217.70.191.14 sport=4000 dport=22 packets=2 bytes=84 src=217.70.191.14 dst=192.44.77.81 sport=22 dport=4000 packets=1 bytes=44 mark=0 use=1
tcp 6 8 CLOSE src=172.23.203.112 dst=217.70.191.14 sport=4000 dport=443 packets=2 bytes=84 src=217.70.191.14 dst=192.44.77.81 sport=443 dport=4000 packets=1 bytes=44 mark=0 use=1
tcp 6 8 CLOSE src=172.23.203.112 dst=217.70.191.14 sport=4000 dport=22 packets=2 bytes=84 src=217.70.191.14 dst=192.44.77.81 sport=22 dport=1517 packets=1 bytes=44 mark=0 use=1

As shown on table V, the three connections have the same port. The first one keeps port 4000 to port 22. The second one keeps port 4000 to port 443. The third one has a modified source port because of a conflict with the first connection.

If the port appears to be modified, the change is not made randomly, there are three classes of ports[11]:

- ports < 512;
- ports between 512 & 1024;
- ports > 1024.

A port can only be changed with a port in the same class.

APPENDIX B
RFC793: CONNECTION STATES

" *A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED. CLOSED is fictional because it represents the state when there is no TCB, and therefore, no connection. Briefly the meanings of the states are:*

- *LISTEN - represents waiting for a connection request from any remote TCP and port.*
- *SYN-SENT - represents waiting for a matching connection request after having sent a connection request.*
- *SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.*
- *ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.*
- *FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.*
- *FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.*
- *CLOSE-WAIT - represents waiting for a connection termination request from the local user.*
- *CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.*
- *LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).*
- *TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.*
- *CLOSED - represents no connection state at all.*"

[11] cf. `/net/ipv4/netfilter/nf_nat_proto_common.c` (Linux 2.6.30 kernel)

