

AIN-SLT: Assuring Interoperability between Heterogeneous Networks (IPv4/IPv6) Using Socket-Layer Translator

Ala Hamarsheh

*Department of Electronics and Informatics ETRO Vrije Universiteit Brussel
Faculty of Engineering, Building K, Office No: 4K222
Pleinlaan 2, 1050 Elsene, Belgium
E-mail: ala.hamarsheh@vub.ac.be*

Marnix Goossens

*Department of Electronics and Informatics ETRO Vrije Universiteit Brussel
Faculty of Engineering, Building K, Office No: 4K222
Pleinlaan 2, 1050 Elsene, Belgium
E-mail: marnix.goossens@vub.ac.be*

Rafe Alasem

*Computer Science Department, Imam Muhammad ibn Saud Islamic University
Office No: FR-90, Riyadh, 11432, Saudi Arabia
E-mail: rafe.alasem@gmail.com*

Abstract

This document describes a bi-directional IPv6/IPv4 Socket Layer Translator (SLT) for border gateway routers. The new technique is called AIN-SLT which stands for Assuring Interconnection between Heterogeneous (IPv4/IPv6) Networks Using SLT. This mechanism assures a smooth heterogeneous communication between IPv6 and IPv4 nodes without using protocol translation. It applies IPv6/IPv4 socket Application Programming Interface (API) translation methodology between two heterogeneous networks by terminating IPv6/IPv4 connection points at the application layer. The AIN-SLT mechanism has many advantages over current translation/tunneling approaches, and so it does not require any configuration at end-users' hosts, it has the ability to translate non-NAT friendly traffic (i.e. FTP and SIP traffic), and it achieves a reliable communication by not breaking end-to-end protocol characteristics and security at physical layer (i.e. IPsec).

Keywords: IPv6 Transition, API Socket Layer Translator, IPv6 Performance, Heterogeneous Networks, IPv4/IPv6 Translator, Address Mapping, IPv4-compatible IPv6 Address.

1. Introduction

Until now the Internet is mainly using the old protocol in its communications (i.e. IPv4 [1] protocol). This protocol was a great success in the past 30 years; however, due to limitation that leads to the exhaustion of address space, IPv4 protocol will not be able to address additional Internet connected devices in the next few years. Some standards and techniques have been proposed by Internet

Engineering Task Force (IETF) to save the available IPv4 address space and to alleviate this shortage to some extent (i.e. NAT [2] and CIDR [3]). Unfortunately, all these proposed techniques and mechanisms cannot ultimately resolve exhaustion of IPv4 address space. A new protocol has been developed by IETF to overcome the shortage of IPv4 address space. IPv6 protocol [4] comes with 128-bit address instead of 32-bit in IPv4 address. Unfortunately the two protocols (IPv4 and IPv6) are incompatible with each other. When both users connected to the Internet via different protocols want to communicate without any restrictions, one of transition techniques must be used. Transition from IPv4 protocol to IPv6 protocol will not happen overnight. The coexistence of both IP protocols will remain for a considerable amount of time. IETF has proposed a number of transition mechanisms to ensure independent, smooth, and stepwise transition to IPv6. These mechanisms are characterized into dual stack, tunneling, and translation [5] [6] [7] [8].

Mechanisms BIA [9], DAC [10], and NCTU-SLT [11] are pure host based mechanisms with a purpose to make the desired host capable to run any type of applications (IPv4/IPv6) without bothering the running applications with current host connectivity. The idea behind them is to decouple the application layer from the underlying communication stacks. Table 1 compares these three mechanisms in terms of the host connectivity, running application version, and its ability to translate non-NAT friendly traffic.

