An Optimized Port Allocation Mechanism in the Context of A+P for Public IPv4 Address Sharing

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

***Abstract*—the IANA free pool of IPv4 addresses will be exhausted soon, how to use scarce IPv4 public addresses more efficiently while migrating to IPv6 is a challenge. A+P is recommended as a complementary method to Dual-stack Lite which aims at address public IPv4 address sharing problem in the context of IPv6 migration. Since A+P suffers from inflexible port allocation, this paper introduces an optimized A+P port allocation mechanism which allows customers negotiate IP-addresses of desired sharing ratios on their requirement. Moreover it enables A+P NAT using random source port selection algorithm which significantly improves security by preventing attacker's easy guessing the five-tuple. The test result shows that this mechanism enables great randomness of source ports selection behavior on A+P NAT.**

Keywords—IPv6 migration; Dual-stack Lite; A+P; Port randomization.

I. INTRODUCTION

The IANA pool for global public IPv4 address allocation is forecasted to exhaust by mid-2011.And IPv4-only legacies are ubiquitous crossing telecom infrastructure. Since 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. With public IPv4 address sharing solutions, higher utilization ratio of available public IPv4 addresses can be achieved. These solutions can be deployed before the majority of the Internet becomes IPv6-capable and most communications could be done through IPv6.

A. IPv4 Address Sharing Solutions

Several solutions have been proposed in IETF to address IPv4 address shortage while migrating to IPv6, such as NAT444 [1], A+P [2], Dual-stack Lite [3]. Recently, IETF reached a consensus on Dual-Stack Lite as a promising solution. It provides the broad band service provider a scalable and easy way to introduce IPv6 while keeping IPv4 reachability to its customers. In Dual-Stack Lite, IPv4 addresses among customers are shared using two technologies: IP in IP (IPv4-in-IPv6) and NAT. A+P is similar to Dual-Stack Lite, the difference is that A+P NAT locates at the customer premises, while Dual-Stack Lite NAT locates at the carrier network.

Many applications require ALG to work through a NAT. At the same time, more and more applications expect incoming connections, such as peer-to-peer ones. Making sure those subscriber-provided services working properly in a Dual-stack Lite environment is important. Unfortunately, service providers are not in the position of provisioning such applications and ALGs. In this case, in [3], A+P is recommended as a complementary method to Dual-stack Lite in order to deal with the subscriber-provided services' ALG issues, which would break some subscriber-provided services if ALG issues are not well treated. Reserving certain ports under the control of customers is one way to enable the Customer Premises Equipment (CPE) A+P NAT to process this kind of traffic. Figure 1 shows an example: Ports 5002-5004 are reserved for A+P NAT; Incoming packet that falls into the A+P ports range bypasses the Dual-stack Lite Carrier Grade NAT (CGN), and directly is sent to the tunnel endpoint, an A+P aware CPE, from the Address Family Transition Router (AFTR). CPE then locally NAT the packet to internal hosts, otherwise the incoming packet will traverse the AFTR's NAT.

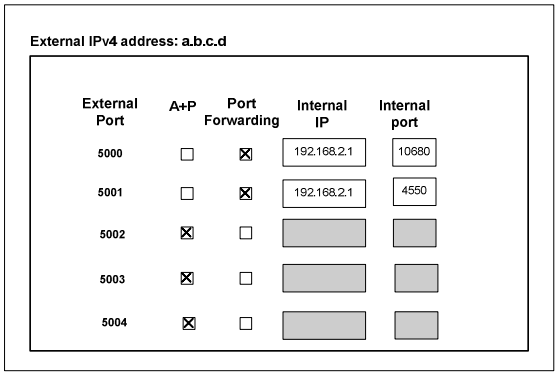

Figure 1. ISP portal address & port control table

B. Motivation and Objectives

Firstly, dynamic port assignment is used in Dual-stack Lite mode to maximize the address sharing ratio. On the other hand,

