Internet Engineering Task Force
Internet Draft
Intended status: Informational
Expires: September 2, 2012

X.Deng
M.Boucadair
France Telecom
Y.Lee
Comcast
X.Huang
Q.Zhao
BUPT

March 1, 2012

Implementing A+P in the provider's IPv6-only network
draft-deng-softwire-aplusp-experiment-results-02.txt

Abstract

This memo describes an implementation of A+P in a provider's IPv6-only network. It provides details of the implementation, network elements, configurations and test results as well. Besides traditional port range A+P, a scattered port sets flavor of A+P is also implemented to verify feasibility of offering non-continuous port sets with A+P approach.

The test results consist of the application compatibility test, UPnP 1.0 extensions and UPnP 1.0 friendly port allocation for A+P, port usage and BitTorrent behaviors with A+P.

This memo focuses on the IPv6 flavor of A+P.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 2, 2012.

Table of Contents

1. Introduction

A+P [RFC6346] is a technique to share IPv4 addresses over IPv6-only network without requiring a NAT function in the provider's network. The main idea of A+P is borrowing some bits from the port number in the TCP/UDP header to identify the end point. Those port numbers assigned to the end point will be used by IPv4 applications. A+P can facilitate network migration to IPv6-only while continue to offer IPv4 connectivity to customers by tunneling IPv4 packets over IPv6-only network.

We implemented A+P in a residential ADSL access network, where IPv6-only access network is provided over PPPoE. In this memo, we first describe the implementation environment including A+P IPv6 prefix format and network elements configurations, then we describes the test results. In particular, this memo focuses on the SMAP function implementation specified in [RFC6346].

For more application test results in A+P environment, please refer to [draft-boucadair-behave-bittorrent-portrange-02] and [draft-boucadair-port-range-01].

2. Terminology

This memo uses the following terms:

o PRR: Port Range Router

o A+P CPE: A+P aware Customer Premise Equipment

3. Implementation environment

3.1. Environment Overview

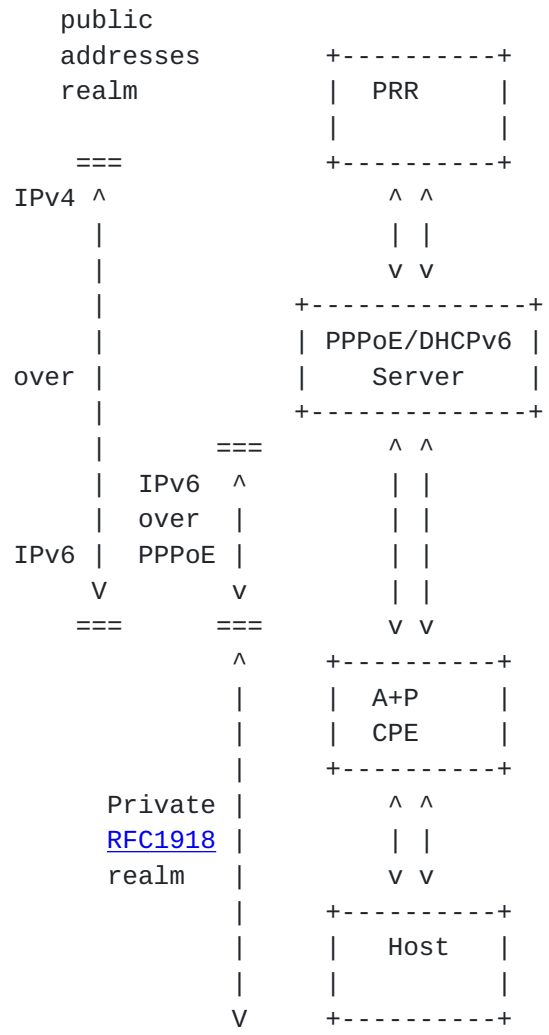

Figure 1 : Implementation Environment

We developed both A+P CPE function and Port Range Router (PRR) function on Linux. A+P CPE function was implemented on Linksys WRT54GS router running OpenWRT 2.6.32. PRR function was implemented on standard Intel based server. Figure 1 shows the high-level network diagram of the test environment.

