Analysis on IPv6 Transition Solutions and Service Tests

Xiaohong Deng, Lan Wang, Tao Zheng, Daqing Gu
France Telecom Group, Beijing, China
{xiaohong.deng; lan.wang; daqing.gu}@orange-ftgroup.com

Eric Burgey
France Telecom R&D Paris, France
eric.burgey@orange-ftgroup.com

***Abstract*—during the long IPv6 transition phase, multiple transition approaches may co-exist in the same network to enable v4-to-v4 communication and v6-to-v4 communication over IPv6 access. In this paper, we present our integrated platform with different transition solutions, such as AplusP, Dual-stack Lite and NAT64/DNS64, and analyze application behaviors and potential issues that may impact on the deployments. Particularly, results of application tests indicate that dual-stack application may break either because application layer is IP version dependent or because application has difficulties with NAT64 traversal if only NAT64/DNS64 is deployed; but same dual-stack application may work well in a Dual-stack Lite and NAT64/DNS64 mixing environment.**

Keywords—IPv6 migration; Dual-stack Lite; NAT64/DNS64; AplusP.

I. INTRODUCTION

The IANA pool for global public IPv4 address allocation is forecasted to exhaust by mid-2011. Yet IPv4-only legacies are ubiquitous crossing telecom infrastructure. As IPv6 and IPv4 are incompatible protocols, IPv6 could not replace IPv4 in order to solve the public IPv4 exhaustion problem immediately. Instead, both protocols will co-exist for a long period of time. The common thinking for more than 10 years has been that the transition to IPv6 will be based solely on the dual stack model until IPv6 takes over IPv4 before we ran out of IPv4. However, this has not happened. The IANA free pool of IPv4 addresses will be depleted soon, well before sufficient IPv6 deployment will exist. As a result, many IPv4 services have to continue to be provided even under severely limited address space. As a result, saving IPv4 address is one of concerns for IPv6 transition solutions. Dual-stack Lite [1], AplusP [2] and NAT64/DNS64 [3], described in this section, are transition approaches that address IPv6 introduction as well as IPv4 address sharing.

A. IPv6 transition Solutions

- Dual-stack Lite

The Dual-stack Lite technology [1] is intended for maintaining connectivity to legacy IPv4 devices and networks when service provider networks make a transition to IPv6-only deployments after the exhaustion of the IPv4 address space.

Dual-stack Lite enables a broadband service provider to share IPv4 addresses among customers while migrating to IPv6 by combining two well-known technologies: IP in IP tunnel and NAT. The principle is simple, 1) moving the current NAT performed on the Home Gateway (HGW) to Carrier Grade NAT; 2) IPv4 traffic are transported over IPv6 access network by IPv4-in-IPv6 softwires, an IP in IP tunnel defined in RFC5571[4]. Dual-stack Lite specification introduces two new terms: the DS-lite Basic Bridging Broad Band element (B4) and the DS-lite Address Family Transition Router element (AFTR). A B4 element is a function implemented on a dual-stack capable node, either a HGW or a directly connected device that creates a tunnel to an AFTR. An AFTR element is the combination of an IPv4-in-IPv6 tunnel end-point and an IPv4-IPv4 NAT implemented on the same node.

As illustrated in Figure 1, each HGW has only IPv6 access, yet many customers are still configured with RFC1918 [5] private addresses. Therefore the traffic generated by terminals is tunneled by the B4 element to the AFTR via an IPv4-in-IPv6 softwire, where B4 element is acting as Softwire Initiator (SI) and the AFTR is acting as Softwire Concentrator (SC). After de-capsulation, the AFTR then translate RFC1918 private addresses realm to public IPv4 address realm. Per subscribers tunnel endpoints ID are identified by AFTR to distinguish RFC1918 private address space per subscriber.

