V6OPS
Internet Draft
Intended status: Informational
Expires: January 9, 2012

X.Deng
T.Zheng
M.Boucadair
L.Wang
France Telecom
X.Huang
Q.Zhao
Y.Ma
BUPT

July 8, 2011

Implementing AplusP in the provider's IPv6-only network
draft-deng-v6ops-aplusp-experiment-results-01.txt

Status of this Memo

Copyright Notice

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 flavour of A+P is also implemented and verified for the sake of distributing incoming ports among customers in a more discrete way. The test results consist of the application compatibility test, UPnP extension for A+P, port usage and BitTorrent behaviour with A+P.

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

Table of Contents

1. Introduction

A+P [draft-ymbk-aplusp-09] is a technique to share IPv4 addresses during the IPv6 transition period without requiring a NAT function in the provider's network. The main idea of A+P is treating some bits from the port number in the TCP/UDP header as additional end point

identifiers to extend the address field, thereby leaving a range of ports available to applications. This feature facilitates migration of networks to IPv6-only while offering the IPv4 connectivity services to customers, because the IPv4 address and the significant bits from the port range can be encoded in an IPv6 address and therefore transporting IPv4 traffic over IPv6 network by stateless IPv6 routing.

We have implemented A+P in a residential ADSL access network, where IPv6-only access network is provided over PPPoE. In this document, we describe the implementation environment including A+P IPv6 prefix format and network elements configurations, and results of application tests as well. The document focuses on the implementation of the SMAP function specified in [draft-ymbk-aplusp-09]:

- o Implement DHCPv6 options to retrieve an IPv4-embedded IPv6 address and a port range.
- o Support of those DHCPv6 options in both the DHCPv6 server side and the DHCPv6 client side.
- o Support of those DHCPv6 options in both the DHCPv6 server side and the DHCPv6 client side.

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

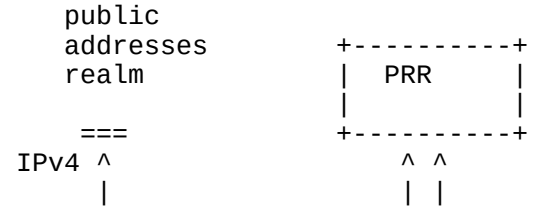
2. Terminology

This document makes use of 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



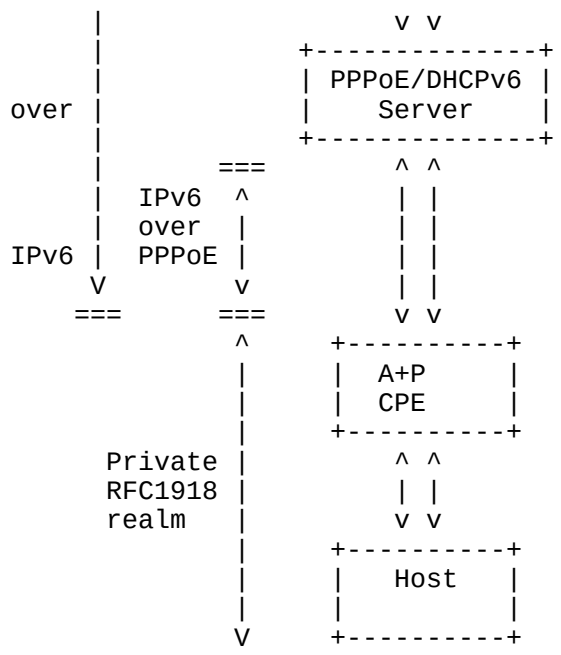


Figure 1 : Implementation Environment

We had developed both A+P home gate way function and Port Range Router (PRR) function on Linux platform and ported the home gate way function to a Linksys wrt 54G CPE, on which an openwrt 2.6.32 (based on Linux kernel) is running.

Figure 2 shows the Parameters of A+P CPE. IPv6 is provisioning over PPPoE to CPE while DHCPv6 server offers IPv6 prefix and A+P

parameters by extended 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

Aplusp CPE, using Netfilter framework, the IPv4 port restricted NAT operation performed by CPE has been implemented by simply rules through iptables tool on Linux. After the port restriceted NAT operation, the IPv4 packets are sent to a TUN interface which is described as a virtual network interface in Linux. Using the IPv4-Embedded IPv6 address format defined in section 3.2.1, an IPv4-in-IPv6 encapsulation/decapsulation is performed by the TUN interface handler.