Figure 2 shows the configuration of A+P CPE. IPv6 prefix was provisioned over PPPoE to CPE by a DHCPv6 server. In addition, it also offered A+P parameters via DHCPv6 options defined in [draft-boucadair-dhcpv6-shared-address-option].

Model	CPU Speed (MHz)	Flash (MB)	RAM (MB)	Wireless NIC	Wireless Standard	Wired Ports
Linksys WRT54GS	200	8	32	Broadcom (integrated)	11g	5

Figure 2 :Parameters of A+P CPE

3.2. Implementation and Configuration of A+P

A+P CPE uses Netfilter framework to implement the port-set restricted NAT. Port set restricted NAT operation was done by iptables rules. After the port restricted NAT operation, IPv4 packets were sent to a TUN interface which was a virtual network interface in Linux. The TUN interface is a virtual interface that performs the IPv4-in-IPv6 function. Using the IPv4-Embedded IPv6 address format defined in section 3.2.1, an IPv4-in-IPv6 function is performed by the TUN interface handler.

PRR bridges the IPv6 access network to the IPv4 Internet. It contains two main functions: 1) IPv4-in-IPv6 encapsulation/decapsulation; Similar to A+P CPE, PRR implementation leveraged the virtual TUN driver handler for IPv4-in-IPv6 function. 2) Destination IPv4 address and layer 4 port based routing function is responsible for routing the IPv4 traffic originated from the IPv4 Internet to the Port Range restricted A+P CPE. The goal of PRR is to deliver the IPv4 packet to the A+P CPE that was assigned with the port number used in the destination port in the layer 4 header. Since PRR delivers the IPv4 packet over IPv4-in-IPv6 tunnel, PRR can embed the IPv4 address and port number in the IPv6 address. The IPv4-Embedded IPv6 address is used to uniquely identify the A+P CPE. Details of how to construct the IPv4-Embedded IPv6 address format is defined in Section 3.2.1.

3.2.1. IPv4-Embedded IPv6 Address Format For A+P CPE

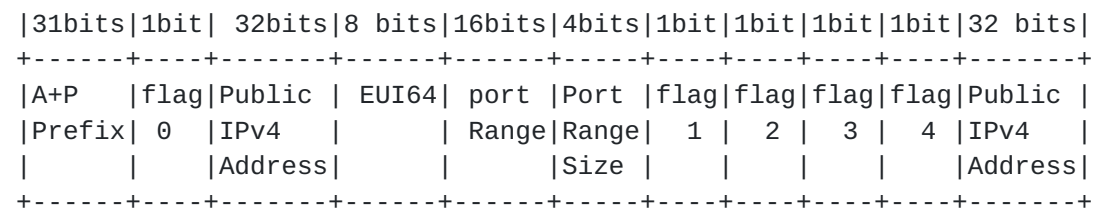

Figure 3 :IPv4-Embedded IPv6 address format

flag0: Is this address used by CPE or PRR?

flag1: Is address shared?

flag2: Is length of invariable present?

flag3: Is port range identifying sub network?

flag4: Reserved?

To facilitate other parties who are also interested in testing A+P solution, we are considering to release this A+P implementation under open source license. For more implementation details, please refer to [Implementing A+P].

3.2.2. DHCPv6 Configurations

DHCPv6 options defined in [draft-boucadair-dhcpv6-shared-address-option] were implemented. These options allow configuring a shared address and a port range using a DHCPv6 option.

3.2.3. Avoiding Fragmentation

Normally the host TCP/IP protocol stack uses TCP protocol stack uses Maximum Segment Size (MSS) option and/or Path Maximum Transmission Unit Discovery (PMTUD) to determine the MTU.

However adding the IPv6 Header and the PPPoE header to the IPv4 packet may exceed the maximum MTU of the wire and consequently results in IP fragmentation.

