Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

Caching in ILNP, Identifier Locator Communication Cache (ILCC) Implementation using Three-Way Handshake

Mohsen Kadi[1], Maher Suleiman[2] and Samih Jammoul[3]
[1, 2 3] Faculty of Information Technology,
Higher Institute for Applied Sciences and Technology,
Damascus, Syria,

Abstract – Identifier Locator Network Protocol (ILNP) is one of the prominent solutions to enhance Internet architecture and overrun current challenges. It is a Host-based Identifier/ Locator Split Architecture scheme that operates based on address rewriting. In order to perform address rewriting, each ILNP host uses a network layer logical cache to store state information related to the communicated hosts called Identifier Locator Communication Cache (ILCC). As ILCC plays a fundamental role in every ILNP packet transmission and reception, it must adopt an effective policy to ensure high network stack performance. In this paper, we discuss ILNP caching and its role in performing ILNP operations. As ILNP caching operate at the network layer, we survey the gain from this caching with different transport layer protocols scenarios. After that, we propose a novel approach to implement ILNP caching policy using three-way handshake mechanism between communicated peers. The proposed approach is evaluated after being implemented in Linux OS kernel, and its performance is compared with other implementation. The obtained results show that our policy keeps network stack execution time at average values and enhances host immunity against denial of service attacks that may target network layer sessions.

***Keywords*: Internet, Routing and addressing architecture, Identifier Locator Network Protocol (ILNP), Caching, Identifier Locator Communication Cache (ILCC), and Network layer session**

I. INTRODUCTION

Many research efforts were spent in the last two decades to develop a shared understanding of the challenges facing Internet routing and addressing architecture. As a result of these efforts, currently, there is a common recognition that Internet routing and addressing architecture is facing challenges in scalability, multihoming, and inter-domain traffic engineering [1, 2, 3].

The Internet routing and addressing scaling problems have motivated the research community to seek effective solutions that evolve Internet architecture and keep pace with the current level of growth without requiring restrictive large increases in network operational cost [4]. Many proposals to tackle these concerns are based on Identifier/ Locator Split Architecture (ILSA).

ILSA modifies the current addressing space (IP address space) by splitting it into two separate addressing spaces: 1) Locators (Loc), which is bound to topology semantics and used as a label attached to a network object to provide topological information about it. 2) Identifiers (ID), which is bound to identity semantics and used to differentiate a specific network object within a group of objects. In addition to these address spaces, ILSA contains a mapping system responsible for ID/ Loc mapping and vice versa [5].

ILSA with its three components: Mapping System, Locators address space, and Identifiers address space, represent a candidate solution to overcome most of the current Internet addressing and routing architecture challenges [1, 5].

One of the ILSA schemes is Identifier Locator Network Protocol (ILNP). ILNP is a Host-based ILSA scheme based on architectural concept that substitutes current address space (IP) with two address spaces, namely the locators and identifiers. The architectural concept behind ILNP derives from the GSE/8+8 proposal for IPv6 [6]. Influenced by that work, a set of experimental request for comments that describe ILNP were presented by RJ Atkinson, SN Bhatti, and the University of St Andrews [7-11]. In February 2011, IRTF Routing Research Group (RRG) co-chairs recommended that IETF pursue work on ILNP as a preferred solution for the scalability challenges that face Internet routing and addressing architecture [12]. As the architectural concept of ILNP is independent of the IP version, there are two instantiations of ILNP. The first is ILNPv4, which is based on IPv4 packet format. While the other is ILNPv6, which is based on IPv6 packet format.