PRR, located in the interconnection point of the IPv6 network and IPv4 network, has been implemented with two main functions: 1) IPv4-in-IPv6 encapsulation/decapsulation; Like CPE, TUN driver is also used in PRR to achieve function IPv4-in-IPv6 encapsulation/decapsulation. 2) destination port based routing function, which is responsible for routing the IPv4 traffic originated from the IPv4 Internet to the Port Range restricted A+P CPE. Destination port based routing is implemented by generating IPv6 destination address, pre-assigned from IPv4 address and port range to each CPE, according to IPv4-Embedded IPv6 address format defined in section 3.2.1.

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

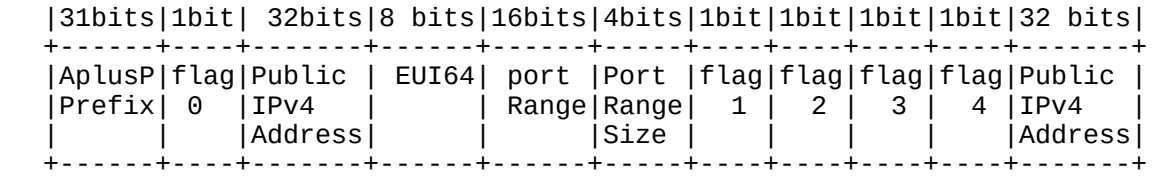


Figure 3 :IPv4-Embedded IPv6 address format

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 test and experiment on AplusP solution, recently, we are considering release this AplusP 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] have been implemented. These options allow to configure a shared address together with a port range using DHCPv6.

3.2.3. Avoiding Fragmentation

Normally the TCP protocol stack will employ Maximum Segment Size (MSS) negotiation and/or Path Maximum Transmission Unit Discovery (PMTUD) to determine

the maximum packet size, and then try to send as large as possible datagram to achieve better throughput. However the IPv4-in-IPv6 encapsulation and the PPPoE header is very likly to cause a larger packet that exceeds the maximum MTU of the wire, and result in undesired fragmentation processing and decrease transmission

efficiency.

A simple solution is to enable iptables on A+P CPE to modify the MSS value of TCP session, using the command like "iptables -t mangle -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss DESIRED_MSS_VALUE". Here the DESIRED_MSS_VALUE is taken into account of common size of IPv4 header without options, common size of TCP header and size of basic IPv6 header and PPPoE header as well.

3.3. Implementing scattered Port Sets for A+P

3.3.1. Scattered Port Sets allocation mechanism

As described in [I-D.ietf-intarea-shared-addressing-issues], a bulk of incoming ports can be reserved as a centralized resource shared by all subscribers using a given restricted IPv4 address. In order to distribute incoming ports as scattered as possible among subscribers sharing the same restricted IPv4 address, other than allocating a continuous range of ports to per subscriber, a solution to distribute bulks of non-continuous ports among subscribers, which also takes port randomization of CPE NAT into account, because port randomization is one protection among others against blind attacks, is elaborated thereby.

On every restricted IPv4 address, according to port set size N, log2(N)bits are randomly chose as subscribers identification bits(s bit) among 1st and 16th bits. Take a sharing ration 1:32 for example, Figure 4 shows an example of 5bits (2nd, 5th, 7th, 9th, 11th) being chose as s bit.

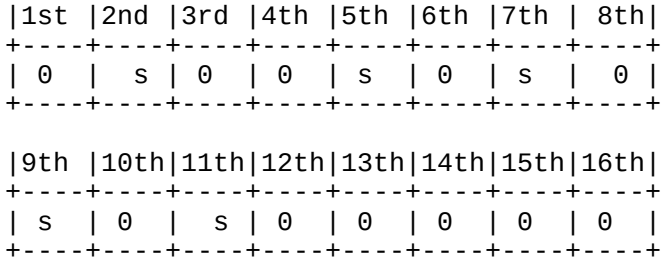


Figure 4 : An s bit selection example (on a sharing ration 1:32 address).

Subscriber ID pattern is then formed by setting all the s bits to 1

and other trivial bits to 0. Figure 5 illustrates an example of subscriber ID pattern which follows the s bit selection of figure 4. Note that the subscriber ID pattern can be different, ensured by the random s bit selection, per restricted IP address no matter whether the sharing ratio varies.