One solution is to add a rule to iptables on A+P CPE to modify the MSS value in TCP SYN and SYN-ACK. This can be done using command "iptables -t mangle -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss DESIRED_MSS_VALUE". The DESIRED_MSS_VALUE is set to exclude IPv4 header, TCP header, IPv6 header and PPPoE header length.

3.3. Implementing non-continuous Port Sets for A+P

3.3.1. Non-continuous Port Sets allocation mechanism

[I-D.ietf-intarea-shared-addressing-issues] states that a bulk of incoming ports can be reserved as a centralized resource shared by all subscribers using a given restricted IPv4 address. We could distribute a range of continuous ports to each subscriber. This may create security concerns such as blind attack. An alternative would be to assign a bulk of non-continuous random ports to each subscriber. The following session would describe the implementation of non-continuous port-set.

Note that the non-continuous port-set allocation mechanism described here is just one possible solution to implement non-continuous port provisioning. The implementation itself is to achieve two goals: 1) Proving of feasibility of non-continuous port-set with A+P approach; 2) Evaluating UPnP 1.0 compatibility with non-continuous port-set. Experiment results are provided in Section 4.2.2. Given a port-set size N, log2(N) bits are randomly chosen as subscribers identification bits(S-bit). S-bit must be chosen between 1st and 16th bits. For example: if sharing ration is 1:32, each subscriber will have five S-bits. Figure 4 shows an example of 5 S-bits (2nd, 5th, 7th, 9th and 11th) for a subscriber.

Subscriber ID pattern is formed by setting all the S-bits to 1 and other trivial bits to 0. Figure 5 illustrates an example of subscriber ID pattern based on S-bits example in Figure 4.

Note that the subscriber ID pattern must be identical for each subscriber that shares the same IPv4 address.

Subscribers ID value is assigned by setting subscriber ID pattern bits (s bits shown in figure 4) to a unique customer value to identify each customer and setting other trivial bits to 1. An example of subscriber ID value, having a subscriber ID pattern shown in the figure 5 and a customer value 0, is shown in the figure 6.

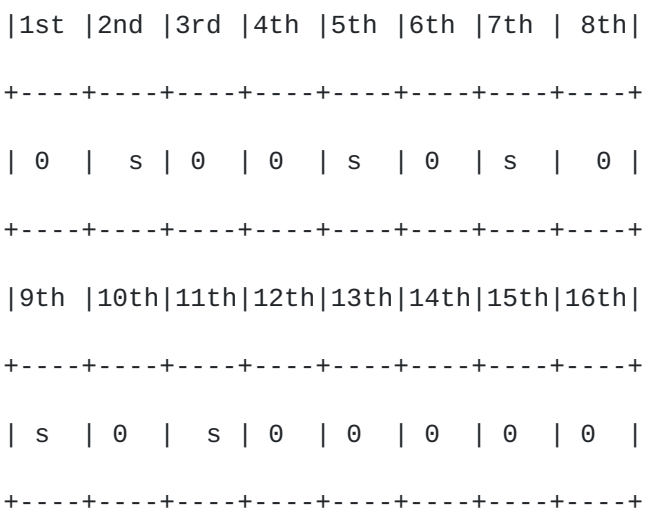

Figure 4 : An S-bit selection example (on a sharing ration 1:32 address).

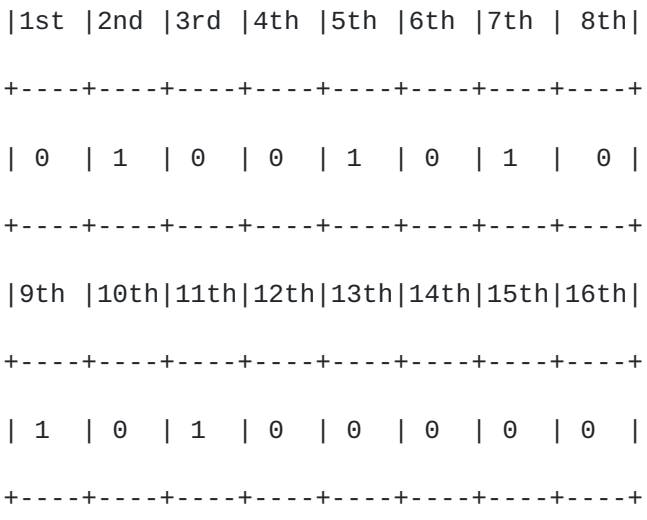

Figure 5 : A subscriber ID pattern example (on a sharing ration 1:32 address).