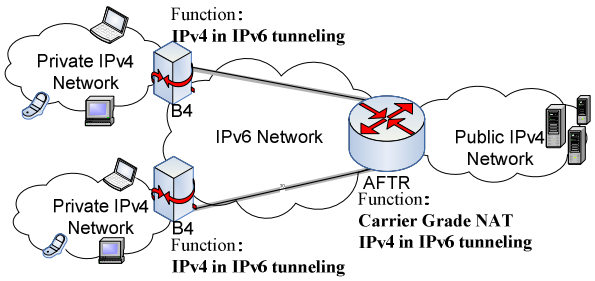

Figure 1. Dual Stack-Lite

- AplusP

The principle in AplusP [2] is straightforward; 16 bits stolen from TCP/UDP port field are attached to the IPv4 address to identify different customers that sharing the same public IP address. Hence, multiple HGW share a common global IPv4 address, but with separate, non-overlapping, port ranges. Each HGW can use the address as if it were its own public address, except that only a limited port range is available to be used. An IPv6 address derived from a pre-assigned public

 ISBN: 978-1-61208-133-5

IPv4 address plus a specified range of ports are allocated to each HGW; as a result, Port Range Router (PRR) can route incoming traffic to destination HGW according to a destination IPv6 address generated from destination IPv4 address and port fields of the IPv4 packet header.

As demonstrated in Figure 2, AplusP uses the same two technologies as DS-lite: IP in IP tunnel and NAT, but in different ways. First, the NAT is performed on the HGW rather than on Carrier Grade and the HGW NAT should ONLY use the restricted source port range. Second, although IPv4 traffic are also transported over IPv6 access network by IPv4-in-IPv6 tunnels as the DS-lite does, the AplusP HWG should pre-assigned an IPv6 address that is derived from a pre-assigned public IPv4 address plus a specified port range so that Port Range Router can route incoming traffic to proper HGW according to a destination IPv4 address and port.

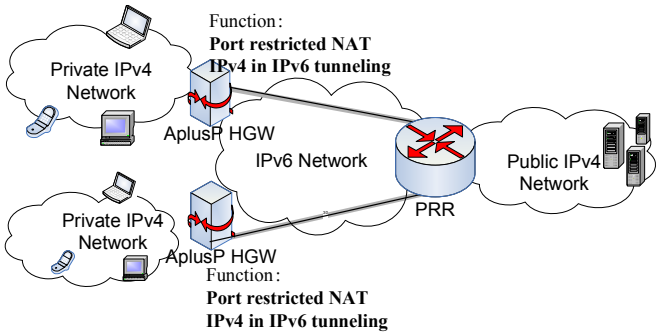

Figure 2. AplusP

- NAT64/DNS64

It has been reached a consensus that both networks coexist until IPv6 takes over IPv4. However, IPv6 growth has been much slower than anticipated. Therefore, IPv6-only deployments face a challenge of communicating with the predominantly IPv4-only rest of the world. Likewise, a similar problem is encountered when legacy IPv4-only devices need to reach the IPv6 Internet. One initial proposal was NAT-PT RFC2766 [6]. However, it has been declared obsolete in RFC4966 [7] due to its various issues. Still, to address this challenge, IPv4/IPv6 translation continued to be a major focus of interest at IETF. Recently, IETF BEHAVE working group has been working on IPv4/IPv6 translation solution and has resulted in several draft documents. The general framework for IPv4/IPv6 translation is described in [8], which also explains the background of the problem and some use cases. NAT64/DNS64 [3], describes a stateful IPv6-to-IPv4 NAT translation which allows IPv6-only clients to talk to IPv4 servers using unicast UDP, TCP, or ICMP. The public IPv4 address can be shared among several IPv6-only clients. Used in conjunction with DNS64, which is a mechanism for synthesizing AAAA resource records (RR) from A RR and NAT64's prefix, NAT64 requires no changes in the IPv6 client or the IPv4 server.

The Figure 3 illustrates NAT64/DNS64 principle and a home network use case of NAT64/DNS64. When a host e.g., Host-1 in IPv6 network wants to talk to a host e.g., Host-2 in IPv4 network, NAT64 together with DNS64, which works as DNS proxy and derive AAAA RR from A RR, translate IPv6 TCP, UDP and ICMP to IPv4. DNS64 can be either standalone device or embedded within NAT64. Dotted line describes a use scenario, where IPv6-only terminals in a home network need to contact with IPv4 peer. In this scenario, no changes are required on HWG and only IPv6 Router Advertisement (RA) or DHCPv6 are expected to offer automatically configuration for IPv6 devices.

