Testbed for the Comparative Analysis of DS-Lite and Lightweight 4over6 IPv6 Transition Technologies

Omar D'yab, Gábor Lencse
Department of Networked Systems and Services
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics,
Műegyetem rkp. 3., H-1111 Budapest, Hungary.
Email: omardyab@hit.bme.hu, lencse@hit.bme.hu

***Abstract*— This paper aims to build a testbed for two of the most prominent IPv6 transition technologies, namely, Lightweight 4over6 and Dual Stack Lite. Four virtual machines were used to facilitate the implementation of each transition technology. Point to Point tunneling plus stateful NAT64 were used to implement Dual Stack Lite. Snabb was used to implement Lightweight 4over6 testbed. The testbed is built in preparation for the benchmarking measurements to facilitate a comparative analysis of the performance of two tunneling IP transition technologies: Dual Stack Lite as a stateful mechanism, and Lightweight 4over6 as a stateless mechanism.**

***Keywords*—Lightweight 4over6; DS-Lite; AFTR; B4; Performance Analysis.**

I. INTRODUCTION

The depletion of the public IPv4 address pool made IPv4 inadequate, in a consequence of the rapid growth of communication technology and the Internet, and eventually, as predicted, the shortage of the available public IPv4 addresses made the transition to the new version of the Internet Protocol inevitable. IPv6 was developed by the IETF (Internet Engineering Task Force), and its address pool is more than sufficient to cover all current and future needs. IPv6 was designed to be more secure and efficient compared to IPv4, allowing the process of migrating from IPv4 to IPv6 to begin, despite the fact that the two versions of IP are incompatible with each other. As the transition from IPv4 to IPv6 may not happen at the same time all over the word due to several reasons (e.g., there are a billion of devices connected to the Internet, and some of them are even not able to work with IPv6), the two versions of IP must live together for a long time and they must somehow co-operate. There are many IPv6 transition technologies developed by IETF [1], based on different strategies which can be combined. IPv4aaS (IPv4-as-a-Service) is the scenario, when the Internet Service Providers (ISP) use only IPv6 in their access and core network, but they still provide an IPv4 Internet access to their customers. The two main approaches of IPv4aaS are:

1. Double Translation: The IPv4 packets of the subscriber are translated into IPv6 packets by a customer edge device, and they are carried in the access and core network of the ISP as IPv6 packets, finally, they are translated back to IPv4 at the boundary of the ISP network. Such technologies are 464XLAT and MAP-T.
2. Tunneling: The IPv4 packets of the subscriber are encapsulated into IPv6 packets by a customer edge device, and they are carried in the access and core network of the ISP as IPv6 packets, finally, they are de-encapsulated at the boundary of the ISP network. Such technologies are Dual Stack Lite, Lightweight 4over6, and MAP-E.

The use of IPv6 transition mechanisms would greatly depend on many factors: cost, growth, deployment, scalability, infrastructure robustness, and preeminently their performance. This paper focuses on preparing a comparative analysis by conducting experiments proposing a testbed of both DS-Lite and Lightweight 4over6 as stateful and stateless tunneling IPv4aaS technologies, respectively. Thus, we pave the way for measuring their performance, and analyzing their results, which can be extremely beneficial when an Enterprise/ISP determines which IPv6 transition technology to use.

This empirical measurement in [2] conducted a performance study of IPv6 and IPv4 through dual-stack sites from all over the world, using performance metrics: connectivity, throughput, packet loss, hop count, and round-trip time (RTT), considering different regions and times. Compared with IPv6, IPv4 had higher latency and lower throughput with intangible improvements since 2004, IPv6, however, had lower packet loss rate and better connectivity. The average hop count of the IPv6 network is very similar to that of IPv4.

Based on the empirical results in [3], it was found that MAP-E was more feasible compared to other transition technologies. MAP-T and 464XLAT had a better performance in terms of latency as translation-based technology, on the other hand, MAP-E and Ds-Lite had a better performance in terms of throughput as encapsulation-based technology, IPv6NET has shown that it has a high level of repeatability, one flaw in IPv6NET is the lack of control data.

Proving that dual stack is the best technique, achieving better performance in solving the limitations of IPv4, [4] proposed a methodology of four phases: Build & Design network, Statistics, Simulator, and the results of the analysis. The analysis is based on three different scenarios (IPv4, IPv6, and Dual-Stack), which were compared using Riverbed simulator, evaluating five performance metrics*:* Delay, Traffic dropped, Jitter, Packet delay, and CPU Utilization. The results have shown that Dual-stack surpassed IPv4 and gave a better performance.