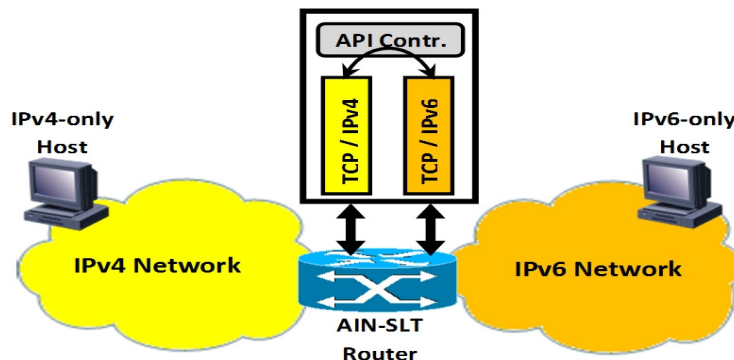
AIN-SLT is router based mechanism which makes a communication between two heterogeneous networks (one of them is working on IPv4-only protocol and other is working on IPv6-only protocol) possible without using protocol translation or tunneling. There will be no need to configure end-user’s host, the Internet Service Provider (ISP) administrator needs to configure border router, moreover deploy new network components. End-users will be able to initiate connections with other hosts connected to different network type. Such network components are: Address Pool IPv4 \diamond IPv6 (AP64) which extends DNS64 [12] functionality or Address Pool IPv6 \diamond IPv4 (AP46) which extends DNS46 [13] functionality (depends on network type – see later) beside alternative DNS server [14]. AP64 is used to synthesize ‘AAAA’ record from ‘A’ record. Likewise, AP46 is used to synthesize ‘A’ record in terms of ‘AAAA’ record.

AIN-SLT extends both DAC and NCTU-SLT functionalities to translate IPv4/IPv6 socket APIs and to translate non-NAT friendly socket APIs as well. Fig. 1 shows AIN-SLT router in which it connects two heterogeneous networks.

Table 1: Compare between BIA, NCTU-SLT, and DAC.

	Dual-Stack Hosts	IPv6-only Hosts	IPv4-only Hosts	IPv6-only Applications	IPv4-only Applications	Translate Non-NAT Friendly Traffic
BIA	✓			✓		
NCTU-SLT	✓			✓		✓
DAC	✓	✓	✓	✓	✓	

Figure 1: AIN-SLT router connecting two heterogeneous networks.



2. Applicability

AIN-SLT could be installed on IPv4/IPv6 dual stack gateway router. One of the most important key aspects of this mechanism is its ability to connect two heterogeneous networks at application layer without breaking end-to-end protocol characteristics and security (i.e. IPsec [15]). Table 2 shows interaction scenarios where it can be installed.

Table 2: All combinations between two heterogeneous networks

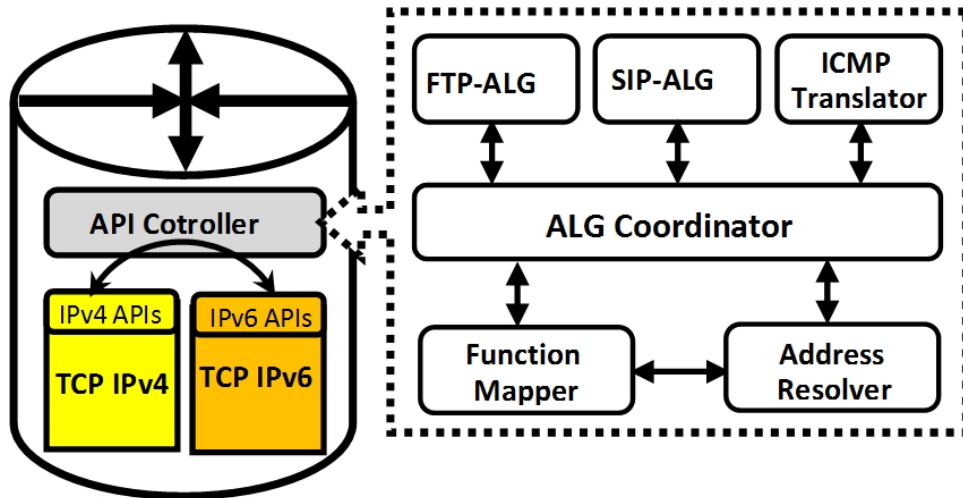
Scenario	Host A	Host B	
1	IPv6	IPv6	Normal
2	IPv6	IPv4	AIN-SLT
3	IPv4	IPv6	AIN-SLT
4	IPv4	IPv4	Normal

AIN-SLT is recommended to be used in the scenarios 2 and 3 only.

3. AIN-SLT Border Router Gateway (ABR) Architecture

API controller layer is one of the most important AIN-SLT components. It consists of the following modules: function mapper, address resolver, ALG coordinator, FTP-ALG, SIP-ALG, ICMP translator. Fig. 2 shows AIN-SLT gateway router architecture.

Figure 2: AIN-SLT gateway router architecture.



