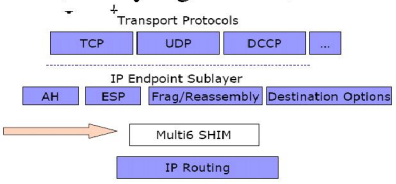
The Implementation of Layer-three Site Multihoming Protocol (L3SHIM)

Kunwoo Park*, Hosik Cho*, Yanghee Choi*, Taekyung Kwon*,
Taewan You†, and Seun

*School of Computer Science and Engineering, Seo
Email: {kwpark, hscho}@mmlab.snu.ac.kr an
†Electronics and Telecommunications Resea
Email: {twyou, syl}@



Abstract— **Layer-three shim (L3SHIM) protocol is designed to support site multi-homing in IPv6 network and now being standardized at shim6 working group in IETF. A host or router can have more than two egress interfaces which are connected to different ISP and configured by distinct IPv6 network prefixes. By using L3SHIM, when an interface is down, another interface can backup for the ongoing connections as it adopts identifier/locator decoupling architecture. Our team is implementing the L3SHIM protocol on Linux based PC using netfilter hooks in the network kernel. We implemented L3SHIM core and REAP component, and verified the feasibility and usefulness of L3SHIM in multi-homed environment by an experiment. We reported the implementation progress to shim6 working group in IETF 67th meeting.**

I. Introduction

In the IP addressing scheme, IP address serves dual role in the Internet: the node identifier and the interface locator. That is, IP address act as both the identifier in transport layer and the locator in network layer. This duality makes hard to support "multihoming" in the Internet since the identifier and locator is tightly coupled. However, decoupling the identifier and locator could be quite useful for some areas like multihoming or mobility as the identifier remains unchanged even if the locator is altered.

To support multihoming and mobility gracefully, several solutions, such as LIN6 [1], HIP [4] and L3SHIM [7], are proposed to make host identification and locating separate from each other. Location Independent Network Architecture (LINA) [2] is proposed to support host mobility. LINA developed LIN6 [1] protocol for host mobility in IPv6 network. LIN6 split up an IP layer into Identification layer and delivery layer. LIN6 use IPv6 address as locator and compose identifier from 64bit LIN id and 64bit prefix of IPv6 address. LIN6 is from a part of WIDE [10] and is already implemented on FreeBSD. HIP [4] provides for limited forms of trust between systems. It enhances mobility, multi-homing and dynamic IP renumbering. It adds identity layer between Transport layer and IP layer. It makes a more secure identifier using public key and hash function. It needs DNS extensions and new agents. Separating the locator and the identifier in HIP, the change of the locator does not affect to the identifier.

Layer-three shim (L3SHIM) protocol [7] is also based on the separation scheme of the locator and identifier. Shim6 working group [5] in the internet engineering task force (IETF) [6] designed the L3SHIM to support the site multi-homing in the IPv6 network and is now standardizing the L3SHIM and peripheral documents. In order to have interoperability with the existing hosts, L3shim use an IPv6 address as an identifier. As the format of addresses used in locator and identifier is same, all identifiers can be locators and vice versa. There are no need to introduce new functionalities in transport layer, so it is easy to deploy in existing network.

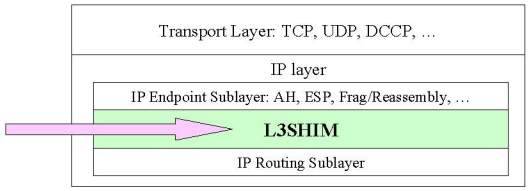


Fig. 1. Where L3SHIM relies

This paper is for the implementation of L3SHIM and its experimental results on IPv6 site multi-homed network environment. The rest of this paper is organized as follows. Section 2 describes the architecture and the operation of L3SHIM protocol. Section 3 contains the L3SHIM implementation environment and design concept. Section 4 shows the experiment setup, scenario and result. Finally, Section 5 concludes this paper.

II. Layer-3 Shim (L3SHIM) Protocol

In this section we introduce layer-3 Shim (L3SHIM) protocol [7]. L3SHIM is a IPv6 based site multihoming protocol proposed by Site Multihoming by IPv6 Intermediation (shim6) working group [5] in Internet Engineering Task Force (IETF) [6]. Unlike other multihoming or mobility protocols, such as LIN6 [1] and HIP [4], L3SHIM does not need any additional infrastructure to run the protocol. Moreover, L3SHIM is interoperable with legacy hosts that does not support the protocol.