Our long-term research plan includes the comparative analysis of various IPv6 transition technologies for IPv4aaS. The comparison has various aspects such as performance, security, efficiency of public IPv4 address sharing, etc. The aim of this paper is to be able to set up the two transition technologies as a preparation to move one step further in the direction of performance analysis. As we highlighted above, there is a lot of research in this domain, yet there is a lack of results that we willing to fill. The rest of this paper is organized as follows. In section II, we give a brief introduction to Dual Stack Lite and Lightweight 4over6 transition technologies. In Section III, we illustrate our testbed topology structure for both technologies: Dual Stack Lite and Lightweight 4over6. In Section IV, we describe our future research work plans. Section V concludes and summarizes our paper.

II. Introduction to DS-Lite and Lw4o6

A. Dual Stack Lite

Dual Stack Lite (or DS-Lite) is an RFC 6333 [5] stateful IPv6 tunneling transition mechanism formed on *Network Address Translation (*NAT*), Tunneling,* and *Native IPv6,* allowing clients with IPv6-only Internet access to reach IPv4-only servers. DS-Lite is used to provide IPv4 connectivity to *Customer Premise Equipment* (CPE) using IPv6, a typical scenario would be connecting a client to an IPv4 web server through an IPv6 tunnel that interfaces to the IPv4 network. There are two main components of DS-Lite transition mechanism:

- The Residential Gateway device known as the *Basic Bridging Broadband* element or *B4* in short, is responsible for encapsulating the IPv4 packets of the client into IPv6 and forwarding them to the AFTR.
- *Address Family Transition Router* device known as *AFTR* is a border relay element that de-encapsulates the IPv4 packets from the IPv6 packets and also performs a NAPT (stateful network address/port translation) to provide the IPv4 packets of the client with a public IPv4 source address.

The ISP – on behalf of the client – performs network address translation, all client packets leaving the ISP's AFTR device heading into the IPv4 network will be given a public IPv4 address, hence, extending the life of the IPv4 address pool, in the view of the fact that one public IPv4 address is being shared amongst many CPE devices, allowing those private addresses of RFC 1918 [6] to be used repeatedly, the process of NAT works as a combination of port mapping and addresses, maintaining the inbound (Server to CPE) and outbound (CPE to Server) traffic to be mapped perfectly from source to destination.

B. Lightweight 4over6

Lightweight 4over6 (or lw4o6) is an RFC 7596 [7] compliant stateless IPv6 tunneling transition mechanism, extending the DS-Lite by moving the NAPT from the AFTR to the B4. Lw4o6 inherits some of its functions and components as follows:

- LwB4: *Lightweight Basic Bridging Broadband*, which performs a NAPT, as well as encapsulates the packets of the client into IPv6. LwB4 acts as one of the IPv6 tunnel endpoints.
- LwAFTR: *Address Family Translation Router*, acts as the other endpoint of the tunnel, looks up a static binding table for a match, decapsulates the packets, and routes them to the intended destination relying on A+P (address and port) softwires.

It is noticeable that the lw4o6 moves the functionality of NAPT from the central element, lwAFTR, to the element at the client, lwB4, resulting in the removal of the stateful component from the central element, thus increasing the scalability of the solution. Lw4o6 has a *Binding Table* inside its Softwire as the core of this technology, consisting of IPv4 address, IPv4 port range and IPv6 of B4, per-subscriber basis entry, make it possible to reverse the process.

III. Testbed Design And Implementation

In preparation for the benchmarking measurements, a simple test system was put together for both DS-Lite and lw4o6 using low memory footprint virtual machines for each transition technology. Each virtual machine was created by **`debian-vm`** written by Daniel Bakai [8] using the Debian 8.9 Linux distribution. VMware workstation 16 Player was used to run each testbed.

A. Dual Stack Lite Testbed

1) Topology and System Description

Point-to-Point tunneling was used to implement DS-Lite. In our system test, each virtual machine has 1 CPU core, 128 MB of RAM, and up to 20 GB of a hard disk. Aforesaid the essential components of this technology are *B4* and *AFTR*, where packets are being encapsulated and decapsulated.