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-133-5

A+P mode allocates ports in a cookie-cutter fashion, i.e., a range of ports are pre-allocated to each CPE. Concerns have been raised when public IPv4 addresses are shared among a large mount of CPE, while only a limited N number of TCP or UDP port numbers are available per CPE in average. In fact, pre-allocating N ports is not encouraged according to several service providers' report: the average number of connections per customer is the single digit, while thousands or tens of thousands of ports could be used in a peak by any single customer browsing a number of AJAX/Web 2.0 sites. If a smaller number of ports per CPE (N in the hundreds) are allocated, it is expected that customer's applications could be broken in a random way over time. If a large number of ports per CPE are allocated (N in a few thousands), the address sharing ratio will be decreased. Furthermore, customers may require different amount of ports, for example, enterprise customers may expect more ports to support simultaneous sessions; some customers have many terminals connected to CPE which may require more ports.

Secondly, a number of "blind" attacks can be performed against the TCP and similar protocols by identifying the transport protocol instance, i.e., the five-tuple (Protocol, Source Address, Destination Address, Source Port, Port). Since A+P pre-allocates N (N<65,536, usually less than several thousands in order to achieve a large sharing ratio more than a single digit) ports to CPE, it is more easy to guess the five-tuple and in turn increase the probability of successful attacks.

This paper proposes a optimized port allocation mechanism for A+P which aims at addressing those two defects of A+P by two efforts, 1) providing customer oriented differentiated services for A+P to allow customer negotiate IP-addresses of desired sharing ratios based on their requirement; 2) providing a source port randomization algorithm to achieve better security by preventing attacker's easy guessing the five-tuple.

C. Orgnaization

The rest of this paper is organized as follows. Section II introduces related works; Section III presents our proposed optimized port allocation mechanism. The simulation results are discussed in Section IV. Finally, this paper is concluded in Section V.

II. RELATED WORK

Several methods have been proposed for allocating A+P parameters. In [4], DHCPv4 Options for allocating port restricted public IPv4 address and a range of ports are defined. Two IPCP Options, Port Range Value Option and Port Range Mask Option to convey one range of ports (either contiguous or not contiguous) pertaining to a given IP address have been discussed in [5]. These two IPCP Configuration Options provide a way to negotiate the Port Range to be used on the customer Premises. The sender can use the Configure-Request message to carry request which Port Range associated with a given IP address is desired, or to request the peer providing the configuration, the peer then can provide this information by NAKing the option, and returning a valid Port Range associated with an IP address.

Both of these methods propose A+P port allocation mechanism and negotiation process, either by using DHCP semantics or by PPP IPCP negotiation. Neither of them addresses the problem described in Section I.

III. OPTIMIZED A+P PORT ALLOCATION MECHANISM

Firstly, a mechanism was designed to allow service provider provisioning the differentiated qualities of service by allocating different sharing ratio IP-addresses to different customer. As customer gets better service with lower sharing ratio IP-address, the one demanding better service pays more for the lower sharing ratio IP-address. The operator could configure IP-addresses pools with different service levels depends on different sharing ratios. As illustrated in Table I, it shows an example seven service level IP-addresses pools was configured according to seven sharing ratios. With sharing ratios varies from 1 to 64, which means available ports for each customer varies from 65,536 to 64, the service level decrease from level 0 to level 6. Level 0 provides one unshared global IP-address to customer as nowadays operator does. Each customer could request different service level IP-address according to their requirements on quality of service, which depends on the sharing ratio, the lower sharing ratio, the higher quality of service with higher price.

TABLE I. AN EXAMPLE OF SERVICE LEVEL ADDRESS POOLS

Service level address pools	Sharing ratio	Available ports
Level 6 address pool	64	1024
Level 5 address pool	32	2048
Level 4 address pool	16	4096
Level 3 address pool	8	8192
Level 2 address pool	4	16384
Level 1 address pool	2	32768
Level 0 address pool	1	65,536

Secondly, to provide a way for customer to generate random ports while guaranteeing them in customer restricted port range, the core idea is simple: Choosing M bits to from a customer ID bits for a set of customers which sharing the same IP-address, and then identifying the customers in the same set by allocating unique M bits customer ID values. Hence the M bits customer ID could guarantee the ports inside the customer restricted port range. With regard to each shared IP-address, as its sharing ratio is given, the number of customer ID bits M is