3.1. Function Mapper

This module is the same function mapper module used in DAC mechanism. It is responsible for intercepting the received IPvX API functions and translating them into equivalent IPvY API functions and vice-versa. Also it queries address resolver module to retrieve appropriate address type to be used in socket API function translation. The function mapper in this mechanism recognizes addresses generated by either AP64 or AP46 devices. For example, if API controller intercepts an IPv4 API function call, it will translate IPv4 parameters into equivalent IPv6, then query address resolver to retrieve the corresponding IPv6 address that will be used in this new call. Likewise, if it intercepts an IPv6 API function, it will translate the parameters into equivalent IPv4, then query address resolver module to retrieve the corresponding IPv4 address that will be used in this new call. Table 3 contains IPv4/IPv6 parameters translated by function mapper as described in [16].

Table 3: Listing basic IPv4 APIs with new IPv6 APIs

IPv4	IPv6
AF_INET sockaddr_in INADDR_ANY inet_ntoa() inet_addr()	AF_INET6 sockaddr_in6 in6addr_any inet_pton() inet_ntop()

3.2. Address Resolver

It is similar in functionality to the one described in DAC mechanism. Unlike DAC address resolver module, it does not maintain any mapping tables. Instead, it works as switching module between function mapper, AP64, and AP46. It queries AP64 or AP46 for specific type of addresses requested by function mapper module.

3.3. ALG Coordinator

Similar to ALG manager used in NCTU SLT mechanism. The responsibility for this module is to check port numbers in API functions intercepted by function mapper module. Checking port numbers helps in making the decision about the appropriate ALG module to forward the parameters to (messages and port numbers). For example, the port number used in FTP [17] is 21 and in SIP [18] is 5060.

3.4. FTP-ALG

The responsibility of this module is to translate traffic with FTP messages (port numbers and IP addresses) from IPv4 to IPv6 and vice-versa. For more information please look at FTP-ALG module in NCTU SLT mechanism.

3.5. SIP-ALG

An IPv4-only SIP user agent running on IPv4-only host can communicate with IPv6-only SIP user agent by making traffic routed to AIN-SLT router to reach SIP-ALG module; this module has the ability to translate IP address information in SIP session and Session Description Protocol (SDP) [19] header fields. Its functionality is identical to the one described in the NCTU SLT.

3.6. ICMP Translator

This module extends the functionality of SIIT [20] mechanism to translate the ICMP messages values (ICMPv4 [21] and ICMPv6 [22]) from one type to another.

4. Network Specifications

Referring to network type, different configurations required to setup this mechanism at border routers. Referring to interoperation scenario of ISP network, the network administrator should deploy AP64 or AP46 beside alternative DNS server. AP64 uses the functionality of DNS64 and it is used to synthesize 'AAAA' record(s) from 'A' record(s). Whereas, AP46 uses the functionality of DNS46 and it is used to synthesize 'A' record(s) from 'AAAA' record(s). One of these devices should be deployed along with ABR. ABR is responsible for connecting two heterogeneous networks at the application layer. Physically ABR is a router with at least one IPv4 interface and IPv6 interface. Functionally it works like NATs in terms of addresses and port numbers mappings. It can be installed in IPv6-only network to make IPv6 hosts capable to initiate communications with other IPv4-only hosts. Besides, it can be installed in IPv4-only network to make IPv4-only hosts capable to initiate communications with IPv6-only hosts. Finally, ABR is responsible for translating socket API function calls from one IP type to

another and vice-versa. Moreover it translates the IP addresses from IP type to another in order to make them compatible with current socket API function call.

4.1. IPv4-only Network Configurations and Behavior

As mentioned earlier, AIN-SLT mechanism can be configured at the IPv4-only networks to make hosts connected to this network capable to initiate communications with other IPv6-only hosts. To follow that, the network administrator should configure its network with AP46 and ABR devices. AP46 device is responsible for resolving IPv6-only host addresses by synthesizing ‘A’ records from resolved ‘AAAA’ records. Besides, it maintains a mapping table to map IP addresses and port numbers that belong to current active connections. This device is accessed by alternative DNS to resolve addresses for IPv6-only hosts, as well as it is accessed by ABR each time an IPv4 socket API function call has arrived from IPv4 host and destined to IPv6 host or an IPv6 socket API function call has arrived from IPv6 host and destined to IPv4-only host. Table 4 shows the structure of AP46 mapping table. Fig. 3 shows IPv4-only network configured with required AIN-SLT components.