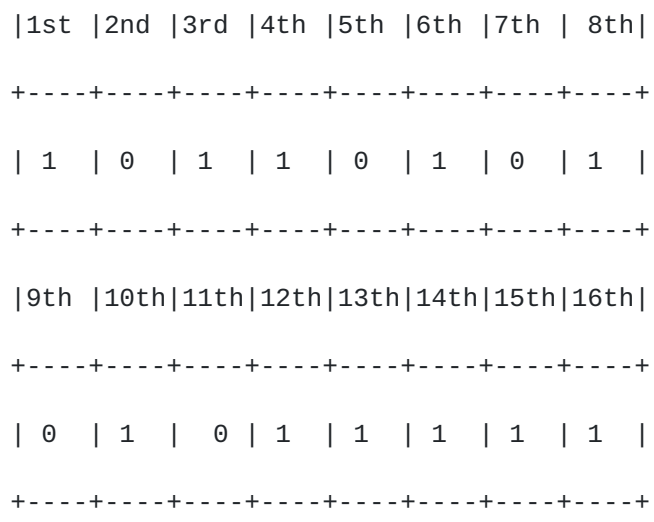

Figure 6 : A subscriber ID value example (customer value: 0)

Subscriber ID pattern and subscriber ID value together uniquely define a restricted port set (Non-contiguous port sets or a contiguous port range, depends on Subscriber ID pattern and subscriber ID value) on a restricted IP address.

Pseudo-code shown in the Figure 7 describes how to use subscriber ID pattern and subscriber ID value to implement a random ephemeral port selection function within the defined restricted port sets on a customer NAT.

```
do{
    restricted_next_ephemeral = (random()|subscriber_ID_pattern)
                                & subscriber_ID_value;
    if(five-tuple is unique)
    return restricted_next_ephemeral;
}
```

Figure 7 : Random ephemeral port selection within the restricted port set

3.3.2. IPv4-Embedded IPv6 Address Format for Non-continuous Port Sets A+P CPE

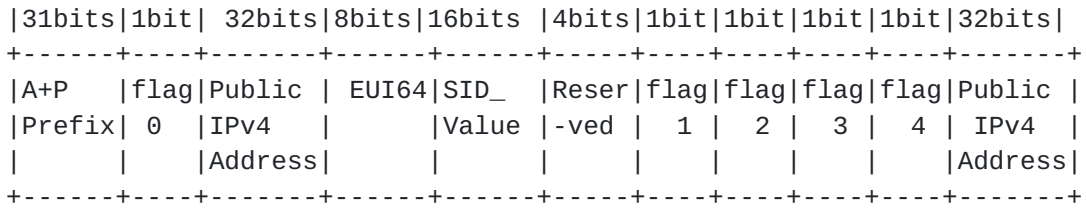

Figure 8 :IPv4-Embedded IPv6 address format

SID Value: Subscriber_ID_Value, which is unique for per subscriber sharing a given restricted IPv4 address. and has been allocated to each subscriber.

flag0: Is this address used by CPE or PRR?

flag1: Is address shared?

flag2: Is length of invariable present?

flag3: Is port range identifying sub network?

flag4: Reserved?