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-133-5

decided by $\log_2^{Sharingratio}$. In order to facilitate the reuse of existing Port Randomization algorithms, two parameters are derived, customer ID pattern and customer ID value, customer ID pattern is derived from setting the customer ID bits to '1' and the left bits to '0', customer ID value is derived from setting the Customer ID bits to a unique value allocated from the operator side and the left bits to '1'. An example is shown in Figure 2, a Level 6 address pool of sharing ratio 64 for a sharing IP-address-A is selected, and then 6 bits are chosen as Customer ID bits, which are the 3rd, 4th, 7th, 9th, 10th, 12th bit. All the customers sharing IP-address-A get "0000101101001100" as customer ID pattern, and each of them get a unique customer ID value. As shown in Figure 3, take the customer#5 for example, for a random generated port, just take a bitwise AND operation with customer ID pattern and then take a bitwise OR operation with customer ID value of customer#5, the result port would be inside the customer#5 restricted port range. Hence it's easy for customer NAT reusing the port randomization algorithms referred in [6], it could be done by slight modification to the existing port randomization algorithms. Take the Algorithm 1: Simple port randomization algorithm [6] for example, the Figure 4 shows the A+P simple port randomization algorithm, only one line code was inserted to the original simple port randomization algorithm.

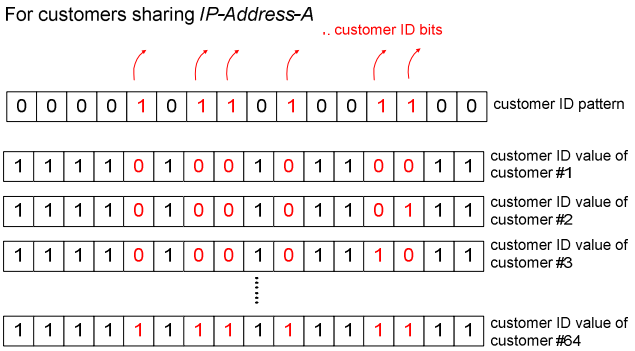

Figure 2. Customer ID pattern and customer ID value

To ensure the customer NAT could generate as random ports as possible to prevent attackers, three heuristic rules to choose M bits of Customer ID are proposed:

1) To avoid allocate a range of continuous ports to customer, the location of M bits for the Customer ID Pattern should not take place in the most significant bits.

2) To avoid allocating only even ports to customers with the least significant bit '0' or only odd ports to customers with the least significant bit '1', M bits Customer ID Pattern should not involve the least significant bit.

3) To avoid allocating a regular range of ports to customer, the location of M bits for the Customer ID Pattern should not take place in the continuous bits.

Pseudocode for Customer ID Pattern bits chosen is shown in the Figure 5.

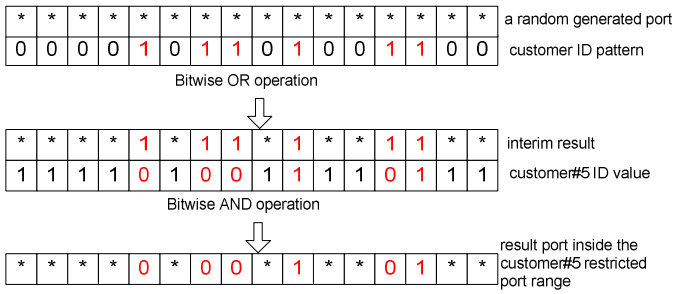

Figure 3. Calculate port inside the customer restricted port range

```
/* A+P Ephemeral port selection function */

num_ephemeral = max_ephemeral - min_ephemeral + 1;
next_ephemeral = min_ephemeral + (random() %
num_ephemeral);
next_ephemeral_in_range = next_ephemeral ||
customer_ID_pattern & customer_ID_value
count = num_ephemeral;
do {
if(five-tuple is unique)
return next_ephemeral_in_range;
if (next_ephemeral == max_ephemeral) {
next_ephemeral = min_ephemeral;
} else {
next_ephemeral++;
}
count--;
} while (count > 0);
```