ILNP is suggested with the assumption that IP will continue as an addressing space for the Internet rather than bringing a novel clean slate addressing spaces. Hence, ILNPv6 does not introduce any additional namespaces. It rather removes the semantics overloading of IPv6 by separating between the Node Identifier, which is a non-topological name for uniquely identifying a node, and the Locator (Loc), which is topologically bound name for an IP subnetwork. Consequently, the 128-bit IPv6 addresses in the packet headers are split into 64 bits for the Loc and 64 bits for node identifiers (NID). A node may have several NIDs and several Locs. NIDs are based on a modified EUI-64 format with a global or local scope, and Locs have the same syntax and semantic of routing prefix in IPv6, allowing current routers to forward traffic without modification. Based on that, with NID that names a node rather than a specific interface of it, a non-topological address space is available to transport layer protocols to use in their session state. At the network layer, Locators with its topological semantics are used for routing and packet forwarding, but never being used in upper-layer protocols. It is worth noting that in ILNP there is a dynamic binding between NIDs and associated Locators, as well as between {NID, Loc} pairs and interfaces available on the node. The changes brought by ILNP evolve the Internet architecture by assigning a clear and pure semantics for node identifier address space and locator address space, and getting rid of the overloaded semantics of IP address space.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

With this naming evolvement, Internet architecture can provide a pure support for many daily requirements like multihoming and mobility. Not to mention that ILNP, as a host-based ILSA scheme, supports incremental deployments by incremental upgrading of end systems protocols stack, with no need to introduce any changes to network backbone [7, 8]. Moreover, as any ILSA scheme, ILNP needs a mapping system to provide a binding between identifier and locator and vice versa. Current specifications of ILNP adopted the concept of using DNS as a mapping system and dedicated one of the ILNP document set RFC6742 [9] to define new DNS resource records for ILNP.

In order for ILNP to function properly, the implementations must have a logical local cache to store state information related to the communicated hosts. The additional session cache is called an Identifier Locator Communication Cache (ILCC). It is located inside the host's network layer. ILCC maintains information about both the local host and current or recent correspondent hosts. For each ILNP session between two hosts, ILCC records currently valid identifiers and locators with their precedence for the local host, the most recent set of identifiers and locators with their lifetime and validity for the correspondent host, and the local and remote nonce values as stated in RFC6741[8].

With ILCC, ILNP has introduced the concept of network layer session. This network layer session provides the most recent mapping between locators and identifiers of the local and correspondent hosts. Based on that, ILCC plays a crucial role in either the reception or the transmission path of ILNP. In the transmission path, the network layer just receives the local and remote identifiers from the transport layer. So, in order to complete the required information for the outbound packet, ILNP uses state information stored in ILCC to fetch the favorable locators for both local and remote hosts. While in the reception path, when the packet is destined to the local host, the network layer receives both local and remote pairs of locators and identifiers. ILNP uses ILCC to perform a validity check on the received pairs. Based on that check, the network layer may decide to drop the received packet. In the case of passing the previous check, the network layer delivers just the identifiers' information to the upper layer.

In order for ILCC to achieve its objective, it must track the changes of correspondent hosts' locators. This tracking is performed using the Locator Update messages. As the ILNP mobility mechanism is an end-to-end mechanism, the host should inform its corresponding hosts about its locator's changes by sending them a Locator Update message. The Locator Update message is defined in RFC6743 [10] as a new ICMPv6 message to be used by an ILNP host to provide its correspondents with its valid set of locators. It is worth noting that if a host expects incoming connections (in server case), it must update the locator value in DNS in order to enable the correct establishment of new connections.

Since ILCC is an essential part of every packet transmission and reception path in ILNP capable host, its design and performance will affect the whole network stack performance. Therefore, the main objective of this article is to present a comprehensive analysis of ILCC usage with the two transport layer protocols, namely UDP and TCP. Depending on the analysis results, a new approach to implement this caching is introduced. The proposed approach implements a caching policy using three-way handshake mechanism. Finally, the proposed approach is evaluated and compared with other implementation of ILCC.

The rest of the paper is organized as follows: a review of related work is presented in Section 2. Section 3 discusses ILNP caching consideration with UDP and TCP as a transport layer protocols. The proposed caching policy and its implementation using three-way handshake are introduced in Section 4. Section 5 presents our evaluation and results. Finally, Section 6 concludes this work.