A. Architecture

L3SHIM is placed in the IP layer. More specifically, L3SHIM relies between the IP endpoint sublayer and the IP

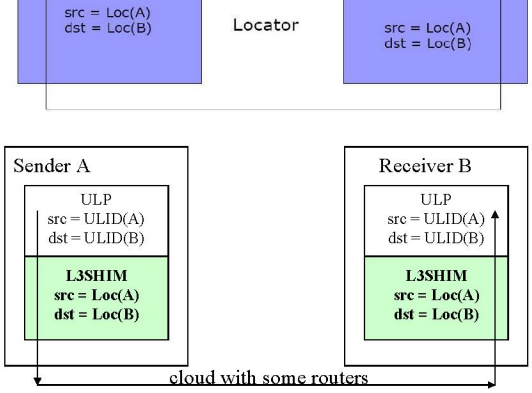


Fig. 2. L3SHIM Operation Feature

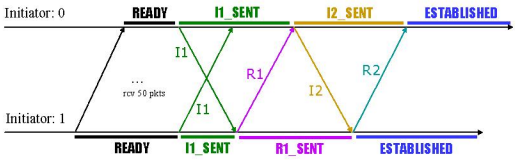


Fig. 3. Locator Exchange Procedure

STATE / EVENT	Operational	Exploring	ExploringOk
Incoming payload packet	STOP Send; START Keepalive	SEND Probe ExploringOk; START Send; GOTO ExploringOk	STOP Send
Outgoing payload packet	START Send; STOP Keepalive	-	START Send
Keepalive timeout	SEND Keepalive	-	-
Send timeout	SEND Probe Exploring; STOP Keepalive; GOTO Exploring	-	SEND Probe Exploring; GOTO Exploring
Reception of the Keepalive message	STOP Send	SEND Probe ExploringOk; START Send; GOTO ExploringOk	STOP Send
Reception of the Probe message State=Exploring	SEND Probe ExploringOk; STOP Keepalive; RESTART Send; GOTO ExploringOk	SEND Probe ExploringOk; START Send; GOTO ExploringOk	SEND Probe ExploringOk; RESTART Send
Reception of the Probe message State=ExploringOk	SEND Probe Operational; RESTART Send; STOP Keepalive	SEND Probe Operational; RESTART Send; GOTO Operational	SEND Probe Operational; RESTART Send; GOTO Operational
Reception of the Probe message State=Operational	STOP Send; START Keepalive	STOP Send; START Keepalive; GOTO Operational	STOP Send; START Keepalive; GOTO Operational

Fig. 4. Behavior of REAP

routing sublayer as shown in figure 1. Locator is a set of information about a host. For example, a host A can connect to a host B by using the locator of the host B. Usually a locator is composed of an IP address and a port number. Upper Layer IDentifier (ULID) is a representative locator of a certain connection. When the current network interface card (NIC) in use goes down, L3SHIM replaces the current locator to the other locator that is valid. However ULID of the connection remains unchanged so that the upper layer, the transport layer, connection is preserved.

B. Operation

When two hosts make a new connection, L3SHIM [7] does nothing. After a host receives 50 packets, it initiates L3SHIM protocol only when the both hosts support L3SHIM protocol. They exchange their current locator sets. If either of one host does not support L3SHIM protocol, nothing happens and the connection is maintained as usually. When the current locator does not work, L3SHIM invokes REAchability Protocol (REAP) [8] to check the state of the connection. Through REAP, L3SHIM detects failure of the connection and explores a new locator pair that is valid. L3SHIM does the locator mapping from the ULID to the current locator of the destination/source field of a packet when the packet is sent, and viceversa when a packet is received. Figure 2 shows the mapping mechanism of L3SHIM.