Figure 4. A+P simple port randomization algorithm

Customer could get IP-address in desired sharing ratio by negotiating Customer ID pattern with operator, the negotiation could be done by several ways, either by negotiating DHCPv4 Option for allocating port restricted public IPv4 address defined in [4], or by negotiating PPP IPCP Options defined in [5] after Softwire [7] is established and IPCP reaches the Opened state. The negotiation process is out of scope.

Note that customers sharing the same IP address share the same customer ID pattern; however customers with different IP addresses, no matter if they are using the same sharing ratio, are using different customer ID patterns which is granted by random customer ID choosing algorithm.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-133-5

```
/* Heuristic algorithm for Customer ID bits chosen  */

M = log2(SHARING_RATIO);
Genbits:
/*Guarantee Heuristic rule 1 and  Heuristic rule 2*/
for (i = 0; i < M;)
{
bits[i] = GenRandomBit();
if(bits[i] == 0 || bit[i] = =15); else i++;
}
/*Sort array in decreasing order in order to match the
condition for rule 3*/
Sort(bits);
/*Guarantee Heuristic rule 3*/
for (i = 1; i < M;)
{
if(bits[i-1] = = bits[i]+1) i++;
else break;
}
if(i= =M) goto Genbits;
return bits;
```

Figure 5. Customer ID choosing algorithm

IV. EVALUATION

A. TCP/UDP Source Port Considerations

Since a successful attack against the TCP or UDP requires the attacker to have knowledge of a valid five-tuple (protocol, source IP address, source port, destination IP address, and destination port). The protocol, source and destination IP addresses are obvious as they are the specific services the attacker will be spoofing. The destination port is also obvious, the attacking aims at well-known services, which announce well-known ports to public. The only difficult part of guessing this is the TCP/UDP source port, since it different for each new TCP/UDP session. As for TCP RESET attack, the attacker could assume the destination port of 179 for BGP, as for Domain Name System (DNS) cache poisoning attack, it could assume the destination port of 53 (the port number IANA has assigned for DNS). For an attacker, additional requirement of a correct source port would increase the difficulty of the attack by a factor or 16. Random source ports would increasing the numerical attack space "from 2^{32} to 2^{48}", hence increase the difficulty of an attack. Unfortunately, even if random source ports are supported by implementations, routers or gateway devices that perform network address translation (NAT), often rewrite source ports for tracking NAT session. When some NAT devices modify source ports without random source port selection algorithms, it will increase the risk of successful attacks.

The following section will evaluate port randomization of A+P NAT with or without our proposed mechanism and its impact on the DNS cache poisoning attacks.

B. DNS Cache Poisoning Attacks

DNS servers usually store results in a cache to speed further lookups for efficiency, which makes DNS server vulnerable to DNS cache poisoning attacks. DNS cache poisoning is a maliciously created that provides data to a caching name server that did not originate from authoritative DNS sources. Once a DNS server has received such non-authentic data and caches it, it is considered poisoned. The answers from a poisoned DNS server cannot be trusted. The clients may be redirected to malicious web sites that will try to steal clients' identity or infect clients' computers with malware.

DNS requests contain a 16-bit transaction IDs, used to identify the response associated with a given request. Unless the attacker can successfully predict the value of the transaction IDs and return a reply first, the server won't accept the attacker's response as valid. Even if it may be possible to guess these transaction ID values in advance, but as long as the server randomizes the source port of the request, the attack may become more difficult, since the fake response must be sent to the exactly same port that the request originated from. The essence of the problem is that DNS resolvers don't always use enough randomness in their transaction IDs and query source ports. Increasing the randomness of transaction IDs and query source ports may increase the difficulty of a successful poisoning attack. United States Computer Emergency Readiness Team (US-CERT)'s Vulnerability Note VU#800113 describes deficiencies in the DNS protocol and implementations that can facilitate cache poisoning attacks. Most implementations do NOT randomise the port number. In most cases, the same port number 53 was always used.