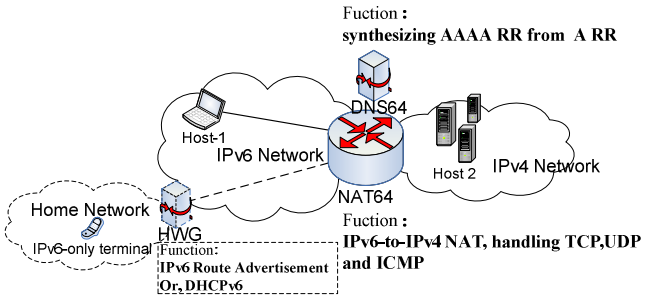

Figure 3. NAT64/DNS64 and a use case

IETF Softwire working group has submit Dual-stack Lite [1] to the IESG, which is responsible for technical management of IETF activities and the Internet standards process, to make it be considered as a Proposed Standard. While NAT64/DNS64 [3l] has been approved by IESG and been sent to RFC Ed Queue. Recently, many vendors have claimed that Dual-stack Lite and NAT64/DNS64 approaches have been included in their product roadmap; for example: Cisco's Carrier-Grade IPv6 (CGv6) Solution and HUAWEI's Carrier-grade NAT (CGN) Solution.

B. Motivation and Objectives

IPv4 address sharing is an important mechanism during transition phase. Application behaviors in terms of port consumption not only impact the deployment factor (i.e., port range size) for AplusP solution but also play an important role in determining the port usage per customer on AFTR for Dual-Stack Lite. During our study, a concern that NAT64 may break existing popular applications has been arising. Hence, it is of interest to test application's NAT64 compatibility. In addition, since solutions addressing different use cases which may co-exist in the same subscribers' network, it is essential for network providers to understand potential issues and how applications cope with multiple transition solutions. Therefore, we have implemented three transition solutions in an integrated platform on which application behaviors were tested.

C. Orgnaization

The rest of this paper is organized as follow. Section II introduces related works; Section III presents our platform in which three transition solutions are implemented and integrated. The test results are discussed in Section IV. Finally, this paper is concluded in Section V.

II. RELATED WORK

A. Opensource AFTR Implementation

Internet Systems consortium (ISC)'s AFTR [9] which is available for free download under the terms of the ISC License

 ISBN: 978-1-61208-133-5

(a BSD style license), implements a Dual-Stack Lite AFTR as described in [1].

A Dual-Stack Lite deployment includes at least one AFTR in the ISP's network core, and one B4, which is the IPv4 default router for all hosts behind it and customer-side tunnel initiator. For testing and demonstration purposes, despite that B4 functionality can also be built into general-purpose computers (e.g., in FreeBSD or Linux), ISC's AFTR has used a Linksys WRT54GL running OpenWrt [10], an embedded Linux distribution for home gateways, and released a WRT54G prebuilt images which is prebuilt with functionalities that make a B4 set up an IPv4-in-IPv6 tunnel with AFTR.

As Dual-stack Lite is under the standardization process in the softwire working group of the IETF and there may be changes to the specification before it is finalized as an RFC, ISC AFTR have been actively tracking the current specification. As such, ISC AFTR considers itself as a work in progress, for testing and experimentation only. It is an open source implementation meant to promote the development of open standards for IPv4-to-IPv6 transition technology.

B. *Open source NAT64/DNS64 Implementation*

Be funded by the NLnet Foundation and Viagénie, Ecdysis [11], has developed an open-source implementation of a NAT64 gateway to run on open-source operating systems such as Linux and BSD. The gateway is comprised of two separated modules: the DNS ALG and the IP translator. The DNS ALG is implemented in two DNS open-source server: Unbound and Bind. The IP translator is implemented in Linux as kernel module using Netfilter facilities and in openBSD as a modification of Packet Filter (PF).

C. *AplusP Implementation*

We had implemented both Aplusp HGW and Port Range Router (PRR) on a Linux platform [12].

For Aplusp HGW, using Netfilter framework, the IPv4 port restricted NAT operation performed by CPE was implemented by simply setting rules through iptables tool on Linux. After the NAT operation on the CPE, the NATed IPv4 packets were sent to a TUN interface which represented as a virtual network interface in Linux and enabled with IPv4-in-IPv6 encapsulation/decapsulation functions developed by us.