1) Initial Contact: When two hosts agree to use L3SHIM protocol, they exchange current locator sets. Figure 3 shows the state maintained by the two hosts. When two hosts make a connection, the state of the both hosts is `READY`. After receiving 50 packets, the host sends I1 message to the other host goes into `I1_SENT` state. If the protocol of the connection is TCP-like, each host receives 50 packets nearly at the same time due to the ACK message. To prevent L3SHIM executed twice, if a host that is not an initiator of the connection and the host is in `I1_SENT` state receives I1 message, it ignores the message. The initiator sends R1 message in response to I1 message. Exchange of I1 and R1 message implies that both hosts support L3SHIM. If `I1_SENT` state expires, this means that peer host does not support L3SHIM. Current locator sets of the hosts are piggybacked in I2 and R2 message respectively. Then two hosts go into `ESTABLISHED` state and L3SHIM starts running.

2) Locator and ULID Mapping: Right after L3SHIM is established, shim layer does nothing. We call this "null mapping". However, if REAP is executed and a new locator is assigned to the associated ULID, shim layer starts its job. Whenever a packet comes to the shim layer from upper layer protocol (ULP), shim changes its destination address from the ULID to the locator currently in use. For the receiver's point of view, when a packet arrives at the shim layer, shim layer changes the destination address from the current locator to the ULID that was first used when initial contact was made although ULID does not work at that point. Shim layer maintains connections over site failures by itself without reporting to the ULP. By doing this, L3SHIM supports multihoming.

3) REAP Operation: REAP [8] is roughly divided into two parts; failure detection and alternative address pair exploration. Failure detection is a mechanism to detect a failure when a currently used pair of addresses (or interfaces) between two communication hosts has failed. When a failure occurs, hosts trigger alternative address pair exploration to pick another pair of addresses (or locators). Specifically, when a host decides to explore for an alternative address, it sends a set of Probe messages to the peer until it gets an Probe message from the peer.

L3SHIM is designed whenever there is data traffic in one direction, there is also traffic in the other direction. If there are no traffic from ULP to send to the peer host for certain period, shim layer generates a control message called Keepalive message. This message is sent in order to notify that the path to the host is still available. This way, it is no longer possible to have traffic in only one direction, so whenever there is data traffic going out, but there are no return packets, there must be a failure, so the full exploration mechanism is started. When A decides that it needs to explore for an alternative

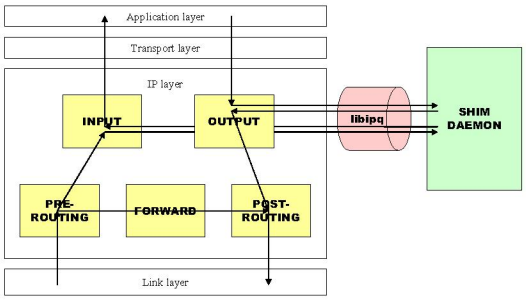


Fig. 5. Implementing L3SHIM

address pair to B, it will initiate a set of Probe messages, in sequence, until it gets an Probe message from B indicating that B has received one of A's messages.

Figure 4 shows the behavior of REAP in the form of a state machine according to each event. [8] defines three states, Operational, Exploring, or ExploringOK, and a node can be in one of them. Operational state is the state that currently there is no problem with the connection. If a node is in the Exploring state, this means that the node has noticed a problem with its locator currently in use. In the ExploringOK state, the node sees traffic from the peer and waiting for the peer to acknowledge the node. Each node maintains two timers, Send timer and Keepalive timer, for Send timeout and Keepalive timeout, respectively. Send timer timeouts when a node sends a traffic but does not receive any traffic from the peer. When a node receives traffic from the peer, the node should send back any traffic (including Keepalive message) before Keepalive timer timeouts. Otherwise, the peer node interprets current connection is lost and proceeds alternative address pair exploration and send Probe message.

III. L3SHIM Implementation

A. Environment and Design

Our target system is the Linux 2.4.x and 2.6.x kernel series. We use netfilter/iptables and ipqueue [9] to capture and mangle packets. Netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack. iptables is a generic table structure for the definition of rulesets. ipqueue delivers packets to the user-space from the kernel-space and viceversa. By invoking two hooks at INPUT hook and OUTPUT hook, we can capture incoming and outgoing packets to the user-space and do the L3SHIM protocol. As figure 5 shows, we implemented SHIM daemon using ipqueue library (libipq).