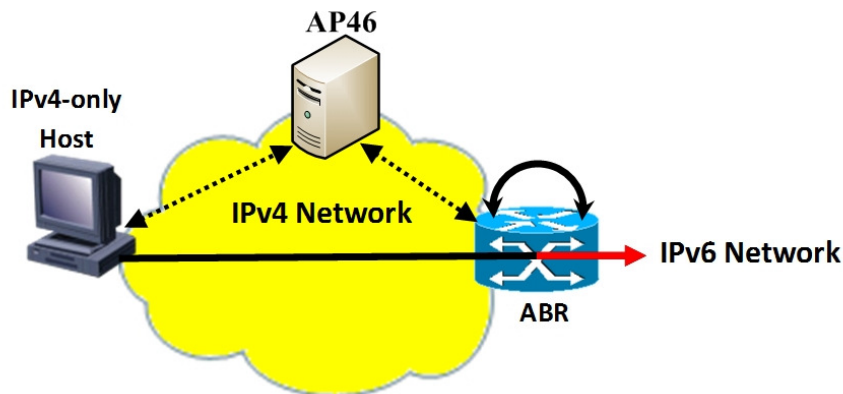
Table 4: Structure of AP46 mapping table

Source IPv4	Dest. IPv6	ABR IPv4	Source Port	Mapped Source Port	Dest. Port
-------------	------------	----------	-------------	--------------------	------------

If IPv4-only host is trying to resolve IPv6-only host address, the AP46 will resolve ‘AAAA’ record(s) instead and then synthesize ‘A’ record(s) from them. AP46 uses unassigned IPv4 address pool [23] to map the IPv6 addresses into corresponding IPv4 addresses. Thereafter, it will map these records in the AP46 mapping table and leaves port number fields empty. Finally, it will create an ‘A’ record using ABR IPv4 address and send created ‘A’ record to IPv4-only host. The IPv4 application will use this ‘A’ record to call IPv4 socket API function. Calling IPv4 socket API function leads to generate an IPv4 packet destined to ABR IPv4 interface. This packet follows IPv4 stack until it becomes an IPv4 socket API function at ABR’s application layer (API Controller). API controller will translate this IPv4 socket API function into corresponding IPv6 socket API function (by translating IPv4/IPv6 function parameters). Finally, it extracts the source and destination port numbers and request AP46 to perform the followings:

- It maps source port number and then updates the related record in the AP46 mapping table by the following values: <source port, mapped source port, destination port>.
- It sends IPv6 address corresponding to source IPv4 address to ABR module.

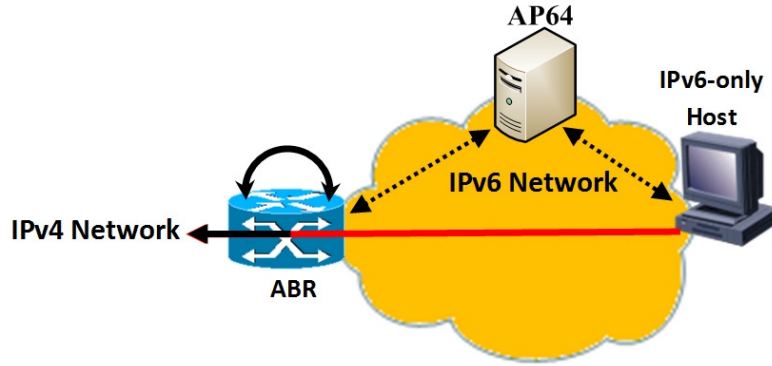
Figure 3: AIN-SLT components at IPv4-only Network



ABR configures its IPv6 interface with IPv4-compatible IPv6 address [24] using AIN-PT Network Specific Prefix (NSP) [25]. ABR will call equivalent IPv6 socket API function using created IPv4-compatible IPv6 address. If a reply is received at ABR side, again it requests AP46 for related

IPv4 address and port numbers. Such information will be used to call the equivalent IPv4 socket API function.

Figure 4: AIN-SLT components at IPv6-only Network



4.2. IPv6-only Network Configurations and Behavior

AIN-SLT mechanism can be configured at IPv6-only networks as well. This leads to make hosts capable to initiate communications with other IPv4-only hosts. To follow that, the network administrator should configure its network with AP64 and ABR devices. AP64 device is responsible for resolving IPv4-only host addresses by synthesizing ‘AAAA’ records from resolved ‘A’ records. Besides, it maintains a mapping table to map IP addresses and port numbers that belongs to current active connections. This device is accessed by alternative DNS to resolve addresses for IPv4-only hosts, and it is accessed by ABR each time an IPv6 socket API function call has arrived from IPv6 host and destined to IPv4 host, or an IPv4 socket API has arrived from IPv4 host and destined to IPv6-only host. Table 5 shows the structure of AP64 mapping table. Fig. 4 shows IPv6-only network configured with required AIN-SLT components.