APPENDIX C
LINKSYS' CONFIGURATION SCRIPTS

```
#!/bin/sh

#################### Configuration du linksys
    ##############
### Usage : ./script (start|stop) ipv6
    gatewayv6
### ipv6 et gatewayv6 sont optionnels
###
#
    ####################################################


#################### Tommy
    ################################
export USER=root
export HOME=/root
export PS1=\u@\h:\w\$
export LOGNAME=root
export TERM=xterm
export PATH=/bin:/sbin:/usr/bin:/usr/sbin
export SHELL=/bin/ash
export PWD=/root


################ Configuration
    ##################


sleep 2 ;
## Specifique a chaque linksys, a remplir par
    utilisateur ##

# La MTU
MTU=1400

# Range de ports, les deux doivent etre egales
# TODO : une seule variable...
PORTS="10000 11000"
PORTS2="10000-11000"

# Doit etre pareil que la configuration dhcp (/
    etc/conf/dhcp)
RESEAU="192.168.2.0/24"
```

```
# Le nom de fichier doit etre different sur les
    deux linksys
FILE_TEMP="/tmp/envoiprr1"

## Generale ##

# Adresse V6 du PRR
PRRV6=2001:660:7301:d002:250:baff:febe:711

# Adresse V4 du PRR
PRRV4=192.108.119.26

# Adresse V4 partagee
LOCALV4=192.108.119.27

# Adresse DNS
DNS="192.108.119.26"

# Script a lancer sur PRR
SCRIPT="/home/reloader/reload.sh"

## Les configurations automatiques (ou en
    arguments) ##

# Prelude : on lance rdisc6 pour forcer l'
    obtention d'une adresse
# (bug openwrt, lance le RA sur eth0 au
    demarrage)
rdisc6 --quiet eth0.1
sleep 2
# Adresse V6 locale
# On peut appeller en argument l'ipv6 a
    utiliser
# Sinon, on tente de la determiner
if test $2
then
  LOCALV6=$2
else
  # TODO : remplacer par une seule commande awk
  LOCALV6_NET=`/usr/sbin/ip -6 addr show dev
      eth0.1 | grep inet6 |grep -m 1 -v fe80::|
       awk '{print $2}'`
  LOCALV6=`echo $LOCALV6_NET | awk -F \/ '{
      print $1}'`
fi


# passerelle V6
# si elle n'est pas en argument, on tente de la
    trouver seul
if test $3
then
  PASSERV6=$3
else
  PASSERV6=`route -A inet6 |grep -m 1 ::/0|awk
      '{print $2}'`
fi

#################### Mise en place locale
    ####################


# On remet a zero les regles iptables
    precedentes
echo "Remise a zero des regles iptables"
iptables --flush
iptables -P FORWARD ACCEPT
iptables --flush -t nat

# On stop le tunnel s'il existe
echo "Arret du tunnel"
ip addr flush dev ip6tnl1
ip link set ip6tnl1 down

# On signale au linksys d'utiliser le bon DNS
echo "nameserver $DNS" > /etc/resolv.conf

# On est jamais trop prudent, et on active le
    forwarding
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding

# On configure le tunnel ipv4 in ipv6
echo "tunnel entre $LOCALV6 (local) et $PRRV6 (
    PRR)"
ip -6 tunnel add ip6tnl1 mode ip4ip6 remote
    $PRRV6 local $LOCALV6
ip link set ip6tnl1 up

echo "On utilise une MTU de $MTU"
ifconfig ip6tnl1 mtu $MTU

# L'adresse IPv4 partagee
echo "IPv4 utilisee : $LOCALV4"
ip addr add $LOCALV4 dev ip6tnl1

# Utile uniquement pour que le linksys possede
    une connexion locale
# il faut egalement un coup de mtu sur le
    tunnel ensuite
echo "Tranche de ports : $PORTS"
echo $PORTS > /proc/sys/net/ipv4/
    ip_local_port_range

# Bug sur openwrt, l'ipv6 remonte jusqu'a la
    carte reseau
ip -6 addr flush dev eth0


# On met l'ipv6 sur la bonne interface
echo "IPv6 sur interface Wan : $LOCALV6_NET et
    passerelle $PASSERV6"
ip -6 addr add $LOCALV6_NET dev eth0.1
ip -6 route add default via $PASSERV6 dev eth0
    .1

# ipv6 locale : utile uniquement si on souhaite
     fournir de l'ipv6 aux clients sur le lan
# a combiner avec un radvd pour annoncer les RA
# inutile pour tommy
# ip -6 addr add 2a01:240:fe55:3::/64 dev br-
    lan

# on communique avec le PRR par le tunnel
ip route add $PRRV4 dev ip6tnl1
# Enfin, on annonce que la gateway IPv4 est le
    PRR
ip route add default via $PRRV4

# on envoie en SNAT sur les bons ports les
    paquets du reseau local (UDP et TCP)
iptables -t nat -A POSTROUTING -s $RESEAU --
    proto 6 -j SNAT --to-source $LOCALV4:
    $PORTS2
iptables -t nat -A POSTROUTING -s $RESEAU --
    proto 17 -j SNAT --to-source $LOCALV4:
    $PORTS2
```

```
############################ Mise a jour du PRR
    #################

# Envoi de la configuration de l'interface


echo "Cpe1AddrV6 $LOCALV6" > $FILE_TEMP
echo "Cpe1PortRange $PORTS2" >> $FILE_TEMP
sleep 2 ;

# on copie d'abord la configuration
/usr/bin/scp -i /root/.ssh/id_rsa $FILE_TEMP
    reloader@[$PRRV6]:/home/reloader/

sleep 5 ;
# La connexion provoquera le script de shell de
    reloader
# Ce script se charge de construire la conf
    entiere du PRR et de relancer
ssh -l reloader -i  /root/.ssh/id_rsa $PRRV6
    $SCRIPT $LOCALV6
```

Listing 1. IPTables configuration script with ip4-in-ip6 tunneling

APPENDIX D
MODIFIED LINKSYS' USER GUIDE

Quick start

Linksys plug: The Linksys must connect to the PRR while booting. To start the device, you must:

1) Plug a RJ-45 cable between the Linksys WAN socket and an Ethernet socket with IPv6 connectivity
2) Plug the electrical wire (after the Ethernet cable)
3) Plug the client computers and get an IP address through DHCP

No extra configuration is needed. A little wait time (30 seconds) could be required to obtain a connectivity to the Internet.
As a piece of information, the Linksys provides a private IPv4 addresse in the 192.168.2.0/24 subnet. The shared public address is 192.108.119.27, and the Linksys uses the 10.000-11.000 port range.