II. RELATED WORK

The architectural description and engineering considerations of ILNP are introduced in RFC6740 [7] and RFC6741 [8]. Whereas the work in [13] represented the first study that moved ILNP from the theoretical stage to the implementation stage. This study presents a proof of concept for mobility support in ILNPv6 using an overlay network that depends on UDP/IPv6. The empirical evaluation of the network layer soft handoff shows that the packet loss with soft handoff is minimal and incurs low overhead. Another study [14] developed an analytical cost model to study the performance of multiple host-based and network-based ILSA schemes. Multiple performance metrics were noted in the analytical model, like connection establishment cost, data packet delivery cost, and the impacts of the number of correspondent hosts. The analysis results indicate that a network with a large number of correspondent nodes decreases the performances of both host-based and network-based ILSA schemes dramatically. The first implementation of UDP over ILNPv6 in Linux kernel was introduced in [15]. The performance evaluation of UDP over ILNPv6 in this implementation shows that ILNPv6 handoff performance is better than Mobile IPv6 in terms of throughput, packet loss, and handoff delay. On the other hand, the authors in [16] presented TCP over ILNPv6 implementation in the Linux kernel. The performance evaluation of it shows that ILNPv6 outperforms Mobile IPv6 in terms of flow continuity, packet loss, and handoff delay. Another study that demonstrates seamless communication across different networks was introduced in [17]. The solution presented in this study is based on ILNP, and it can work on existing infrastructure and requires only modifications to the end systems involved in communication. The evaluation results show that ILNP outperformed MIPv6, whereas ILNP had almost zero gratuitous loss during handoff, and very little disordering, as well as very consistent packet transfer and throughput.

While the previous work on ILNP either provides literature for ILNP as RFC6740 [7] and RFC6741 [8], provides an analytical cost model to study the ILNP performance as [14], or demonstrates this protocol support for mobility like [13, 15, 16, 17], the current work scope is performance optimization of ILNP through focusing on one of the most important components that enables ILNP to fulfill its objective, which is ILCC. Also, this work provides a novel ILCC policy that ensures high network stack performance.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

III. ILNP CACHING (ILCC) CONSIDERATIONS

In TCP/IP protocol stack, there are two main protocols that operate at the transport layer. The first one is TCP, which provides connection-oriented, reliable, ordered, and error-checked exchange of flow controlled stream of bytes between processes running on hosts communicating over an IP network. The second protocol is UDP, which provides a simple best-effort datagram exchange service between processes running on hosts communicating via an IP network. In the following, we will analyze the ILNP network layer sessions (or ILCC entries) added value and growth with these protocols.

III.1 ILNP caching with UDP

The host can create a UDP socket using `socket(2)` system call. Although UDP is connectionless, the host can create either a connected or nonconnected UDP socket. For example, any host can create connected UDP sockets by using `connect(2)` system call on that socket. While a nonconnected UDP socket can receive datagrams from any host, the connected UDP socket is used when the process running on the host demands messages exchange with only one peer. Usually, the client-side uses a connected UDP socket, but there are server applications that use connected UDP sockets like Trivial File Transfer Protocol (TFTP).
When the process uses a nonconnected UDP socket, it can receive datagrams from all hosts without any restriction. Moreover, its unrestraint receive will increase when it is named (by `bind(2)` system call) using wildcard that represents all network interface controllers (NIC's) available to the host. In this case, the socket can receive datagrams from all hosts via all available NIC's.
Taking into account the nature of nonconnected and connected UDP socket, we have three different cases for hosts' communication using these sockets:

1- Two nonconnected UDP sockets:

- ILCC entry added value: It is useless to create ILCC entry at the network layer to track locators' changes since each host can receive from any host without restriction. This means, creating such ILCC entry on either or both sides will not provide any added value to the communicating processes. Instead, it will increase the receiving and transmitting path overhead and increase resources consumption at both hosts.
- ILCC entry count: This case will cause uncontrolled growth in ILCC in each host, as each host may communicate with an unlimited number of hosts using its socket.

Consequently, creating ILCC entries for this case must be avoided.