Table 5: Structure of AP64 mapping table

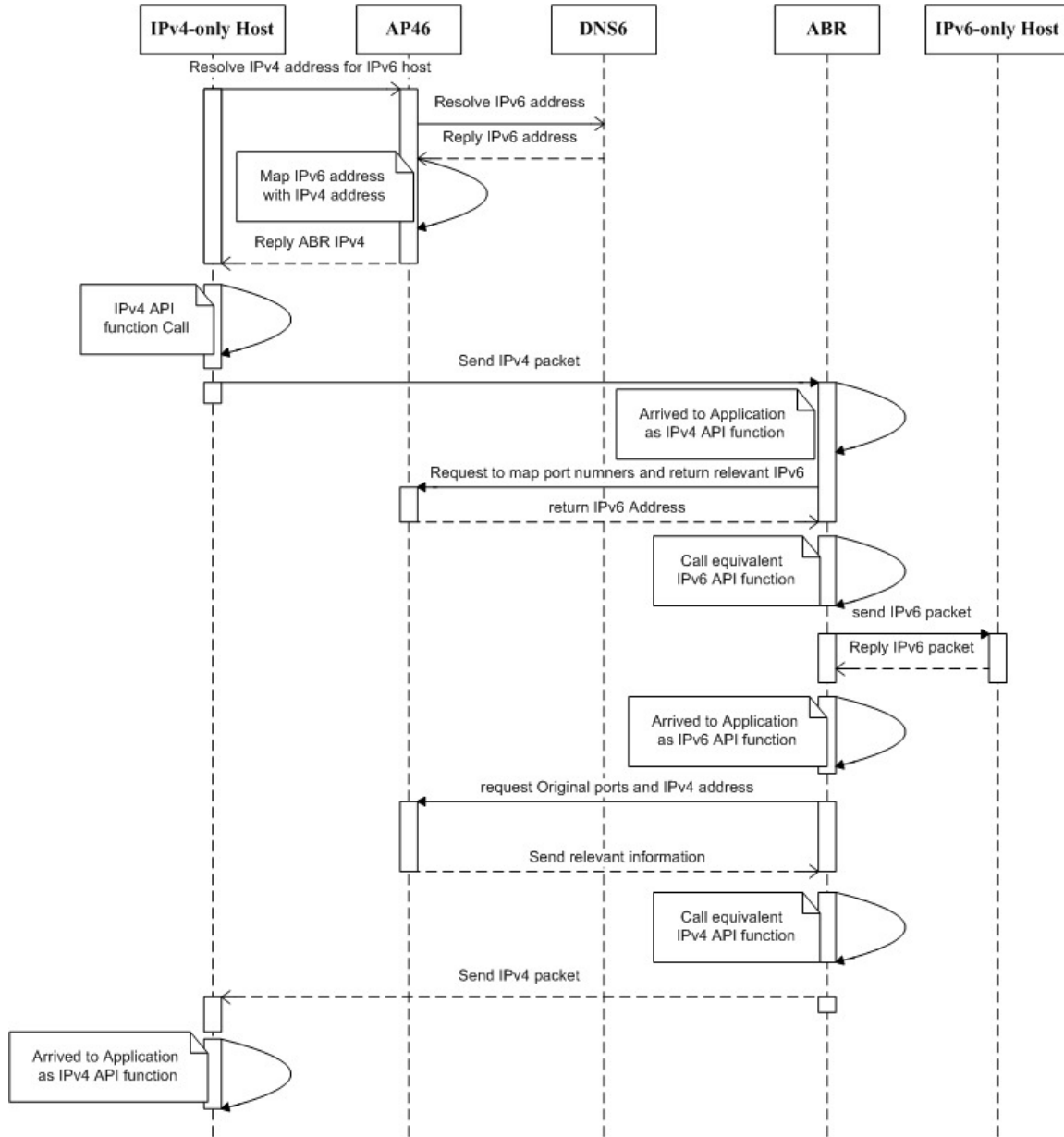
Source IPv6	Dest. IPv4	ABR IPv4	Source Port	Mapped Source Port	Dest. Port
-------------	------------	----------	-------------	--------------------	------------