PRR, located in the interconnection point of the IPv6 network and IPv4 network, is implemented with two main functions: 1) IPv4-in-IPv6 encapsulation/decapsulation; 2) destination port based routing function, which is for the IPv4 traffic originated from the IPv4 Internet and destined to the shared IPv4 address realm of the operator. Likewise, TUN driver is also used in PRR to achieve function 1). Function 2) is realized by pre-assigning an IPv6 prefix that maps from IPv4 address and port range to each CPE, and generating IPv6 destination address according to IPv4 destination address and port. To facilitate test and experiment on AplusP solution, recently, we are considering release this AplusP implementation under open source license.

III. IPv6 TRANSITION SOLUTION IMPLEMENTATION

A. *Overview of Implementation*

Based on the related work stated in previous section, first, we integrated all of these three IPv6 transition solutions: AplusP, Dual-stack Lite and NAT64 into a realistic ADSL access environment, which consists of HGW, DSLAM, PPPoE server and DHCPv6 server. As illustrated in Figure 4, we customized two types of HGWs, running OpenWrt, on Linksys WRT54GL: a) AplusP enabled home gateway has been uploaded with our AplusP HGW functions and configured with PPPoE client and DHCPv6 client; b) Dual-stack Lite/NAT64 enabled home gateway, providing both Dual-stack Lite and NAT64 solution for the same subscriber, has been configured with IPv4-in-IPv6 tunneling, PPPoE client and DHCPv6 client for Dual-stack Lite; and IPv6 Route Advertisement (RA) for NAT64. The IPv6 provisioning to HGWs is via IPv6 over PPPoE provided by a PPPoE server, where a DHCPv6 server is co-located.

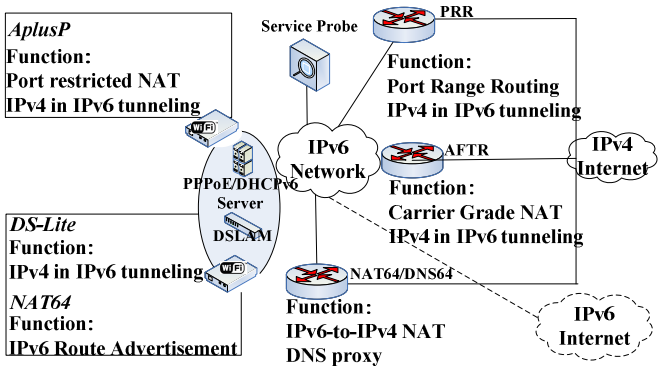

Figure 4. IPv6 transition solutions mock up

Currently, we only have IPv4 Internet access and in future we will also have access to the IPv6 Internet which is illustrated by dotted line in Figure 4.

Because new DHCPv6 options for AplusP and Dual-Stack Lite have not been standardized yet, we used user defined options for mockup purpose. The DHCPv6 server was configured to convey different set of DHCPv6 options, some of which are user defined, to AplusP HGW and Dual-stack Lite/NAT64 HGW separately. The user defined DHCPv6 options are shown in Figure 5.

```
#AplusP DHCPv6 options
option dhcp6.gateway code 54 = ip6-address;
option dhcp6.ipv4 code 55 = ip-address;
option dhcp6.port code 56 = unsigned integer 16;
option dhcp6.range code 57 = unsigned integer 16;

#Dual-Stack Lite DHCPv6 options
option dhcp6.softwire code 58 = ip6-address;
option dhcp6.name-servers code 59 = ip6-address;
option dhcp6.pubadd code 91 = ip6-address;
option dhcp6.radvd code 92 = ip6-address;
option dhcp6.defgateway code 90 = ip6-address;
```

Figure 5. DHCPv6 options for AplusP and Dual-Stack Lite

B. *Service Probe in AplusP*

Besides PRR, AFTR and NAT64/DNS64, we also developed and deployed a Service Probe in our IPv6 network, which use IPv6 TCP socket to ask AplusP HGW for NAT session usage, and store AplusP NAT statistics in a Mysql

database to further analyze application behaviors in terms of port and session consumptions. The detailed test results and analysis are presented in the next section.