2- Nonconnected with connected UDP socket:

- ILCC entry added value: The movement of the connected UDP socket to a new routing prefix does not affect the nonconnected UDP socket ability to receive packets from it. However, this movement disrupts nonconnected UDP socket ability to transmit packets to the connected UDP socket, as the new location of the later will be unknown to the first. On the other hand, the movement of the nonconnected UDP socket to a new routing prefix results in a connected UDP socket not being able to reach it due to its unknown location. Moreover, this movement leads to a connected UDP socket inability to receive from the nonconnected UDP socket. In order to overcome this problem, the connected UDP socket must have an ILCC entry to track the other side location changes. Furthermore, the nonconnected UDP socket must have a counterpart ILCC entry in order to figure out which correspondents are interested in its location changes to inform them via ICMP Locator Update Message.
Consequently, the ILCC entries at both sides would provide seamless mobility to the communicated hosts via locators' changes tracking, without the need to establish a new socket at either sides.
- ILCC entry count: At the host with connected UDP socket, we have just one ILCC entry related to the socket. On the other side, due to the nature of a nonconnected UDP socket, the number of entries grow linearly with the number of corresponding hosts.

As ILCC entries provide added value especially to the host with connected socket and taking into consideration that just one ILCC entry at the host with a connected socket is not sufficient to ensure this added value, a second ILCC entry at the nonconnected host must be also created if the available resources allow that.

3- Two connected UDP sockets:

- ILCC entry added value: In this case, the two ends of the connection exchange series of packets and each end will send and receive packets from only the other end. By consequence, creating an ILCC entry is mandatory at each host to support mobility, avoid packets drops, and track locations changes.
- ILCC entry count: In this case, we have just one ILCC entry at each end.

Although this case is less common than the others, obviously adding one and only one ILCC entry at each end will be sufficient to result in a great benefit to both ends, with very little overhead. Thus the entries must be established at both hosts.

III.2 ILNP caching with TCP

As TCP is a connection-oriented transport layer protocol, and each end is restricted to communicate with only one end, the ILCC entries are always required for each connection. Moreover, the overhead of creating new ILCC entry for each new connection can be neglected compared with its added value to the communicated hosts.
According to the previous arguments, it is clear that ILCC entries are essential for ILNP to support mobility when communicating ends use connected UDP sockets or TCP sockets. In TCP, whenever the TCP listening socket accepts a new connection, a new ILCC entry must be established. However, we should not omit that ILCC growth caused by TCP sockets is limited to the allowed number of concurrent open connections in the host, and that number differs from host to host based on its resources. While in UDP, when nonconnected socket exchange data with connected sockets, its ILCC entries count increase linearly with the number of corresponding hosts, without restriction.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

As in ILNP, with each packet reception, the host must do a validity check on the received {NID, L} pairs via ILCC lookup, and at each packet transmission, the host must perform an ILCC lookup to pick up the most appropriate locators. Subsequently, the unlimited growth in ILCC combined with the need to consult it in each transmission or reception of packets results in a network stack latency challenge to the hosts. This latency challenge may lead to decrease the benefits of the protocol and thus abandon using it. Moreover, the unlimited ILCC growth makes the host vulnerable to Denial-of-service (DoS) attack.

In summary, ILCC entries are mandatory when TCP is the transport layer protocol, while these entries are not always required in the case of UDP. When these entries are required, the addition to ILCC must be appropriately managed in order to avoid uncontrolled ILCC growth and its resulted implications like undesired latency in reception and transmission, and host vulnerability to DoS attack.

IV. ILNP CACHING POLICY USING THREE-WAY HANDSHAKE

In order to overcome the problem of uncontrolled ILCC growth with its resulting negative impact on hosts, including latency and vulnerability to DoS attack, we introduce three-way handshake as an enabling mechanism of our proposed policy for ILCC entries creation. With this policy, adding a new entry to ILCC is not a pure network layer decision, but rather a decision taken in an integrated manner between the network layer and transport layer. Noting that ILNP engineering consideration RFC6741 [8], which introduces ILCC concept, does not specify a specific policy for ILCC entries creation, we present this policy enabling mechanism with both UDP and TCP transport layer protocols.

IV.1 Three-way handshake with UDP

UDP is a nonconnected transport layer protocol, and within the TCP/IP stack, its datagrams are transported using IP protocol that provides a nonconnected service. On the other hand, ILNP architectural concept introduces the network layer session between the communicated hosts. This session requires communicated hosts to store state information required for ILNP to operate properly. Consequently, when we seek to transport UDP traffic over ILNP, we are trying to add a network layer session concept to a protocol that does not use this concept either at the network layer or transport layer.