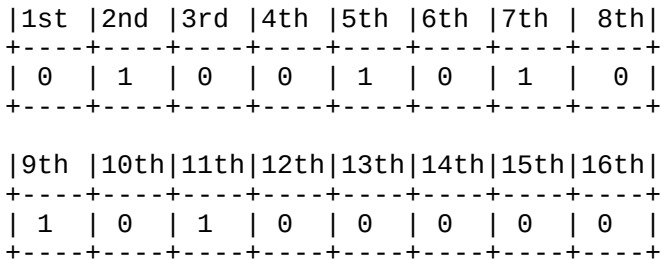


Figure 5 : A subscriber ID pattern example (on a sharing ration 1:32 address).

Subscribers ID value is then assigned by setting subscriber ID pattern bits (s bits shown in figure 4) to a unique customer value 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.

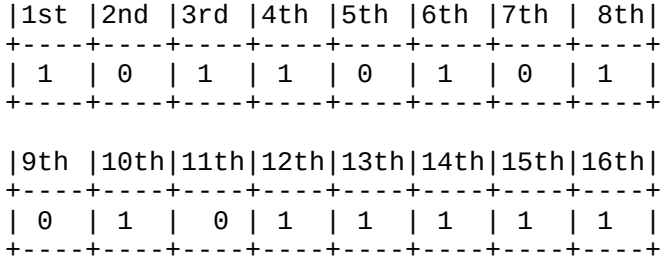


Figure 6 : A subscriber ID value example (customer value: 0)

Subscriber ID pattern and subscriber ID value together uniquely defines 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 Scattered Port Sets A+P CPE

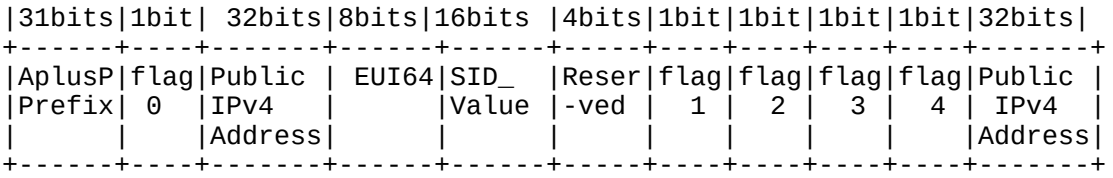


Figure 8 :IPv4-Embedded IPv6 address format

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?

PRR maintains a mapping table, which consists 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 scattered Ports Set A+P NAT on Linux

With a linux kernel 2.6.32.36, only one line of linux kernel code is changed, as shown in the figure5, and the same IPtables command line interface is used with the only one change of semantic that the original staring of port range becomes SID_Value and the ending port of a port range becomes SID_Pattern. The command line with iptables to configure a scattered Ports Set A+P is illustrated in the figure 11.

```
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 scattered port 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 scattered ports set A+P NAT

4. Application Tests and Experiments in A+P Environment

A set of well-known applications have been tested in this IPv6 flavor of A+P environment to access A+P impacts on them. The test results show that IPv6 flavor of A+P has the same impacts on applications as IPv4 flavor A+P does [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:Aplusp impacts on applications

For P2P (Peer-to-Peer) applications, when some of them listening on specific port to expect inbounding connection, it is likely to fail due to the listening port is out of A+P port range. 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

To make P2P application work properly with port restricted NAT , we have designed extensions including new variables, new errorcodes 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.

4.3. Port Usage of Applications

Port consumptions of applications not only impact the deployment factor (i.e., port range size) for AplusP 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 AplusP CPE for NAT session usage, and store AplusP NAT statistics in a Mysql database for further analysis of application behaviors 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 16: 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 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

[1] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[Fab1999] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[draft-ymbk-aplusp-09]

R. Bush., " The A+P Approach to the IPv4 Address Shortage", draft-ymbk-aplusp-09 (work in progress), February 17, 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. 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.

Thanks to Jan Zorz for comments.

Authors' Addresses

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

Email: xiaohong.deng@orange-ftgroup.com

Mohamed BOUCADAIR
France Telecom
Rennes,35000 France

Email: mohamed.boucadair@orange-ftgroup.com

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

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

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