As stated above, some DNS implementations use a number of mechanisms to protect themselves from the attacks. First of all, queuing received request eliminates the possibility of birthday attack. Secondly, sending the request from the random dynamic UDP port and accepting only answers to this source port. In consequence the probability of successful attack is significantly decreased because the attacker has to guess two 16-bits numbers (both UDP port number and transaction ID). Because these numbers are independent the success probability would be in this case $P = \frac{1}{2^N}$, where N is in practice near 32 (IANA has reserved 0 through 1023 for The Well Known Ports, not all 65,536 could be Ephemeral Port). Even if the attacker uses a high speed connection, the probability of success is relatively small, because the attacker is not able to generate about 2^{32} fake answers in time when DNS is waiting for the reply from the authoritative DNS.

Therefore, upgrading DNS server and DNS resolver to implementations of good randomness is essential to defence against DNS Cache Poisoning Attack. However when DNS resolver behinds routers, firewalls, or other gateway devices

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-133-5

that perform network/port address translation (NAT), if the NAT device does not implement random ephemeral port selection algorithms, consequently it will remove source port randomness implemented by DNS server and stub resolvers. Experiments have been conducted to evaluate if A+P NAT bring deficiency of port randomness when a DNS resolver behinds a NAT.

C. *Comparison of DNS Port Randomness in Three NAT Scenarios*

There is a web-based DNS randomness test tool on the Domain Name System Operations Analysis and Research Centre (DNS-OARC) [8] to help user estimate if their name servers are vulnerable to DNS Cache Poisoning Attacks. This estimation was based on the randomness score of source port and Transaction ID Randomness of DNS resolver. We use the djbdns [9] which sends requests form various source ports and only accepts answers sent to source port, and put it behind NAT devices to test if there is potential randomness deficiency that A+P NAT may bring in.

Three scenarios are given for comparison: In scenario A, DNS resolver behinds a NAT implementing "simple port randomization" algorithm recommended in [6]; in scenario B, DNS resolver behinds a NAT implementing "A+P simple port randomization algorithm" due to our optimized A+P port allocation; in scenario C, DNS resolver behinds an A+P NAT implementing "port increment by 1" algorithm. The estimation result shows that our "A+P simple port randomization algorithm" inherits greater randomness than the algorithm1 recommended in [6]. As shown in Figure 6 and Figure 7, both of them are evaluated as GREAT source port randomness. On the contrary, the traditional A+P NAT "Increment by 1 algorithm" removed source port randomness implemented by the tested DNS server, and is evaluated as POOR source port randomness as shown in Figure 8.

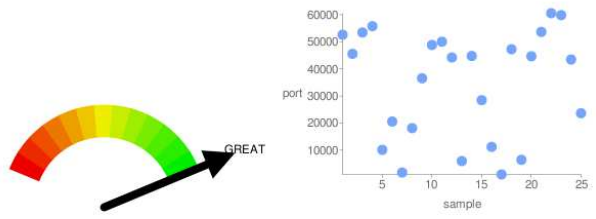

Figure 6. Evaluation DNS randomness in Scenario A

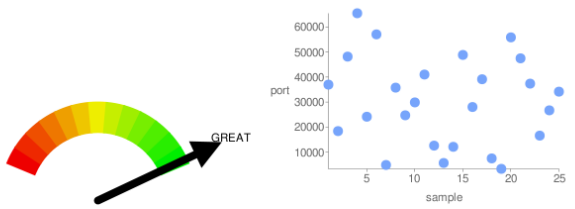

Figure 7. Evaluation DNS randomness in Scenario B

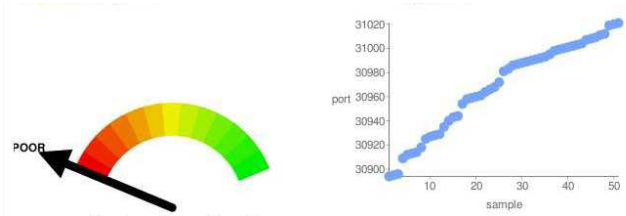

Figure 8. Evaluation DNS randomness in Scenario C

The scatter diagram of 25 sampled ports of scenario A, scenario B and scenario C are shown in Figure 9-11 respectively. The source ports generated by "Simple port randomization NAT" and "A+P simple port randomization NAT" are well distributed, varies from lower bound to upper bound; while "A+P port increment by 1 NAT" generates sequential ports in a very limited range from 30900 to 30925.