In case one of the communicating hosts require ILNP network layer session, we lack the mechanism of informing the other host about this requirement. This case is equivalent to one end use `connect(2)` system call on its socket. The other scenario is when both ends want to inform each other's with their needs for ILNP network layer session, which is equivalent to both ends call `connect(2)` system call on their sockets. Noting that the establishment of this session cannot be forced on the two hosts for reasons discussed earlier in section III, but rather we must equip the hosts with a mechanism that allows them to request session creation.

Through this mechanism, creating a network layer session is controlled by a policy based on the new network layer session added value and the approval of the correspondent. We also took into consideration that many factors may affect correspondent approval, such as the count of network layer sessions and available resources.

Nevertheless, as both UDP & ILNP protocols do not have dedicated control messages for ILCC entries negotiation between communicating hosts, we will use ICMPv6 as a carrier for control messages between hosts. The new ICMPv6 ILCC request message described in this paper enables UDP over ILNP sockets to complete handshake that results in adding ILCC entry at each host. This message represents a control plan that is independent of the data plan in order to maintain the nature of the data exchange provided by the UDP (non-blocking and simple best effort service). Fig. 1 shows this message structure and fields.

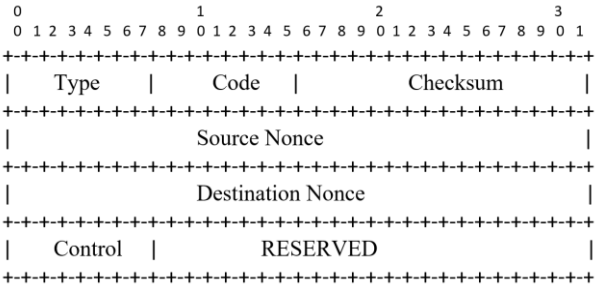

Fig. 1 ICMPv6 ILCC request message

Type: 160, Code: 0, Source Nonce: Sender nonce value. Destination Nonce: Destination nonce value (if it is known). Control: Type of message.

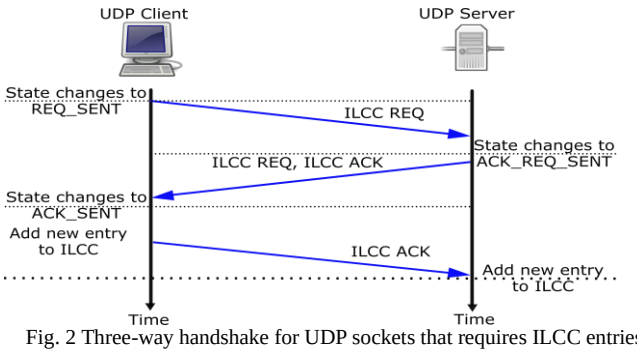

Fig. 2 Three-way handshake for UDP sockets that requires ILCC entries

Based on the control field, we define three types of this message:

1. Request to create ILNP session
2. Request to create ILNP session and acknowledge the received request.
3. Acknowledge the ILNP session creation.

The first and third message types are used by the host that acts as client, and the second message is used by the server host. The message includes the Nonce values of both ends, so each end has all the necessary data to create the new ILCC record. Fig. 2 shows the time sequence of exchanging this message's types. Using this sequence, most of the handshake overload takes place on the host that initiates ILCC entry creation, and usually, this host is the client, whereas the server has to send only one control message.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

IV.2 Three-way handshake with TCP

As TCP is a connection-oriented transport protocol that requires connection establishment before data exchange, there is state information stored at each host's transport layer. Moreover, TCP has a defined control plan for state establishment and maintenance. Based on that, we do not have to adopt a new control plan for creating ILNP's sessions. Instead, TCP three-way handshake is used to achieve this purpose.

V. EVALUATION

The purpose of our evaluation is to demonstrate the necessity of adopting an effective policy for adding entries to ILCC. Also, the adding decision must depend on both the transport layer state and the network layer. The evaluation uses only UDP flows, and in order to perform it, ILNP implementation is required.