B. Netfilter

Netfilter [9] is a framework for packet capturing, mangling and filtering implemented in linux kernel. Netfilter has five "hooks" which are well-defined points in a packet's traversal in linux kernel. At each hook, we can capture the packet through netfilter framework and do the job we have registered.

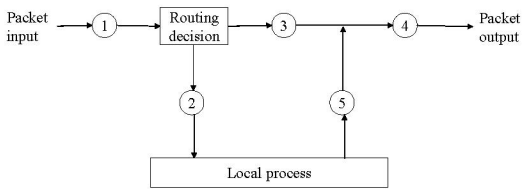


Fig. 6. A Packet Traversing the Netfilter System

For example, we can discard the packet (NF_DROP), allow it to pass (NF_ACCEPT), tell netfilter to forget about the packet (NF_STOLEN), or ask netfilter to queue the packet for userspace (NF_QUEUE). Moreover, by the ip_queue driver, netfilter can send packets to userspace.

Figure 6 shows 5 hooks of netfilter in the network stack.

(1) NF_IP_PRE_ROUTING hook: A packet is passed to the netfilter framework

(2) NF_IP_LOCAL_IN hook: If the packet is destined for the local process, the netfilter framework is called before being passed to the process (if any).

(3) NF_IP_FORWARD hook: If the packet is destined to pass to another interface, the netfilter fame work is called.

(4) NF_IP_POST_ROUTING hook: The packet passes a final netfilter hook before being put on the wire again.

(5) NF_IP_LOCAL_OUT hook: A packet is created locally.

We hook packets at the point of NF_IP_LOCAL_IN(2) and NF_IP_LOCAL_OUT(5) for L3SHIM implementation. Append L3SHIM extension header and change IP addresses via mapping table at NF_IP_LOCAL_IN. At NF_IP_LOCAL_OUT, decode L3SHIM extension header and change IP address to its ULID. Signaling information are carried on L3SHIM extension header.

C. Iptables and ip_queue

Iptables apply rules to the netfilter. There are MATCH, TARGET, and TABLE. MATCH filters packets to mangle. TARGET modifies the packets according to the TABLE. Netfilter hooks all packets from NF_IP_LOCAL_IN and send the packets to the shim6 target. Shim6 target will append shim6 extension header and mangle the source/ destination address referring the MANGLE table.

Libipq is a development library for iptables userspace packet queuing [4]. Netfilter provides a mechanism for passing packets out of the stack for queueing to userspace, then receiving these packets back into the kernel. These packets may also be modified in userspace prior to reinjection back into the kernel.

D. L3SHIM Daemon

In our implementation, L3SHIM is located at userspace. Every packets that comes into the kernel and goes out from the kernel is captured by netfilter and sent to L3SHIM daemon through ip_queue. L3SHIM daemon stores and maintains all the information related to connection, interface, current state, etc. Figure 7 shows the data structure we used. Table1 stores state information about current connections and table2

자료 구조

```
{
ned char local_addr[16];
ned char peer_addr[16];

ned short local_port;
ned short peer_port;
ned long TS;
ned short proto;
ned short CNT;
ned short STATE;
ned short initiator;

igned long TS2;
igned short CNT2;
ned long nonce;
ned short is_valid;

{
ned char ulid[16];
ned char ip[16];
ned short pref;
ned short is_valid;

{
ned char ip[16];
ned short pref;
ned short is_valid;
```