The topology was built as illustrated in Fig. 1, separated by three virtual networks: for instance, VMnet10 is an IPv6-only network between B4 (eth2) and AFTR (eth1), on the contrary, both VMnet5 and VMnet7 are IPv4-only networks, VMnet5 on the left side is the network between the Client (eth1) and B4 (eth1), while VMnet7 on the right side is the network between the AFTR (eth2) and the Server (eth1), Table I shows the configuration for each interface and VMware settings for each virtual machine.

TABLE I. DS-LITE LINUX AND VMWARE NETWORK SETTINGS

VM setting	Client	B4	AFTR	Server
eth0 Linux	DHCP	DHCP	DHCP	DHCP
eth1 Linux	10.0.0.5	10.0.0.10	2001:db8:2::2	192.0.2.2
eth2 Linux	N/A	2001:db8:2::1	192.0.2.1	N/A
eth0 VMware	NAT	NAT	NAT	NAT
eth1 VMware	VMnet5	VMnet5	VMnet10	VMnet7
eth2 VMware	N/A	VMnet10	VMnet7	N/A

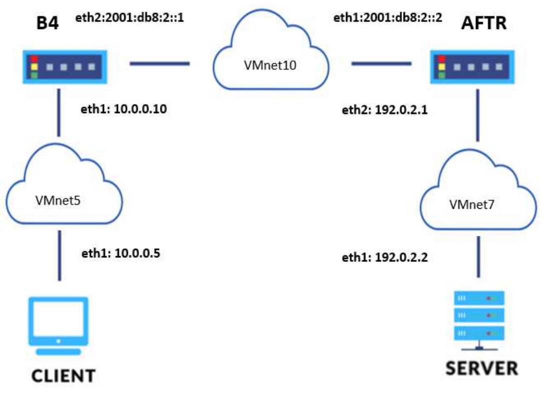

Fig. 1. Dual Stack Lite Testbed Topology

2) B4 Testbed Implementation

The main configuration for encapsulating at the B4, starts by setting up a new IPIP6 virtual interface, the remote address will be the IPv6 address of the AFTR, and the local address is the IPv6 address of the B4 itself, all going via eth2 of the B4, the configuration of B4 to be added as follows:

```
> ip link add name ipip6 type ip6tnl local 2001:db8:0:1::1 remote
2001:db8:0:1::2 mode any dev eth2
> ip link set dev ipip6 up
```

Configuraing a route for encapsualted packets through the tunnel, using kernel IP forwarding to enable forwarding feature for both IPv4 and IPv6, the following script was added:

```
> ip route add 192.0.2.0/24 dev ipip6
> sysctl -w net.ipv4.ip_forward=1
>sysctl -w net.ipv6.conf.all.forwarding=1
```

3) AFTR Testbed Implementation

AFTR, on the other side, has a similar configuration align with the B4 tunnel configuration, as follows:

```
> ip link add name ipip6 type ip6tnl local 2001:db8:2::2 remote
2001:db8:2::1 mode any dev eth1
> ip link set dev ipip6 up
> ip route add 10.0.0.0/24 dev ipip6
> iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
> sysctl -w net.ipv4.ip_forward=1
> sysctl -w net.ipv6.conf.all.forwarding=1
```

B. Lightweight 4over6 Testbed

1) Topology and System Description

Snabb [9] is an open-source implementation that was originally developed to support Deutsche Telekom's Tera Stream network. Snabb is a simple and fast packet networking, written with Lua programing language for high-performance computing by minimizing dependencies including the LuaJIT a just-in-time compiler to deliver multiple applications as a single binary.

Snabb [9] was chosen to implement lw4o6 for the following causes:

1. Its simplicity and ability to support IPv4 and IPv6.
2. It is also practical for high packet rate applications.
3. It can process up to 50 Mbps of a virtual Ethernet card (Virtio-net) network traffic per server.
4. It supports virtualization [10].

In our testbed, each virtual machine has 1 CPU core, 128 MB of RAM, and up to 20 GB of a hard disk, one exception was the lwAFTR memory, which due to loading huge pages was increased to 2 GB. As mentioned, the core components of this technology are: *lwB4* and *lwAFTR*. ***Lw4o6-tun*** is the tunnel where packets are being encapsulated and decapsulated.