When IPv6-only host is trying to resolve IPv4-only host address, AP64 will resolve ‘A’ record(s) instead and then synthesize ‘AAAA’ record(s) from them. AP64 is algorithmically configured to synthesize ‘AAAA’ record from the received ‘A’ record. Also there are additional parameters configured in the AP64. One of these parameters is the IPv6 prefix (AIN-PT NSP prefix). AP64 uses this IPv6 prefix and destination IPv4 address to synthesize ‘AAAA’ record(s). Thereafter, it will map these records in the AP64 mapping table and leaves port number fields empty. Finally, it will synthesize an ‘AAAA’ record from ABR IPv4 address and send it to IPv6-only host. The IPv6 application will use this ‘AAAA’ record to call IPv6 socket API function. Calling IPv6 socket API function leads to generate an IPv6 packet destined to ABR IPv6 interface. This packet follows the IPv6 stack until it becomes an IPv6 socket API function at ABR’s application layer (API Controller). API controller will translate this IPv6 socket API function into corresponding IPv4 socket API function (by translating IPv4/IPv6 function parameters). Finally, it extracts the source and destination port numbers and request AP64 to perform the followings:

- It maps source port number and then updates the related record in the AP64 mapping table by the following values <source port, mapped source port, destination port>.
- It sends IPv4 address corresponding to source IPv6 address to ABR module.

Based on this, ABR will call equivalent IPv4 socket API function. If a reply is received at ABR, again it requests AP64 for related IPv6 address and port numbers. Such information will be used to call equivalent IPv6 socket API function. Fig. 5 shows a sequence diagram where an IPv4 host initiates a communication with another IPv6 host.

Figure 5: An IPv4 host initiates a communication with IPv6 host



5. Experimental Results

This section evaluates some of the performance parameters of AIN-SLT mechanism with other protocol translation mechanism i.e. BDMS [26]. The performance evaluations of both AIN-SLT and BDMS were conducted in terms of latency and throughput parameters. As explained in the next two subsections, AIN-SLT shows better performance over BDMS mechanism. All experiments were simulated using OMNET++ [27] simulator. We used TrafGen [28] tool to generate TCP traffic

between client and server. This tool has the ability to define the parameters for certain traffic flows (i.e. packet size, destination, ON and OFF in seconds, etc) between client and server.

5.1. Latency

Latency is the time taken for a packet to be transmitted across a network connection from sender to receiver [29]. In evaluating the performance of the AIN-SLT mechanism, the average transmission latency is measured first. Latency is calculated as appears in the following formulas Eq. (1) and Eq. (2).

$$\text{Latency} = \frac{\sum_{i=1}^{100} \text{Latency}(i)}{100} \tag{1}$$

$$\text{Latency}(i) = \frac{\sum_{j=1}^N Td(j) - Ts(j)}{N} \tag{2}$$

Where,

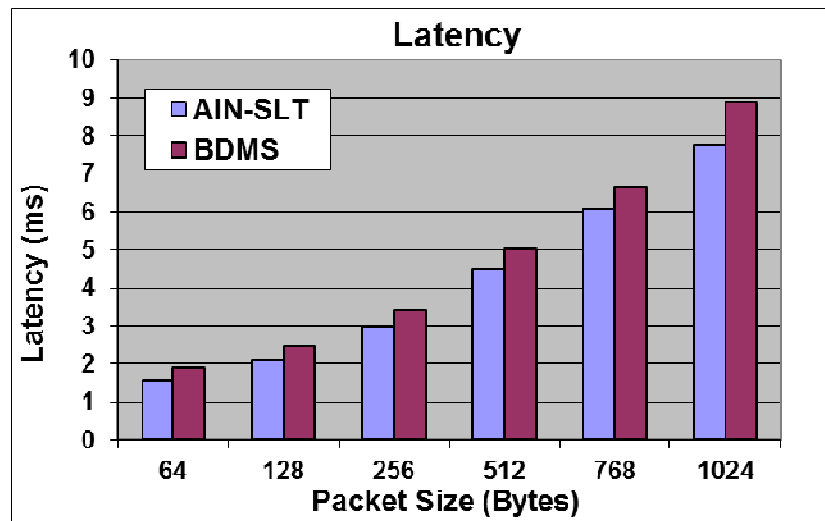
Td: is the time at destination host.

Ts: is the time at the source host.

N: number of received packet at the source host (round trip)

For accuracy, test is repeated 100 times. The reported value of the latency at a specified packet size is the average of the recorded latency values. The overhead occurs due to the time required for translating IP headers and address pool resolutions.