To support non-continuous port-set, PRR maintains a mapping table which contains the pairs of restricted IPv4 address and it's Subscriber ID Pattern. To form an IPv6 destination address for incoming packet, PRR could find the right SID Pattern according to a destination IPv4 address, and then apply a simple operation shown in the figure 9.

```
SID_Value = Destination_Port | (~SID_Pattern);
```

Figure 9 :PRR calculates SID Value

3.3.3. Customize a non-continuous Ports Set A+P NAT

On a Linux kernel 2.6.32.36, only one line of linux kernel code was Changed to implement this feature. Figure 10 shows the change. Figure 11 show the IPtables commands required in the PRR. The beginning of port range changed to SID_Value and the ending of the port range changed to SID_Pattern.

```
bool nf_nat_proto_unique_tuple(...)

  ...

//The Original code:

 //*portptr = htons(min + off % range_size);

 // was changed to:

   *portptr = htons((ntohs(off) | min ) & max );

   ...
```

Figure 10:Function of finding a unique 5-tuple for a non-continuousport sets A+P NAT

```
iptables -t nat -A POSTROUTING -o eth0 -p tcp -j SNAT --to-source
a.b.c.d: SID_Value-SID_Pattern --random

iptables -t nat -A POSTROUTING -o eth0 -p udp -j SNAT --to-source
a.b.c.d: SID_Value-SID_Pattern --random
```

Figure 11: IPtables commands for a non-continuousports set A+P NAT

4. Application Tests and Experiments in A+P Environment

A set of well-known applications was tested. The tests compared A+P over IPv6 and simple A+P without encapsulation on a pain IPv4 network. The test results showed that both share the same impacts [draft-boucadair-port-range-01]. Web browsing (IE and Firefox), Email (Outlook), Instant message(MSN),Skype, Google Earth work normally with A+P. For more details, please refer to [draft-boucadair-port-range-01].

4.1. A+P Impacts on Applications

Application	A+P impacts
IE	None
Firefox	None
FTP(Passive mode)	None
FTP(Active mode)	require opening port forwarding
Skype	None
Outlook	None
Google Earth	None
BitComet	UPnP extensions may be required, when listening port is out of A+P range; other minor effects(see Section 4.4)
uTorrent	UPnP extensions may be required, when listening port is out of A+P range; other minor effects(see Section 4.4)
Live Messenger	None

Figure 12: A+P impacts on applications

P2P (Peer-to-Peer) applications using specific port for inbounding connection are likely to fail, because the specific ports may not be available for that A+P subscriber. Some UPnP extensions may be required to make P2P applications work properly with A+P. Other minor effects of A+P are discussed in Section 4.4.

4.2. UPnP extension experiment

4.2.1. UPnP 1.0 extension

To make P2P application work properly with port restricted NAT , we have designed extensions including new variables, new error codes as well as new actions to UPnP 1.0, and have them implemented with [Emule], [open source UPnP SDK 1.0.4 for Linux] and [Linux UPnP IGD 0.92].

In figure 5, a new error code is proposed for the existing "AddPortMapping" action to explicitly indicate the situation that the requested external port is out of range.

ErrorCode	errorDescription	Description
728	ExternalPortOutOfRange	The external port is out of the port range assigned to this external interface

Figure 13:New ErrorCode for "AddPortMapping" action

New state variables have been introduced to reflect the valid port range. The definitions of these state variables are shown in figure 6.

Variable Name	Req. or Opt.	Data Type	Allowed Value	Default Value	Eng. Units
PortRangeLow	O	ui2	>=0	0	N/A
PortRangeHigh	O	ui2	<=65535	65535	N/A

Figure 14: New state variables for port range

Correspondingly, new actions, GetPortRangeLow and GetPortRangeHigh, defined to retrieve port range information are illustrated in figure 7. An IP address should be provided as argument to invoke the new actions, for the port range is associated with a specific IP address.

Action Name	Argument	Dir.	Related StateVariable
GetPortRangeLow	NewExternal IPAddress	IN	ExternalIPAddress
	NewPortRange Low	OUT	PortRangeLow
GetPortRangeHigh	NewExternal IPAddress	IN	ExternalIPAddress
	NewPortRange High	OUT	PortRangeHigh

Figure 15: New actions for port range

Please refer to [UPnP Extension] for more details of UPnP extension experiment in A+P.