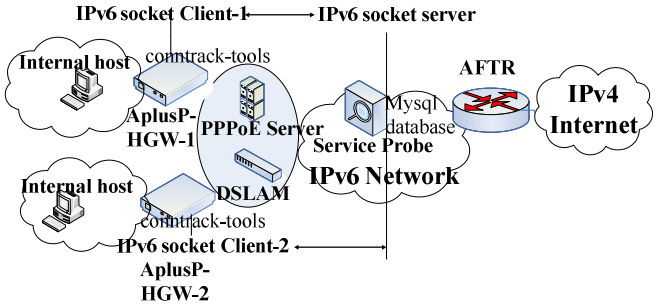

Figure 6. Service Probe and AplusP HGW

To implement service probe, we have configured conntrack-tools [13], which is the module that provides stateful packet inspection for iptables, on the AplusP HGW to collect statistics of the iptables NAT usage. All the statistics of AplusP HGW NAT are then sent via IPv6 socket to Service Probe which is responsible for further storage and analysis of application behaviors, more specifically, TCP/UDP ports and sessions consumption during communication. Service Probe socket is designed by I/O multiplexing approach so that it can monitor multiple AplusP HGWs at the same time, e.g., in our test bed there are two HGW as shown in the Figure 6. Database design on Service Probe is simple and shown in Figure 7 and Figure 8. The first table in the Mysql database stores key information of all NAT sessions, received from AplusP HGW, including (si, sp, di, dp, protocol, msi, msp, sla, ela, status, pr) where the end time which is N/A until the session is expired, and the status of the session which is either active when the session is valid or history when the session is expired. By scanning the first table, for per internal host, a second table is instanced to keep tracking and storing port numbers and session numbers that this internal client is using in an every second basis.

si	source address
sp	source port
di	destination address
dp	destination port
proto	the protocol
msi	mapped source address
msp	mapped source port
sla	the start time of a session
ela	the end time of a session
status	either active or history session
pr	port rage

Figure 7. Fields description of Service Probe's first table

ip	IP address of a client
pn	The port number of a client used
sn	The session number of a client used
ttime	The current time
pr	Port range

Figure 8. Fields description of Service Probe's second table

C. *Dual-stack Lite and NAT64 implementation*

Introducing NAT64 may bring impacts, for instance some applications may break due to incompatible with NAT64. Yet with both Dual-stack Lite and NAT64 enabled network, it was not clear that how application may behave in the multiple transition solution enabled network. For example, we were not certain whether apps choose IPv6 over IPv4 or if they choose both. Therefore we have tested apps' compatibility with NAT64 as well as apps' compatibility with two transition solution enabled network. Furthermore, we have observed apps' behaviors in terms of IP preference, more specifically, how applications, in a both Dual-stack Lite and NAT64 enabled subscriber network, deal with AAAA and A DNS record and which IP version protocol (IPv4 or IPv6) is preferred to initiate communication. To do so, a Dnsmasq (a DNS forwarder), whose upstream DNS server is configured with DNS64, was installed in the B4 element as shown in Figure 9. DNS64 returns both AAAA and A RRs to Dnsmasq which in turn forwards the responses to the host behind B4. Since we do not have native IPv6 access yet, AAAA RRs returned by DNS64 are generated from A RRs and NAT64's prefix instead of native AAAA RRs.

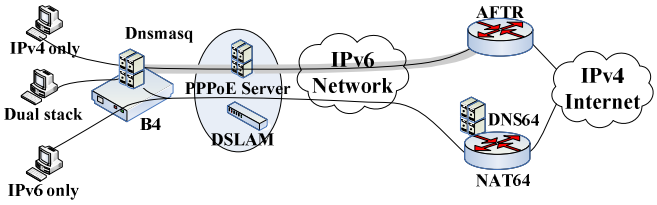

Figure 9. Dual-Stack and NAT64

IV. TEST RESULTS AND ANALYSIS

A. *Application behaviors in interms of port/session consumptions*

We tested popular applications, such as news website, video website, skype, BitTorrent and GoogleEarth, to investigate how many ports and sessions they are costing on the NAT mapping table dynamically from the first NAT bindings being established to the last one being destroyed. All the figures in this sub-section are derived from the Mysql database described in sub-section B of the previous section.

As illustrated in Figure 10, when open a news website (e.g., [15]) that often contains a number of images and flashes, it takes up to four minutes from the first NAT binding being established to the last one being destroyed. During the four minutes new NAT bindings are established while the old ones are expiring. For IE, the port consumption dramatically rose and reached the peak of 20 ports at the 18th seconds and then decreased gradually to zero at the 200th seconds. While for firefox, after opened a dozen of ports at the beginning, it then gradually increased to 25 until the 120th second, and finally dropped gradually to zero until the 240th second. It is evident that even if visit the same website, port consumption varies from web browser to web browser.