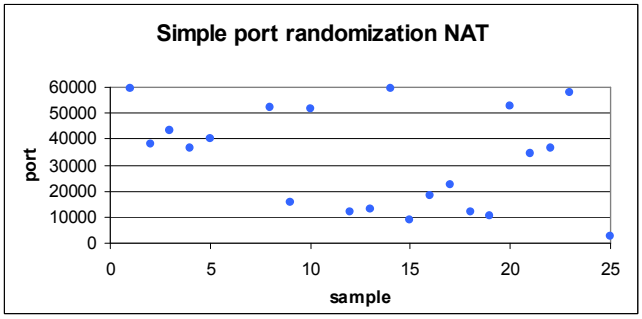

Figure 9. Sampled ports from DNS randomness test in Scenario A

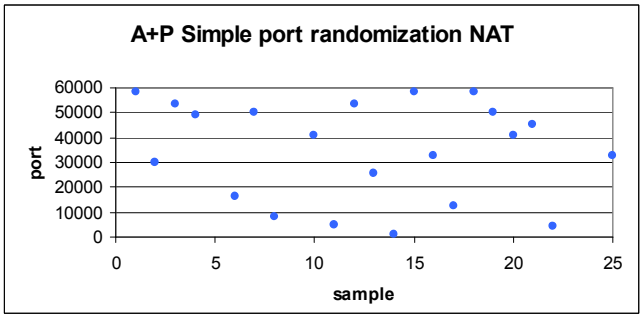

Figure 10. Sampled ports from DNS randomness test in Scenario B

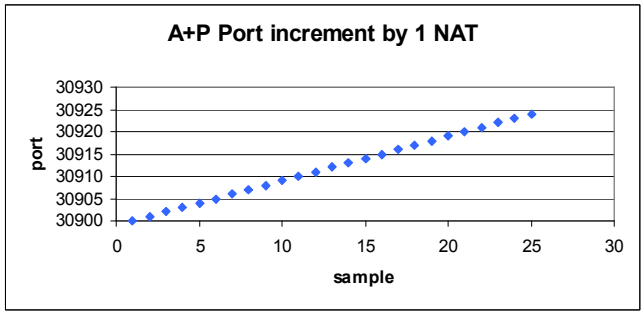

Figure 11. Sampled ports from DNS randomness test in Scenario C

Furthermore, more ports are sampled and Standard Deviation (STDDEV) is used as a measure of randomness as indicated in [8]. The port randomness in scenario A-C are tested. Table II shows the relationship of Port Randomness Score and STDDEV range. In each experiment, the same numbers of ports were sampled for scenario A-C, and the samplings repeat six times. Figure 12-15 shows the STDDEV comparison of the three scenarios when 100, 500, 1000, 5000 ports were sampled respectively, From these four figures, we can see that "Simple port randomization NAT" and "A+P simple port randomization NAT" almost have the same good performance on STEDEV, while "A+P port Increment by 1 NAT" is far underperformance.