4.2.2. UPnP 1.0 friendliness attempts

Application	Behaviors
Microtorrent v2.2 (also known as uTorrent)	call GetSpecificPortMapping by incremental by 1 each time, until find an external port available, and then call AddPortMapping,or return error after five failures
Emule v0.50a	call AddPortMapping, after finding the external port not available return error
Azureus v4.6.0.2	call AddPortMapping, after finding the external port not available, try the same port 5 more times by call AddPortMapping, then return error
Shareazav2.2.5.7	call GetSpecificPortMapping, after finding the external port not available, return error without issuing AddPortMapping

Figure 16 UPnP 1.0 applications behaviors of asking for an external port

The Behaviors test results in the previous figure shows that if a request of external port failed, some UPnP 1.0 applications, namely Microtorrent v2.2 and Azureus v4.6.0.2 attempt to issue (at most) 4 more times request until succeed. With each external port request attempts, the desired external port is incremented by 1 of the previous requested external port.

Hence, allocating port sets in a way that each A+P subscriber has sub sets interval less than 5 would make some UPnP 1.0 applications succeed in 5 times retrying. For example, In case a Subscriber ID Pattern 0x02 that makes 2 customers sharing one IPv4 address, and customer 1 have the available ports
{ 0,1 | 4,5 | 8,9 |12,13|....} while customer 2 have the available ports:
{ 2,3 | 6,7 | 10,11|14,15|....}

Microtorrent v2.2 and Azureus v4.6.0.2 would be compatible with port restriction feature of A+P.

IGD:1 is known to be broken in shared address environment [RFC6269]; IGD:2 mitigates the issues encountered in IGD:1. The efforts, documented in section 4.2, were attempts before standardization of IGD:2.

4.3. Port Usage of Applications

Port consumptions of applications not only impact the deployment factor (i.e., port range size) for A+P solution but also play an important role in determining the port limitation of per customer on AFTR for Dual-Stack Lite.

Therefore we have also developed and deployed a Service Probe in our IPv6 network, which use IPv6 TCP socket to ask A+P CPE for NAT session usage, and store A+P NAT statistics in a Mysql database for further analysis of application behaviours in terms of port and session consumptions.

In figure 8, the maximum port usage of each application is the peak number of port consumption per second during the whole communication

process. The duration time represents the total time from the first NAT binding entry being established to the last one being destroyed.

Application	Test case	Maximum port usage	Duration (seconds)
IE	browsing a news website	20-25	200
	browsing a video website	40-50	337
Firefox	browsing a news website	25-30	240
	browsing a video website	80-90	230
Chrome	browsing a news website	50-60	340
	browsing a video website	80-90	360
Android Chrome	browsing a news website	40-50	300
	browsing a video website	under 10	160
Google Earth	locating a place	30-35	240
Android Google Earth	locating a place	10-15	240
Skype	make a call	under 10	N/A
BitTorrent	downloading a file	200	N/A

Figure 17: Port usage of applications

4.4. BitTorrent Behaviour in A+P

[draft-boucadair-behave-bittorrent-portrange] provides an exhaustive testing report about the behaviour of BiTtorrent in an A+P architecture. [draft-boucadair-behave-bittorrent-portrange] describes the main behavior of BitTorrent service in an IP shared address environment. Particularly, the tests have been carried out on a

testbed implementing [ID.boucadair-port-range] solution. The results are, however, valid for all IP shared address based solutions.

Two limitations were experienced. The first limitation occurs when two clients sharing the same IP address want to simultaneously retrieve the SAME file located in a SINGLE remote peer. This limitation is due to the default BitTorrent configuration on the remote peer which does not permit sending the same file to multiple ports of the same IP address. This limitation is mitigated by the fact that clients sharing the same IP address can exchange portions with each other, provided the clients can find each other through a common tracker, DHT, or Peer Exchange. Even if they can not, we observed that the remote peer would begin serving portions of the file automatically as soon as the other client (sharing the same IP address) finished downloading. This limitation is eliminated if the remote peer is configured with bt.allow_same_ip == TRUE.