Figure 6: Latency analysis



Latency was measured by sending packets of sizes (64, 128, 256, 512, 768, and 1024 bytes) from IPv4-only client to IPv6-only server. Fig. 6 shows the comparative latency for IPv4 packets transmitted by AIN-SLT and BDMS mechanisms as the size of the packets varied from 64 to 1024. As appears in Fig. 6, AIN-SLT shows better latency due to the minimal processing overhead compared with BDMS mechanism. Overhead varies from 2% to 13% as the packet size is varied. This overhead occurred in AIN-SLT due to AP64 and AP46 requests. BDMS overhead is generated due to DNS64 requests plus CPU processing time in translating IPv4 and IPv6 headers.

5.2. Throughput

Throughput is the maximum data rate that is transmitted over the communication path per unit of time at which none of the frames is dropped by the device [29]. Throughput is calculated from Eq. (3).

$$\text{Throughput} = \frac{\sum_{i=1}^{100} \frac{N_{\text{Rev}} \times P_{\text{size}} \times 8}{(T_{\text{end}} - T_{\text{start}}) \times 1 \times 10^6}}{100} \quad (3)$$

Where,

N_{Rev} : number of packets received at destination point.

P_{size} : packet size in bytes.

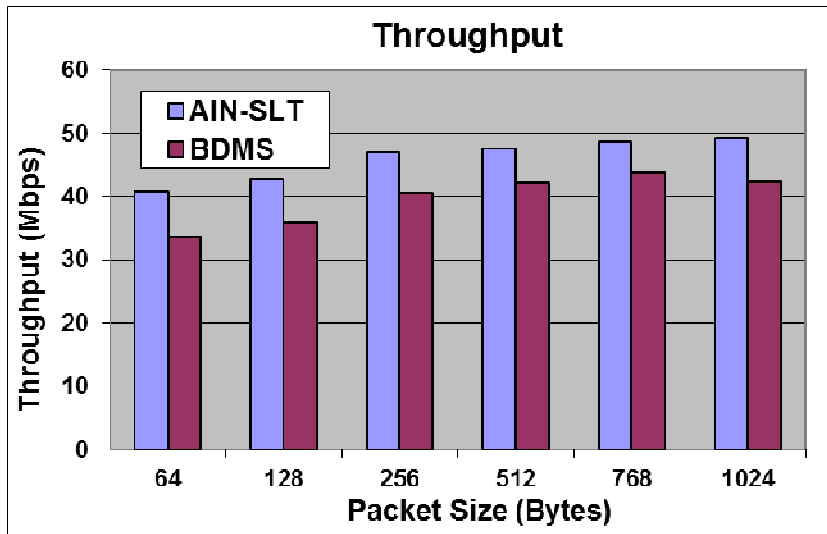
T_{end} : time at destination host (second).

T_{start} : time at source host (second).

For accuracy, test was repeated 100 times. The reported value of the throughput at a specified packet size is the average of the recorded throughput values.

Fig. 7 is the comparative throughput for IPv4 packet transmitted by AIN-SLT and BDMS mechanisms as the size of the packet varied from 64 to 1024 bytes. AIN-SLT shows better throughput over BDMS mechanism. The overhead varies from 2% to 13% as the packet size varied (overhead decreases with increasing the packet size). The overhead occurs due to the time required for translating IP headers and address pool resolutions.

Figure 7: Throughput analysis



6. Conclusion

This paper described a new IPv6 transition tool. It allows IPv4-only hosts to initiate communications with other IPv6-only hosts and vice-versa. Since it terminates the IPv4/IPv6 connection points at application layer, the communication proceeded without breaking end-to-end protocol characteristics and security. In comparison to other protocol translation based mechanisms (i.e. BDMS), AIN-SLT showed better performance in terms of latency and throughput parameters. There are two main limitations of this mechanism, ABR bottleneck and single point of failure.

7. Acknowledgment

This work was funded by European Union, Erasmus Mundus External Cooperation Window (EMECW) programme under project number 141085-EM-1-2008-BE-ERAMUNDUS-ECW-L02, Belgium.