To our knowledge, there is only one publicly available prototype implementation of ILNPv6, which specifically is used to demonstrate mobility in ILNP. This implementation is developed by the University of St Andrews and available in [18] for Linux kernel v4.9. The architecture of this prototype, as well as its functionality are described in [19]. Currently, another ILNP implementation in Linux kernel v4.4 is being developed at the Higher Institute for Applied Sciences and Technology (HIAST) as part of our work on ILNP host-based mobility. This implementation adopts our proposed caching policy. In order to perform our evaluation, we compared the HIAST prototype against St Andrews prototype in terms of ILCC growth and network stack latency.

V.1 Experiment Configuration

In our experiment, as ILNP is a host-based ILSA, we used two hosts (attacker and target hosts) on the same LAN. The attacker and the target hosts were each a separate virtual machine. The two hosts were connected using a 1Gbps link. Table 1 shows the specifications of each host.

The target host does not run any UDP service, so each received UDP packet will be dropped. The attacker host uses a simple C program to generate UDP datagrams that destined to the target machine. The attack program uses PF_PACKET socket to generate IPv6 packets with nonce destination option, so the target host handles these packets as ILNPv6 packets. Each generated packet uses a randomly generated source NID, so the target host will consider it as a new packet from a new correspondent host. Using the resource listed in Table 1, the attack program generates about 43 UDP packets with nonce destination option header each second, resulting in a traffic flow of 35Kbps for each running instance of the attacking program.

Each test round lasts for 60 seconds during which we trace the execution time of ip6_input_finish function using trace-cmd command that interacts with the Ftrace tracer built inside the Linux kernel. We choose ip6_input_finish because it is the last IP layer function in the Linux kernel receiving path, and it parses the packet's destination options extension header that includes the nonce option. The final reason for choosing this function is that both HIAST and St Andrews prototypes process nonce option and perform ILCC look up and housekeeping at it.

Table 1: Specification of testbed hardware and software

	Attacker Host	Target Host
CPU	Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz (4 virtual CPUs)	Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz (4 virtual CPUs)
RAM	4 GB DDR4	4 GB DDR4
Operating System	Ubuntu 18.04.1 LTS	Debian GNU/Linux 9
OS Kernel	Linux 4.15.0-45-generic	Either a modified Linux kernel v4.9 or a modified Linux kernel v4.4
Ethernet Adapter	Intel 82545EM Gigabit Ethernet Controller	Intel 82545EM Gigabit Ethernet Controller
Ethernet Driver	e1000 v7.3.21-k8-NAPI	e1000 v7.3.21-k8-NAPI

V.2 Results

In the first test scenario, we ran one attacking process at the attacker host. In this case, ip6_input_finish execution time at St Andrews prototype increased with time progress to reach 424 µs at the end of the test. This undesired increase is a natural result of adding new ILCC entry after each packet reception, despite there is no listening UDP service at the target host. While for the HIAST prototype, the function execution time remained around 34 µs throughout the test period, as shown in Fig. 3 (a). This stability in the function execution time is expected since no entries are added to ILCC due to the absence of any transport layer service that requires this addition.

Fig. 3 (b) shows ip6_input_finish execution time at both St Andrews and HIAST prototypes when the attacker host runs five attacking processes. As St Andrews adds new ILCC entry for each received packet and has no restriction on ILCC growth, the execution time curve increases rapidly. After thirty seconds, the kernel starts to produce scheduling error messages with a wobbly curve. In contrast, the HIAST prototype keeps a steady pace curve throughout the test due to adopting the three-way handshake as a caching mechanism.

In the last test scenario, the attacker host runs ten attacking processes. Fig. 3 (c) shows a faster increase in St Andrews execution time of ip6_input_finish function. After twenty seconds, the target host becomes irresponsive with a considerable amount of kernel scheduling error messages as a result of significant ILCC growth and costly lookups. On the other side, we have almost fixed execution time for ip6_input_finish function.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