 ISBN: 978-1-61208-133-5

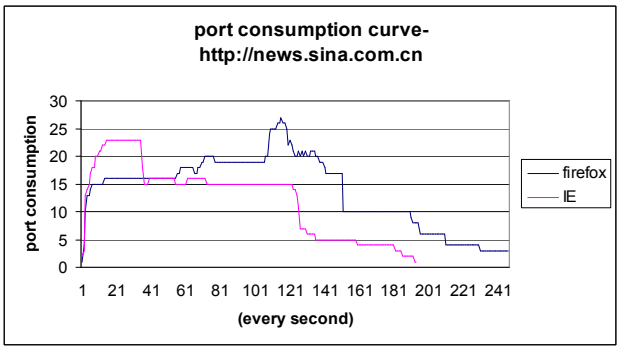

Figure 10. Comparision of port consumption between firefox and IE

Figure 11 shows same evidence that firefox consumes more ports than IE when open a video website which cost up to 80 ports during browsing its main page.

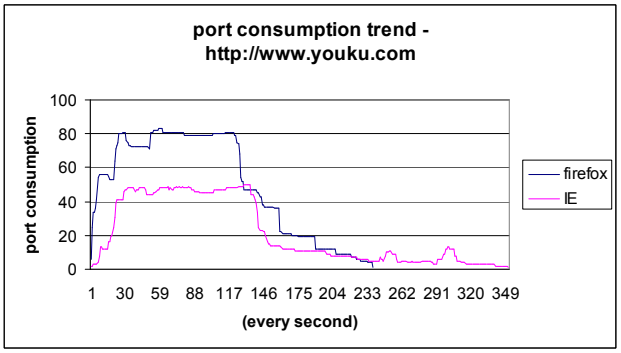

Figure 11. Sampled ports from DNS randomness test in Scenario B

Port consumption comparison among different applications, including BitTorrent, Google Earth, skype and using firefox to visit news website and video website, is demonstrated in the Figure 12. BitTorrent has constantly occupied hundreds of ports while downloading and dominated in the port consumption. Firefox has consumed dozens of ports for about two minutes and ranked second after BitTorrent, while others merely cost less than 25 ports during a whole communication process for each.

The session consumption comparison among the same set of applications are also illustrated in Figure 13. Unlike other apps which consume similar amount of sessions as ports, BitTorrent established five hundreds of sessions even though the port consumption was relatively low (under a hundred) in the first minute of the communication, because when BitTorrent initiates a downloading it first uses the same source port to connect to the different destinations (destination IP and port) therefore one source port multiplexing different sessions. Besides, Skype is another example that uses one source port to multiplex different sessions thereby saving source port consumptions on NAT.

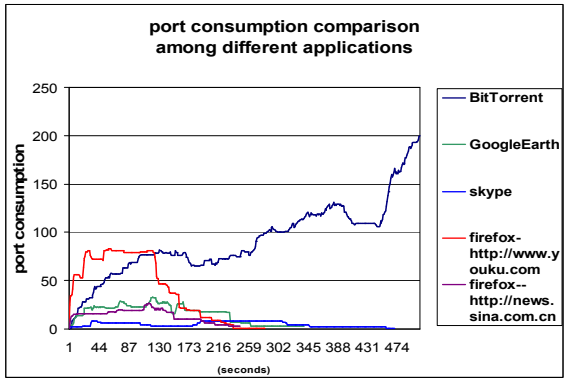

Figure 12. Port consumption comparison among different apps

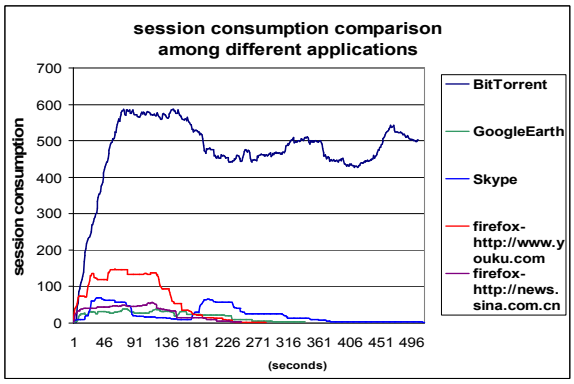

Figure 13. Session consumption comparison among different apps

Furthermore, we have tested and compared mobile apps and PC apps. The test results shown in Figure 14 and Figure 15 indicates that even the same app, either web-browser chrome or Google Earth, the mobile release - Android chrome and Android Google Earth consumed fewer ports than the PC release – Windows chrome and Windows Google Earth respectively.