References

- [1] Postel J., 1981, "Internet Protocol", *Internet Engineering Task Force RFC 791*.
- [2] Srisuresh, P., K. Egevang, 2001, "Traditional IP Network Address Translator (Traditional NAT)", *Internet Engineering Task Force RFC 3022*.
- [3] V. Fuller, T. Li, 2006, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", *Internet Engineering Task Force RFC 4632*.
- [4] Deering, S., R., Hinden, 1998, "Internet Protocol, Version 6 (IPv6) Specification", *Internet Engineering Task Force RFC 2460*.
- [5] B. Forouzan, 2003, "TCP/IP Protocol Suite"; *McGraw-Hill*.
- [6] J. F. Kurose, K. W. Ross, 2003, "Computer Networking A Top-Down Approach Featuring the Internet", *Pearson Education, New York*.
- [7] N. Oliver, V. Oliver; 2006, "Computer Networks Principle, Technologies and Protocols for Network Design", *John Wiley and Sons, England*.
- [8] J. Chen, Y. Chang, C. Lin, 2004, "Performance Investigation of IPv4/IPv6 Transition Mechanisms", *Journal of Internet Technology*, Volume 5 No.2, pp. 163-170.
- [9] Lee S., Shin M-K., Kim Y-J., Nordmark E., Durand A., 2002, "Dual stack Hosts Using "Bump-in-the-API" (BIA)", *Internet Engineering Task Force RFC 3338*.
- [10] Ala Hamarsheh, Marnix Goossens, Rafe Alasem, 2011, "Decoupling Application IPv4/IPv6 Operation from the Underlying IPv4/IPv6 Communication (DAC)", Accepted for publication in the *American Journal of Scientific Research*.
- [11] Chen W.E., Su C.Y., Lin Yi-B., 2005, "NCTU SLT: A Socket-layer Translator for IPv4-IPv6 Translation", *IEEE Communications Letter*, 9(10): 865-867.
- [12] Bagnulo M., Sullivan A., Matthews P., Beijnum I., 2010, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", *Internet Engineering Task Force Draft*.
- [13] X. Li, C. Bao, 2010, "DNS46 for the IPv4/IPv6 Stateless Translator", *Internet Engineering Task Force Draft*.
- [14] Internet Architecture Board, 2000, "IAB Technical Comment on the Unique DNS Root", *Internet Engineering Task Force RFC 2626*.
- [15] S. Kent, K. Seo, 2005, "Security Architecture for the Internet Protocol", *Internet Engineering Task Force RFC 4301*.
- [16] Gilligan R., Thomson S., Bound J., McCann J., W. Stevens, 2003, "Basic Socket Interface Extensions for IPv6", *Internet Engineering Task Force RFC 3493*.
- [17] Postel J., Reynolds J., 1985, "File Transfer Protocol (FTP)", *Internet Engineering Task Force RFC 959*.
- [18] Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M., Schooler E., 2002, "SIP: Session Initiation Protocol", *Internet Engineering Task Force RFC 3261*.
- [19] M. Handley, V. Jacobson, C. Perkins, 2006, "SDP: Session Description Protocol", *Internet Engineering Task Force RFC 4566*.
- [20] X. Li, C. Bao, F. Baker, 2010, "IP/ICMP Translation Algorithm", *Internet Engineering Task Force Draft*.
- [21] J. Postel, 1981, "Internet Control Message Protocol", *Internet Engineering Task Force RFC 792*.
- [22] A. Conta, S. Deering, M. Gupta, 2006, "Internet Control Message Protocol (ICMPv6)", *Internet Engineering Task Force RFC 4443*.
- [23] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, 1996, "Address Allocation for Private Internets", *Internet Engineering Task Force RFC 1918*.

- [24] R. Hinden, S. Deering, 2006, "IP Version 6 Addressing Architecture", *Internet Engineering Task Force RFC 4291*.
- [25] Bao C., Huitema C., Bagnulo M., Boucadair M., Li X., 2010, "IPv6 Addressing of IPv4/IPv6 Translators", *Internet Engineering Task Force RFC 6052*.
- [26] Ra'ed AlJa'afreh, John Mellor, Mumtaz Kamala, Basil Kasasbeh, 2008, "Bi-Directional Mapping System as a New IPv4/IPv6 Translation Mechanism", *Tenth International Conference on Computer Modeling and Simulation*.
- [27] "OMNeT++ Network Simulation Framework", <http://www.omnetpp.org>.
- [28] Isabel Dietrich, "OMNET++ Traffic Generator", <http://www7.informatik.uni-erlangen.de/~isabel/omnet/modules/TrafGen/>
- [29] I. Raicu, S. Zeadally, 2003, "Evalating IPv4 to IPv6 transition mechanisms". *IEEE International Conference on telecommunications*.