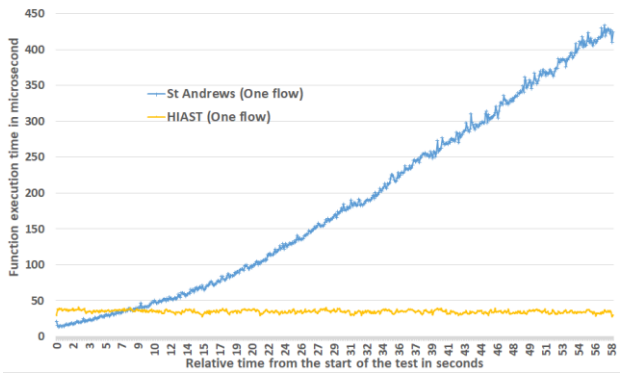

Fig. 3 (a) ip6_input_finish execution time in microseconds during 60 seconds with one attacking process

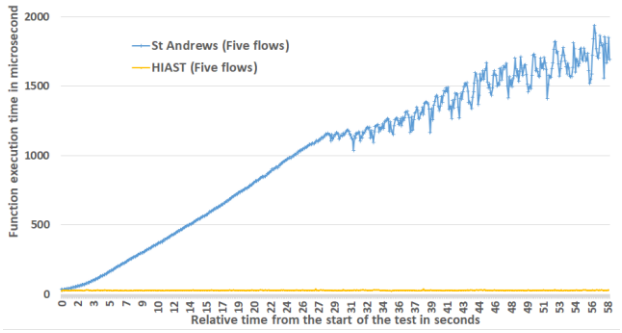

Fig. 3 (b) ip6_input_finish execution time in microseconds during 60 seconds with five attacking processes

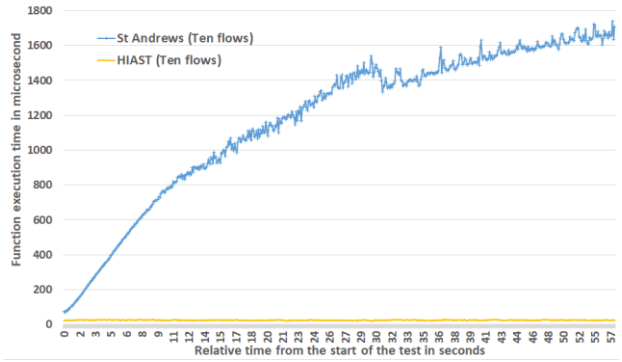

Fig. 3 (c) ip6_input_finish execution time in microseconds during 60 seconds with ten attacking processes

Fig. 4 shows the average of ip6_input_finish execution time of both St Andrews and HIAST prototypes. It can be seen that the average time at St Andrews prototype increases with the increase of the attacking processes. This increase is a consequence of adding new ILCC entries based only on IP layer decision without referring to transport layer requirements. With this uncontrolled ILCC growth, the cost of ILCC lookups is compounded. In our testbed configuration, the target machine became irresponsive with a considerable amount of kernel scheduling error messages when ip6_input_finish execution time reaches 1200 µs. On the other side, at HIAST prototype that adopts three-way handshake mechanism as a caching policy for ILCC, the ip6_input_finish execution time is varying around 28 µs regardless of the number of attacking process.

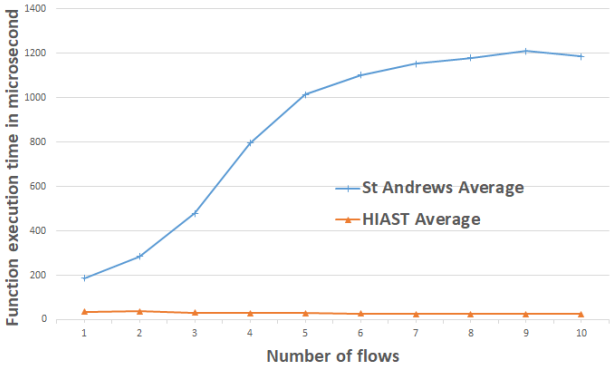

Fig 4 ip6_input_finish average execution time in microseconds for each test scenario