Minimal configuration

IPv6 connectivity: The Linksys device must access an IPv6 link. Here are the extra criteria to ensure good performances:

- the network must answer with a Router Advertisment when the Linksys device sends a Router Solicitation packet
- link's MTU must be higher than 1400
- IPv6 address 2001:660:7301:d002:250:baff:febe:711 must be reachable, without any filtering on the TCP/22 port (SSH) or on IPv6 protocol #4 (IPv4 encapsulation).

Known issues

WiFi: The Linksys is not WiFi compatible. The installed OpenWRT version does not make it possible to use the WiFi chipset.

PRR limits:

- Only TCP & UDP are transmitted. ICMP does not work (there is no notion of ports in ICMP, so that the PRR drops ICMP packets)
- FTP malfunctions

Backup connection to the Linksys

In case of emergency, a SSH connection is available on the Linksys coming from a client machine on the LAN. The host address is 192.168.2.1. The link to the Linksys is made by a script (located in `/script`). If the autoconfiguration does not work, it is possible to initalize a configuration manually by calling `/script start IPV6-PUBLIQUE-WAN GATEWAY-V6`.

However, a simple electrical unplug-replug should make it possible to reinitialize the connection.

ACKNOWLEDGMENT

The authors would like to thank Laurent Toutain, lecturer in Telecom Bretagne (France), Patrick Maillé, lecturer in Telecom Bretagne (France), Tanguy Ropitault, researcher in Telecom Bretagne (France), Rémi Després, consultant (unipersonal company) at RD-IPtech and all our volunteer testers.

REFERENCES

[1] M. Boucadair, *Procedure to bypass DS-lite CGN/AFTR: IPv6 Tunnel Endpoint Extension Header*, draft-boucadair-softwire-cgn-bypass-00.txt, November 2009.
[2] M. Boucadair, *Procedure to bypass DS-lite AFTR*, draft-boucadair-softwire-cgn-bypass-01.txt, March 2010.
[3] M. Boucadair, *Deploying Dual-Stack Lite (DS-Lite) in IPv6 Network*, draft-boucadair-dslite-interco-v4v6-03, February 2010.
[4] M. Boucadair, *Flexible IPv6 Migration Scenarios in the Context of IPv4 Address Shortage*, draft-boucadair-behave-ipv6-portrange-04, October 2009.
[5] M. Boucadair, *Behaviour of BitTorrent service in an IP Shared Address Environment*, draft-boucadair-behave-bittorrent-portrange-02.txt, January 2009.
[6] Iljitsch van Beijnum, *Comcast sees end of IPv4 tunnel, beginning IPv6 trial*, http://arstechnica.com/tech-policy/news/2010/01/comcast-running-out-of-ipv4-addresses-beginning-ipv6-trial.ars, January 2010.
[7] Salman A. Baset and Henning Schulzrinne, *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*, http://www.cs.columbia.edu/%7Elibrary/TR-repository/reports/reports-2004/cucs-039-04.pdf, September 2004.
[8] Andy Boyett and Nicolas Thill, *Backfire 10.03 Beta*, https://forum.openwrt.org/viewtopic.php?id=23829, March 4th, 2010.
[9] S. Alexander and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, http://www.ietf.org/rfc/rfc2132.txt, March 1997.
[10] Jon Postel, *Transmission Control Protocol*, http://www.ietf.org/rfc/rfc793.txt, September 1981.
[11] C. Perkins, *IP Encapsulation within IP*, http://www.faqs.org/rfcs/rfc2003.html, October 1996.
[12] Netfilter Core Team, *nf_conntrack_proto_tcp.c*, http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.32.y.git;a=blob;f=net/netfilter/nf_conntrack_proto_tcp.c;h=ba2b769372834dafcb10bc124027842a86027850;hb=HEAD, February 23th, 2010.
[13] Netfilter Core Team, *nf_nat_proto_common.c*, http://git.kernel.org/?p=linux/kernel/git/stable/linux-2.6.32.y.git;a=blob;f=net/ipv4/netfilter/nf_nat_proto_common.c;h=6c4f11f514461a5a244ba1d70180f42ad82940b4;hb=HEAD, February 23th, 2010.
[14] The Google Team, *Google over IPv6: FAQ*, http://www.google.com/intl/en/ipv6/faq.html, 2010.
[15] Lorenzo Colitti, *IPv6 at Google*, http://www.apricot2010.net/__data/assets/pdf_file/0007/18997/IPv6-at-Google.pdf, 2010.