The topology was built as illustrated in Fig. 2, separated by three virtual networks: VMnet16 represents the Lw4o6-tun tunnel which is an IPv6-only network that connects lwB4 with lwAFTR, On the contrary, both VMnet15 and VMnet17 are IPv4-only networks, VMnet15 on the left side is the network between the Client (eth1) and B4 (eth1), while VMnet17 on the right side is the network between the AFTR (eth2) and the Server (eth1). Table II shows the configuration for each interface and VMware settings for each virtual machine.

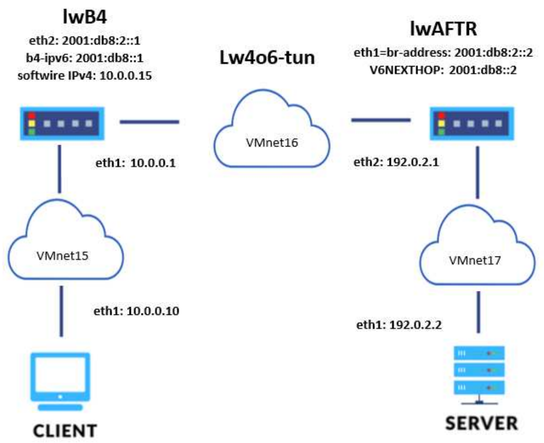

Fig. 2. Lightweight 4over6 Testbed Topology

2) LwB4 Testbed Implementation

The high-level steps of configuring the tunnel, as per our testbed, is initially creating an IP4 in IP6 tunnel with a source

TABLE II. LW4O6 LINUX AND VMWARE NETWORK SETTINGS

VM setting	Client	lwB4	lwAFTR	Server
eth0 Linux	DHCP	DHCP	DHCP	DHCP
eth1 Linux	10.0.0.10	10.0.0.1	br-address: 2001:db8:2::2	192.0.2.2
eth2 Linux	N/A	2001:db8:2::1	192.0.2.1	N/A
internal interface	N/A	b4-ipv6: 2001:db8::1 softwire IPv4: 10.0.0.15	V6NEXTHOP: 2001:db8::2	N/A
eth0 VMware	NAT	NAT	NAT	NAT
eth1 VMware	VMnet15	VMnet15	VMnet16	VMnet17
eth2 VMware	N/A	VMnet16	VMnet17	N/A

address on the B4 of 2001:db8::1/64 and a destination address of 2001:db8:2::2. Once created, we allocate an IPv4 address to the tunnel itself, which is the 10.0.0.15 address, correspondingly, we set up a default route so that all IPv4 traffic is going to be sent via that tunnel, thereupon, we set a source NAT rule, so that all the traffic passes through that tunnel gets Natted to have a source address of the 10.0.0.15; thus 10.0.0.15 address can be added once the tunnel is created. In terms of the size of the supported MTU (Maximum Transmission Unit) on the bridge interface, whether tunneling or translating, each packet gets bigger, for the purpose of our testbed 1400 to 1500 would be sufficient, normally it's 1500. Dealing with IPv6 routing in Linux we need to be very explicit, hence, a metric of 249 was set.

2001:db8::2 is the interface address of lwAFTR referred as ***V6NEXTHOP***, since the address of the tunnel endpoint is a remote route, sending traffic over there, we going to go over the address of lwAFTR on eth2 interface using the lwB4 source address with metric 249 and would be preferred to a default route, variables were used to replace addresses and interfaces prior to the actual script aligned with what we have in the binding table rule as follows:

```
> ip link set eth2 up
> ip -6 addr add 2001:db8:2::1 dev eth2
> ip -6 tunnel add lw4o6-tun remote 2001:db8:2::2 local  2001:db8::1/64
   mode ipip6 encaplimit 4 hoplimit 64 tclass 0x00 flowlabel 0x00000
> ip link set dev lw4o6-tun mtu 1400 up
> ip addr add 10.0.0.15 /32 dev lw4o6-tun nodad
> iptables -A POSTROUTING -t nat -o lw4o6-tun -j SNAT --to-source
   10.0.0.15
> ip route add 192.0.2.0/24  dev lw4o6-tun proto static
> ip -6 route add 2001:db8:2::/64 via 2001:db8::2  dev eth2 src 2001:db8::1
   metric 249
```

3) *LwAFTR Testbed Implementation*

There are two ways of running lwAFTR using Snabb:

1. On-a-stick: lwAFTR <=> IPv4/IPv6, in this mode (enabled by *--on-a-stick interface0*), both IPv4/IPv6 external/internal interfaces are on a single physical interface.
2. Bump-in-the-wire: IPv4 <--> lwAFTR <--> IPv6 in this mode (enabled by *--v4 interface0 --v6 interface1*) both external/internal interfaces are on a dedicated physical interface.