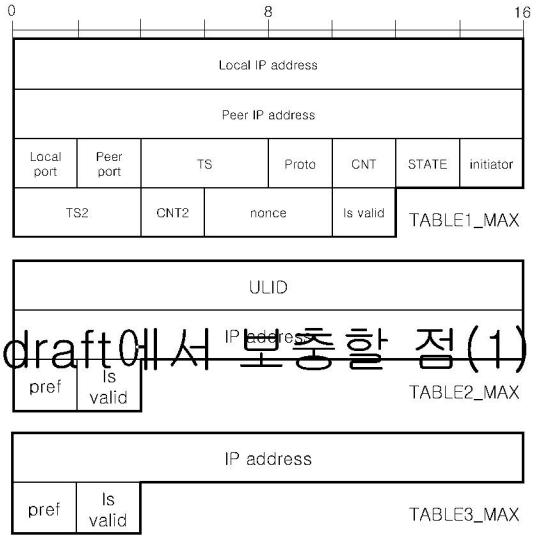


Fig. 7. Data Structure

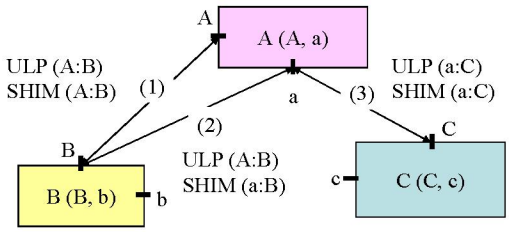


Fig. 8. Experiment Scenario

and table3 maintains local and peer locators respectively. STATE field in table1 is the state of connection including initial states (READY, I1_SENT, I2_SENT, R1_SENT, ESTABLISHED) and REAP states (OPERATIONAL, EXPLORING, EXPLORINGOK). pref field in table2 and table3 is the preference of locators. When the current underlying locator does not work, new trial is made according to the preference of locators. Currently, we haven't implemented security issues about HBA, CGA parameters.

Since we implement L3SHIM in userspace, our code is almost decoupled from linux kernel. Therefore, we can easily adopt the code to other operating systems such as Microsoft Windows. The only thing we have to do is to build a system like netfilter, or find a packet capturing module that delivers packets from kernel space to the userspace. However this kind of implementation has some draw backs. Although userspace coding is easier and faster than kernel coding, it performance is quite low. Which means that every packet has to traverse inside and outside of the kernel and this makes the network performance low. Our implementation team has a plan to put L3SHIM into the kernel space as a future work.

IV. Experiments

L3SHIM was designed for the site multi-homing in IPv6 network. To utilize the full functionalities of L3SHIM protocol, the test-bed needs two or more IPv6 network prefixes from different ISPs. As the connection to IPv6 network is not easy to obtain, we emulated the site multi-homed environment using multiple addresses on an interface. In IPv6, a network interface can have an arbitrary number of addresses. Deleting ULID while in communications will cause the same effect to the path failure on ISP network. And then we look in the operation of L3SHIM and REAP protocols.

Our experiment scenario is described in figure 8. We set three hosts, namely host A, B and C, Let's assume each host's ULID is the upper case of the host name and the other locator is the lower case of the name. At first, host A and B makes a connection with ULID A and ULID B (1). When the interface A goes down, L3SHIM invokes REAP and changes the locator of host A to a. Upper Layer Protocol (ULP) does not see the change and the connection is retained with ULID A and ULID B. If host A connects to host C, A's ULID becomes a. Therefore connection is established between ULID a and ULID C. We used an application called VideoLan [11] for our streaming server and client. We could see that when the interface went down, streaming is stopped and restarted after 10 seconds due to the REAP mechanism.

V. Conclusion and Future Work

In this paper, we introduced L3SHIM protocol and the implementations. To the best of our knowledge, we are the first team to implement L3SHIM protocol. We reported our implementation status in IETF 67th meeting shim working group. We are willing to cooperate with other teams by running each other's L3SHIM protocol together.

Acknowledgment

This work was supported in part by the Brain Korea (BK) 21 project of the Ministry of Education, 2006, Korea.

References

[1] T. Oiwa, M. Kunishi, M. Ishiyama, M. Kuhno, and F. Teraoka, "A Network Mobility Protocol based on LIN6," VTC Fall 2003.
[2] M. Ishiyama, M. Kunishi, K. Uehara, H. Esaki, and F. Teraoka, "LINA: A New Approach to Mobility Support in Wide Area Networks," IEICE, August 2001.
[3] C. Perkins, "IP Mobility Support," IETF RFC 2002, Oct. 1996
[4] R. Moskowitz, "Host Identity Protocol (HIP) Architecture," IETF RFC 4423, May. 2006
[5] http://www.ietf.org/html.charters/shim6-charter.html
[6] http://www.ietf.org/
[7] E. Nordmark, "Level 3 multihoming shim protocol," IETF draft, May. 2006
[8] J. Arkko, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming," IETF draft, June, 2006
[9] http://www.netfilter.org
[10] www.wide.ad.jp, "Widely Integrated Distributed Environment (WIDE)"
[11] www.videolan.org