The second limitation occurs when a client tries to download a file located on several seeders, when those seeders share the same IP address. This is because the clients are enforcing bt.allow_same_ip parameter to FALSE. The client will only be able to connect to one sender, among those having the same IP address, to download the file (note that the client can retrieve the file from other seeders having distinct IP addresses). This limitation is eliminated if the local client is configured with bt.allow_same_ip == TRUE, which is somewhat likely as those clients will directly experience better throughput by changing their own configuration.

Mutual file sharing between hosts having the same IP address has been checked. Indeed, machines having the same IP address can share files with no alteration compared to current IP architectures.

5. Security Considerations

TBD

6. IANA Considerations

This document includes no request to IANA.

7. Conclusion

Despite A+P introduces some impacts on existence applications, issues of P2P applications due to the port restricted NAT have been resolved by UPnP extension experiment in our test bed, and other issues are shared by other IP address sharing solutions. Therefore, from our work, it has been proved that deploying both port range and non-

continuous port sets A+P in the Service Provider's IPv6 network during IPv6 transition period is feasible.

8. References

8.1. Normative References

[Implementing A+P]

Xiaoyu ZHAO.,"Implementing Public IPv4 Sharing in IPv6 Environment", ICCGI 2010

[UPnP Extension]

Xiaoyu ZHAO., "UPnP Extensions for Public IPv4 Sharing in IPv6 Environment", ICNS 2010

8.2. Informative References

[RFC6346]

R. Bush., " The Address plus Port (A+P) Approach to the IPv4 Address Shortage", August,2011.

[draft-boucadair-dhcpv6-shared-address-option]

M. Boucadair., "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions", draft-boucadair-dhcpv6-shared-address-option-01 (work in progress), December 21, 2009

[draft-boucadair-port-range-01]

"IPv4 Connectivity Access in the Context of IPv4 Address Exhaustion", draft-boucadair-port-range-01(work in progress), January 30, 2009

[Emule]

http://www.emule-project.net/. [Accessed October 26, 2009]

[UPnP SDK 1.0.4 for Linux]

http://upnp.sourceforge.net/. [Accessed October 26, 2009].

[Linux UPnP IGD 0.92].

http://linuxigd.sourceforge.net/. [Accessed October 26, 2009].

[draft-boucadair-behave-bittorrent-portrange]

M. Boucadair.,"Behaviour of BitTorrent service in an IP Shared Address Environment", draft-boucadair-behave-bittorrent-portrange-02.txt

9. Additional Authors

Lan Wang
France Telecom
Hai dian district, 100190, Beijing, China

Email: lan.wang@orange-ftgroup.com

Tao Zheng
France Telecom
Hai dian district, 100190, Beijing, China

Email: tao.zheng@orange-ftgroup.com

Yan MA
Beijing University of Post and Telecommunication
Email: mayan@bupt.edu.cn

10. Acknowledgments

The experiments and tests described in this document have been explored, developed and implemented with help from Zhao Xiaoyu, Eric Burgey and JACQUENET Christian.

Appreciation to Randy Bush's intitial idea of documenting these experience results, for share the knowledge of what we have learnt with the community.

Thanks to Jan Zorz for comments.

11. Authors' Addresses

Xiaohong Deng
France Telecom
Hai dian district, 100190, Beijing,
China

Email: dxhbupt@gmail.com

Mohamed BOUCADAIR
France Telecom
Rennes,35000 France

Email: mohamed.boucadair@orange-ftgroup.com

Yiu L. Lee
Comcast
One Comcast Center
Philadelphia, PA 19103
U.S.A.

Email: Yiu_Lee@Cable.Comcast.com

Xiaohong Huang
Beijing University of Post and Telecommunication
Email: huangxh@bupt.edu.cn

Qin Zhao
Beijing University of Post and Telecommunication
Email: zhaoqin.bupt@gmail.com