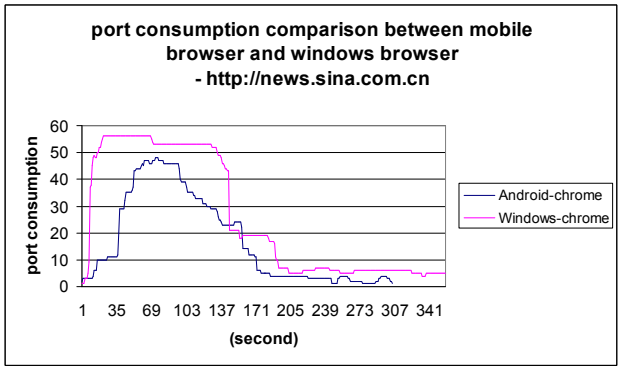

Figure 14. Comparison between mobile and Windows broswer

 ISBN: 978-1-61208-133-5

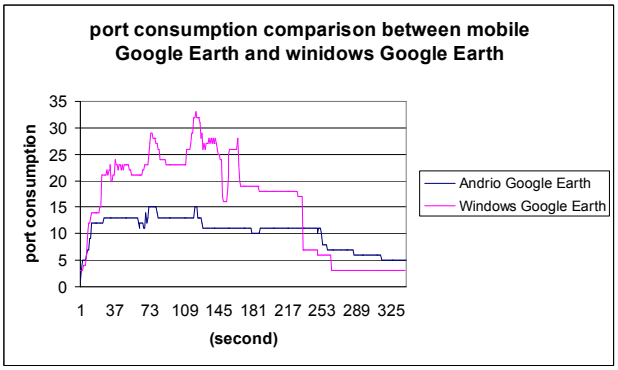

Figure 15. Comparison between mobile and Windows GoogleEarth

B. *Application compatibility with NAT64*

Tested Apps	Compatible	Non-compatible
Firefox (Non-vedio) v3.6.12	√	
video website		√
IE (Non-vedio) v6.0	√	
Skype v5.0	√	
Google Earth v5.2.1	√	
Live Messenger 2009		√
QQ 2010		√
uTorrent v2.2		√
BitComet v1.23		√

Figure 16. Apps compatibility with NAT64

Test results of apps' compatibility with NAT64 are listed in the Figure 16. Vesting video website, Live Messenger, QQ, uTorrent and BitComet break in a NAT64 only network, while firefox, IE, Skype and Google Earth work well with NAT64.

C. *Application compatibility with NAT64 and Dual-stack Lite mixing network*

Besides investigation on NAT64 solution, we have tested if the both Dual-stack Lite and NAT64 enabled network has influence on apps. In Figure 17, the test results show that all of them work well without configuration change, manual restart or other human interference.

Tested Apps	Compatible
Firefox (Non-vedio) v3.6.12	√
video website	√
IE (Non-vedio) v6.0	√
Skype v5.0	√
Google Earth v5.2.1	√
Live Messenger 2009	√
QQ 2010	√
uTorrent v2.2	√
BitComet v1.23	√

Figure 17. Apps compatibility with both NAT64 and DS-Lite enabled environment

D. *Application behaviors in terms of IP version preference*

Regarding both Dual-stack Lite and NAT64 enabled subscriber network, we further tested and analyzed whether IPv4 or IPv6 is preferred to initiate the communication. It has been shown that all the applications that we investigated issued both A and AAAA DNS query, and except QQ (an Instance Messenger) and BitComet (a BT Client) completely ignored AAAA RR, all other applications made use of both IPv6 and IPv4 to communicate with peers/servers. IPv6 usage portion depends on apps and use cases, some of which use all IPv6 while others used all IPv4; some of which use major in IPv6 while others used major in IPv4. As a result, as illustrated in Figure 18, we classified apps into five categories: 1) all IPv6, 2) major IPv6, 3) half/half, 4) major IPv4 and 5) all IPv4. Again, as we do not have native IPv6 access, all the IPv6 communication was talking to IPv4 servers/peers via NAT64.