VI. CONCLUSION

ILNP architectural concept substitutes current address space (IP) with two address spaces, namely the locators and identifiers. Using this architectural concept, ILNP removes the semantics overloading of IP. Also, ILNP has introduced the concept of network layer session or ILCC, which provides the most recent mapping between locators and identifiers of the local and correspondent hosts. Consequently, ILCC plays a crucial role in both the receiving and transmitting path of ILNP host. While RFC 6741 [8] describes the content of this cache, it does not adopt an efficient policy for adding entries to this cache. This paper provides an analysis of ILCC entries growth and its side effects in case of TCP and UDP protocols. Based on this analysis, a novel approach to implement ILNP caching policy using three-way handshake mechanism is proposed. By using this policy, adding new entries to ILCC is a decision based on the network and transport layer state. After implementing an ILNPv6 prototype, including our proposed caching policy, an evaluation is performed against St Andrews ILNPv6 prototype in terms of ILCC growth and stack latency. The results show that St Andrews's prototype experience uncontrolled ILCC growth, so the network stack latency increases with the increase of ILCC entries. Also, the host running this prototype is vulnerable to DoS attack due to not adopting an efficient policy for adding entries to ILCC. While HIAST's prototype with its caching policy does not introduce this network stack latency and surmounts DoS vulnerability.

As a future work, we plan to complete the HIAST ILNPv6 prototype and then move the code into Linux kernel mainline. This will allow researchers to conduct further studies to evaluate the performance of this protocol in various scenarios. We will also evaluate the host-based mobility support in our ILNPv6 prototype and compare it with other implementations.

REFERENCES

[1] Meyer, David, Lixia Zhang, and Kevin Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, 2007.
[2] Huston, Geoff. "Analyzing the Internet's BGP routing table." The Internet Protocol Journal 4.1 (2001): 2-15.
[3] Feng, Bohao, et al. "Locator/identifier split networking: A promising future Internet architecture." IEEE Communications Surveys & Tutorials 19.4 (2017): 2927-2948.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

[4] Saleh, Adel AM, and Jane M. Simmons. "Technology and architecture to enable the explosive growth of the internet." IEEE Communications Magazine 49.1 (2011): 126-132.
[5] Ramírez, Wilson, et al. "A survey and taxonomy of ID/Locator Split Architectures." Computer Networks 60 (2014): 13-33.
[6] O'DELL, Mike. "GSE-an alternate addressing architecture for IPv6." draft-ietf-ipngwg-gseaddr-00. txt (1997).
[7] Atkinson, R., S. Bhatti, and U. St Andrews. Identifier-locator network protocol (ILNP) architectural description. RFC 6740, November, 2012.
[8] Atkinson, R., and S. N. Bhatti. "Identifier-locator network protocol (ILNP) engineering considerations." IRTF, RFC 6741 (E) (2012).
[9] Atkinson, R., S. N. Bhatti, and S. Rose. "DNS resource records for the identifier-locator network protocol (ILNP)." IRTF, RFC 6742 (E). 2012.
[10] Atkinson, R., and S. N. Bhatti. ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6). RFC 6743, November, 2012.
[11] Atkinson, R., and S. Bhatti. IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6). RFC 6744, November, 2012.
[12] Li, Tony, and L. Zhang. Recommendation for a routing architecture. RFC 6115, February, 2011.
[13] Phoomikiattisak, Ditchaphong, and Saleem N. Bhatti. "Network layer soft handoff for IP mobility." Proceedings of the 8th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks. 2013.
[14] Abid, Nahla, Philippe Bertin, and Jean-Marie Bonnin. "A comparative cost analysis of identifier/locator split approaches." 2014 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2014.
[15] Phoomikiattisak, Ditchaphong, and Saleem N. Bhatti. "Mobility as a first class function." 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). IEEE, 2015.
[16] Phoomikiattisak, Ditchaphong, and Saleem N. Bhatti. "End-To-End Mobility for the Internet Using ILNP." Wireless Communications and Mobile Computing 2019 (2019).
[17] Yanagida, Ryo, and Saleem N. Bhatti. "Seamless internet connectivity for ubiquitous communication." Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019.
[18] ILNP public release 1. Available at https://ilnp.github.io/ilnp-public-1, Apr. 2020 (Accessed June 20, 2020).
[19] R. Yanagida, and S. N. Bhatti. “End-to-end networking with ILNP in Linux.” 2019 netdev0x13 Technical Conference on Linux Networking, Prague, Czech Republic. Mar 2019.