TABLE II. PORT RANDOMNESS SCORE

Port Randomness Score	STDDEV Range
GREAT	3980 - 20,000+
GOOD	296 - 3980
POOR	0 - 296

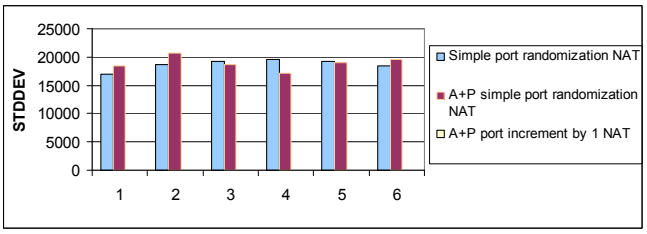

Figure 12. STDDEV Comparison of 100 ports

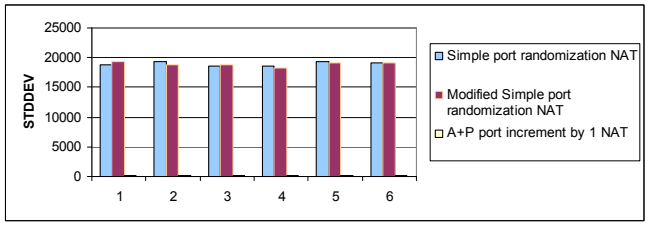

Figure 13. STDDEV Comparison of 500 ports

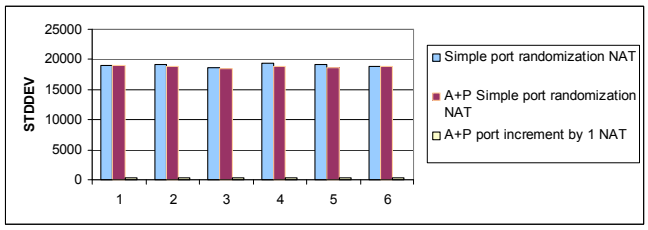

Figure 14. STDDEV Comparison of 1000 ports

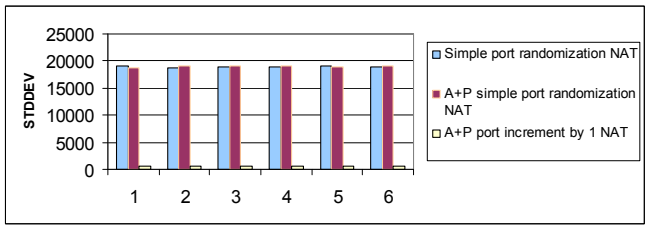

Figure 15. STDDEV Comparison of 2000 ports

V. CONCLUSION

This work introduces an optimized port allocation mechanism for A+P enhancement which has two key features: 1) it provides customer oriented differentiated services by allowing customer negotiates IP-addresses of desired sharing ratios based on their requirement; 2) it supports port randomization. The test result shows that, without our mechanism, "A+P port incremental by 1 NAT" brings randomness deficiency to DNS server and consequently makes DNS server vulnerable to DNS poisoning attacks, while our "A+P simple port randomization NAT" has as great randomness as " simple port randomization NAT " which doesn't bring randomness deficiency to DNS server. Hence this work mitigates A+P's two major defects coming along with allocating ports in cookie-cutter fashion.

REFERENCES

[1] J. Yamaguchi, Ed., "NAT444 addressing models", draft-shirasaki-nat444-isp-shared-addr-03(work in progress), March 8, 2010.

[2] Bush, R., "The A+P Approach to the IPv4 Address Shortage", draft-ymbk-aplusp-04 (work in progress), October 27, 2009.

[3] Durand, A., "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", draft-ietf-softwire-dual-stack-lite-02 (work in progress), March 8, 2010.

[4] Bajko, G. and T. Savolainen, "Port Restricted IP Address Assignment", draft-bajko-v6ops-port-restricted-ipaddr-assign-02 (work in progress), November 2008.

[5] M. Boucadair, Ed, "Port Range Configuration Options for PPP IPCP", draft-boucadair pppext-portrange-option-01(work in progress), July 2009.

[6] M. Larsen, Ed, " Transport Protocol Port Randomization Recommendations", draft-ietf-tsvwg-port-randomization-07 (work in progress), November 30, 2009.

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

[8] Web-based DNS Randomness Test, https://www.dns-oarc.net/oarc/services/dnsentropy, March, 2011

[9] Djbdns, http://cr.yp.to/djbdns.html, March, 2011.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-133-5