- all IPv6

For web browsers, AAAA records have higher priority and when visit non-video website, both firefox and IE used IPv6 to talk (to IPv4 server) through NAT64.

- major IPv6

Skype and Google Earth used AAAA to initiate IPv6 connections (to IPv4 server) via NAT64. Yet, according to our captured packets (wireshark), we still found that there were a few IPv4 connections.

- half/half

When firefox and IE were used to visit a video website (e.g., [15]), the main pages were downloaded through IPv6 via NAT64 and after IPv6 video downloading failure due to NAT64 traversal failure, firefox and IE then requested IPv4 and vedio downloading was done via AFTR.

Tested Apps	All IPv6	Major IPv6	Half/ half	Major IPv4	All IPv4
Firefox (Non-vedio) v3.6.12	√				
video website			√		
IE (Non-vedio) v6.0	√				
Skype v5.0		√			
Google Earth v5.2.1		√			
Live Messenger 2009				√	
QQ 2010					√
uTorrent v2.2				√	
BitComet v1.23					√

Figure 18. Apps classified by IPv6 usage portion

 ISBN: 978-1-61208-133-5

- major IPv4

During login and authentication phase, Live Messenger were using IPv6, but after that, because the IPv6 client did not send the same application layer message as the IPv4 client, the IPv6 client (behind NAT64) failed to get reply from IPv4 server. Then, Live messenger automatically switched to IPv4 by which all the rest of communication were done. What we have learned from this case is that the application layer should be IP version agnostic in order to decrease impacts introduced by IPv6 transition solutions. uTorrent(a BT client) is another instance that used IPv6 for login/authentication but IPv4 for the data exchange, for IPv6 peer was not able to talk to IPv4 peer.

- All IPv4

Although QQ and BitComet issued both A and AAAA quires, they completely ignored AAAA RR and only used IPv4 for communication.

V. CONCLUSION

It is likely to have multiple transition approaches in the same subscriber network. Therefore, we have implemented AplusP, Dual-stack Lite and NAT64/DNS64 in an integrated platform and investigated application behaviors in this platform. Firstly, port/session consumption on NAT that impacts on the deployment factors for both AplusP and Dual-stack Lite has been tested. Secondly, application's NAT64 compatibility is presented. Results of application tests indicate that dual-stack application may break either due to IP version dependent of application layer or NAT64 traversal difficulties if only NAT64/DNS64 is deployed; yet same dual-stack application may work well in a Dual-stack Lite and NAT64 mixing environment.

REFERENCES

[1] Durand, A., "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", draft-ietf-softwire-dual-stack-lite-06 (work in progress), August 11, 2010.

[2] Bush, R., "The A+P Approach to the IPv4 Address Shortage", draft-ymbk-aplusp-08 (work in progress), January 5, 2011.

[3] M. Bagnulo and P. Matthews., " Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in progress), July 10, 2010.

[4] RFC5571, B. Storer, "Softwire Hub and Spoke Deployment Framework with Layer Two Tunneling Protocol Version 2 (L2TPv2)", June 2009.

[5] RFC1918, Y. Rekhter,"Address Allocation for Private Internets", February 1996.

[6] RFC2766, G. Tsirtsis, "Network Address Translation - Protocol Translation (NAT-PT)", February 2000.

[7] RFC4966, C. Aoun, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", July 2007.

[8] F. Baker, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework, August 17, 2010.

[9] Internet Systems Consortium, https://www.isc.org/software/aftr, March, 2011

[10] OpenWrt, a Linux distribution for embedded devices, http://openwrt.org/, March, 2011.

[11] Ecdysis: open-source implementation of a NAT64 gateway , http://ecdysis.viagenie.ca/, March, 2011.

[12] Z. Xiaoyu, X. DENG, "Implementing Public IPv4 Sharing in IPv6 Environment", April 2010.

[13] conntrack-tools, Connection tracking userspace tools for Linux, http://conntrack-tools.netfilter.org/, March, 2011.

[14] http://www.sina.com.cn, March, 2011.

[15] http://www.youku.com, March, 2011.

 ISBN: 978-1-61208-133-5