The distinction between on-a-stick and bump-in-the-wire is purely related to the L2/physical topology. In both modes the external/internal interfaces of lwAFTR have distinct MACs and IPs, the only difference if it uses one or two physical Ethernet ports, for our testbed purposes, we have decided to use On-a-stick mode.

The lwAFTR of Snabb requires creating a bridge, and virtual Ethernet interfaces of the bridge, to set it up using Linux kernel. The following script was added to the lwAFTR:

```
> ip link add veth0 type veth peer name veth1
> ip link set veth0 up
> ip link set veth1 up
> ip link set eth1 up
> brctl addbr lwaftr
> brctl addif lwaftr veth0
> brctl addif lwaftr eth1
> ip link set lwaftr up
```

Packet towards the internal/v6 interface will look like this. I.e., destination MAC is the MAC address of the lwAFTR ***internal-interface***, destination IPv6 is the ***br-address*** configured for the softwire, source IPv6 is the ***b4-ipv6*** configured for the softwire. Inside is an IPv4 frame with a source address equal to the IPv4 address configured in the ***softwire***, and the destination address is an IPv4 address on the internet. LwAFTR will remove the IPv6 encapsulation and forward it to the ***external-interface***.

In the other direction (packets arriving at the v4/external-interface of lwAFTR) will look like this: Destination MAC is the MAC address of the ***external-interface***, destination IPv4 address is the ***ipv4 address*** configured in the ***softwire***. Lwaftr will encapsulate in IPv6 as shown before and forward on the internal interface.

LwAFTR transitions from v4-in-v6 to v4 as maintained by the *binding table*, but also vice versa from v4 to v4-in-v6, hence, the core configuration for Snabb is building a lwAFTR bridge which can be configured by constructing its *binding-table* through the *soft-wire configuration*. In the binding table, ***b4-ipv6*** must match lwB4 ***internal-interface***, and ***br-address*** must match lwAFTR ***eth1 address***, which are 2001:db8::1 and 2001:db8:2::2, respectively, as per our testbed. Consequently, the Softwire configuration is to be added by creating a *lwaftr.conf* file as follows:

```
softwire-config {
 instance {
  device "veth1";
  queue {
   id 0;
   external-interface {
    ip : 192.0.2.1;
    mac "02:00:00:00:00:01";
    next-hop {
     ip : 192.0.2.2;
    }
```

B4, eth1
```
13:48:18.286616 IP 10.0.0.5 > 192.0.2.2: ICMP echo request, id 2059, seq 1, length 64
13:48:18.292882 IP 192.0.2.2 > 10.0.0.5: ICMP echo reply, id 2059, seq 1, length 64
```

AFTR, eth1
```
13:48:18.287700 IP6 2001:db8:2::1 > 2001:db8:2::2: IP 10.0.0.5 > 192.0.2.2: ICMP echo request, id 2059, seq 1, length 64
13:48:18.292331 IP6 2001:db8:2::2 > 2001:db8:2::1: IP 192.0.2.2 > 10.0.0.5: ICMP echo reply, id 2059, seq 1, length 64
```

Server, eth1
```
13:48:18.290712 IP 192.0.2.1 > 192.0.2.2: ICMP echo request, id 2059, seq 1, length 64
13:48:18.290767 IP 192.0.2.2 > 192.0.2.1: ICMP echo reply, id 2059, seq 1, length 64
```

Fig. 3. The tcpdump capture of the DS-Lite

LwB4, eth1
```
14:46:48.042862 IP 10.0.0.10 > 192.0.2.2: ICMP echo request, id 3158, seq 56, length 64
14:46:48.045742 IP 192.0.2.2 > 10.0.0.10: ICMP echo reply, id 3158, seq 56, length 64
```

lwAFTR, eth1
```
14:46:48.909121 IP6 2001:db8:2::1 > 2001:db8:2::2: IP 10.0.0.15 > 192.0.2.2: ICMP echo request, id 3158, seq 68, length 64
14:46:48.913058 IP6 2001:db8:2::2 > 2001:db8:2::1: IP 192.0.2.2 > 10.0.0.15: ICMP echo reply, id 3158, seq 68, length 64
```

Server, eth1
```
14:46:48.325435 IP 10.0.0.15 > 192.0.2.2: ICMP echo request, id 3158, seq 187, length 64
14:46:48.323472 IP 192.0.2.2 > 10.0.0.15: ICMP echo reply, id 3158, seq 187, length 64
```

Fig. 4. The tcpdump capture of the lw4o6

```
     }
     internal-interface {
       ip 2001:db8::1;
       mac "02:00:00:00:00:02";
       next-hop {
         ip 2001:db8::2;
       }
     }
   }
 }
 external-interface {
   allow-incoming-icmp true;
   error-rate-limiting {
     packets 600000;
     period 2;
   }
   generate-icmp-errors true;
   mtu 1500;
   reassembly {
     max-fragments-per-packet 40;
     max-packets 20000;
   }
 }
 internal-interface {
   allow-incoming-icmp true;
   error-rate-limiting {
     packets 600000;
     period 2;
   }

   generate-icmp-errors true;
   hairpinning true;
   mtu 1540;
   reassembly {
     max-fragments-per-packet 40;
     max-packets 20000;
   }
 }

 binding-table {
   softwire {
     ipv4 10.0.0.15;
     psid 0;
     b4-ipv6 2001:db8::1;
     br-address 2001:db8:2::2;
     port-set {
       psid-length 0;
     }
    }
   }
  }
```

C. *Ds-Lite and lw4o6 traffic*

Traffic from client to the server has been captured using **tcpdump -nli** command on eth1 of B4, AFTR, and Server of the Ds-Lite, as well as eth1 of lwB4, lwAFTR, and Server of the lw4o6 as shown in Figs. 3 and 4, respectively.

IV. Future Research Work

Our future plan is to move one step further in the direction of a comparative performance analysis of DS-Lite as a stateful and Lightweight 4over6 as a stateless tunneling transition technologies, certainly, using our experience gained with our current testbeds, we started the measurements using a bare-metal system, the plan is to publish the results this year.

V. Conclusion

In this paper we have investigated two of the most prominent tunneling IPv4aaS IPv6 transition technologies: Dual Stack Lite and lightweight 4over6, given a brief introduction, and proposed two testbeds for both technologies which manifest to be suitable for our next research work towards the scalability of the two technologies.

Acknowledgment

The authors would like to thank Snabb community, especially, Ian Farrer and Max Rottenkolber for the help and support they provided in configuring Snabb.

REFERENCES

[1] G. Lencse and Y. Kadobayashi, "Comprehensive survey of IPv6 transition technologies: A subjective classification for security analysis", IEICE Transactions on Communications, vol. E102-B, no.10, pp. 2021-2035. doi: 10.1587/transcom.2018EBR0002.

[2] Li, K.-H.; Wong, K.-Y. "Empirical analysis of IPv4 and IPv6 networks through dual-stack sites". Information, 2021, vol. 12, no. 6, 246. doi: 10.3390/info12060246.

[3] M. Georgescu, H. Hazeyama et al., "Empirical analysis of IPv6 transition technologies using the IPv6 Network Evaluation Testbed", EAI Endorsed Transactions on Industrial Networks and Intelligent Systems vol 2, no 2, (2015), doi :10.4108/inis.2.2.e1.

[4] M. R. A. Ahmed and S. S. A. Shaikhedris, "Network migration and performance analysis of IPv4 and IPv6", 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2021, pp. 1-6. doi: 10.1109/ICCCEEE49695.2021.9429664

[5] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, "Dual-stack lite broadband deployments following IPv4 exhaustion", IETF RFC 6333, 2011, doi:10.17487/RFC6333.

[6] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, 1996, doi: 10.17487/RFC1918.

[7] Y. Cui, Q. Sun, M. Boucadair, T. Tsou, Y. Lee et al., "Lightweight 4over6: An extension to the dual-stack lite architecture", IETF RFC 7596, 2015, doi:10.17487/RFC7596.

[8] D. Bakai, "Debian-VM", [Online]. Available: https://git.sch.bme.hu/bakaid/debian-vm

[9] Snabb, "snabbco/snabb", [Online]. Available: https://github.com/snabbco/snabb.

[10] Snabb Virtualization, "Igalia/snabb", [Online]. Available: https://github.com/Igalia/snabb/tree/lwaftr/src/program/lwaftr/doc.