Design of a Software Tester for Benchmarking Lightweight 4over6 Devices

Ahmed Al-hamadani, Gábor Lencse
Department of Networked Systems and Services
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Budapest, Hungary
Email: alhamadani@hit.bme.hu, lencse@hit.bme.hu

***Abstract*—Several IPv6 transition technologies have been developed to overcome the problem of IPv4 depletion and to accelerate the full deployment of IPv6. The Benchmarking Working Group of IETF has standardized a benchmarking methodology for these technologies in the RFC 8219. One of the most important ones of these technologies is lightweight 4over6 (lw4o6), which is classified as an encapsulation technology in the RFC 8219. This paper aims at designing an RFC 8219 compliant test program for the lw4o6 devices, more specifically, the lightweight Basic Bridging BroadBand (lwB4) and the lightweight Address Family Transition Router (lwAFTR). For this purpose, we overviewed the operational requirements, specified the scope of measurements, and disclosed the design considerations for this tester.**

***Keywords*—benchmarking; IPv6 transition technologies; lightweight 4over6; lwAFTR; lwB4**

I. Introduction

The actual depletion of IPv4 addresses in 2011[1] makes the adoption of IPv6 a necessity more than ever before. However, the ongoing IPv6 transition seems to be a lengthy task due to the numerous challenges it faces. Thus, the coexistence of IPv4 and IPv6 is crucial until, at least, the dominance of the latter one. As a result, many technologies have been developed over the past few years to help in this transition. The IETF's RFC 8219 [2] classifies these transition technologies into four categories, namely, Dual Stack, single translation, double translation, and encapsulation, and defines a comprehensive methodology for their benchmarking.

Dual Stack [3] is a mechanism that includes both IPv4 and IPv6 stacks at the same time in the network nodes, but one of them can be activated for communication at any given time. The benchmarking of Dual Stack interconnecting devices can be sufficiently possible with the existing RFC 2544 [4] and RFC 5180 [5] compliant measurement tools.

The single translation technologies can be benchmarked with the help of the single Device Under Test (DUT) setup of RFC 8219 [2]. Siitperf [6], an RFC 8219 compliant Stateless IP/ICMP Translation (SIIT), also called stateless NAT64, Tester, is an example benchmarking tool that uses this type of setup.

The double translation technologies and encapsulation technologies can be benchmarked in two ways, either using the dual DUT setup or using the single DUT setup of RFC 8219. The first one means that devices that implement the two essential components of the technology, e.g. lwB4 and lwAFTR of lw4o6, are benchmarked together. However, this solution hides the potential asymmetries, when one of the devices becomes a bottleneck. Therefore, RFC 8219 requires that the above-mentioned two devices are also benchmarked one by one according to the single DUT setup.

Several benchmarking tools have been proposed in the literature. Raumer et. al. [7] classified these tools into two different categories, hardware-based and software-based. Hardware-based benchmarking tools are powerful at controlling the transmission rates and can get precise latency measurements, but they are limited to the set of predefined benchmarks, not flexible enough to new situations, and relatively expensive. These are some of the reasons behind not getting widespread utilization. Examples of such devices include those of Keysight [8], Spirent [9], and Xena [10]. The NetFPGA [11], a relatively cheaper and more flexible hardware solution, is an open-source FPGA-based network card that can be deployed in implementing benchmarking measurements [7]. For instance, the traffic generators in [12] and [13] are NetFPGA-based and they can produce precise inter-packet delays and latencies results. In contrast, most of the existing software-based benchmarking tools are originally packet generators that are used in benchmarking tasks. It is correct that they rely on inexpensive commodity hardware and are open-source software, so they can be easily modified, but they are relatively slower and produce less accurate measurements than the hardware-based tools [14]. Additionally, most of them (e.g. Iperf [15] and Harpoon [16]) typically cannot handle high packet rates [7]. Several papers in the literature compare the performance and accuracy among different hardware-based and software-based tools under various conditions and using distinct parameters. Among these papers are [17], [18], and [14]. This paper aims to design a software-based tester that uses Intel's DPDK [19] as it offers high-speed packet processing and typical memory management. Moreover, most of the existing benchmarking tools, hardware-based or software-based, rely on the old RFC 2544 [4] and do not comply with RFC 8219 [2], which is more specific to the IPv6 transition technologies. As far as we know, the only existing RFC8219 compliant testers are dns64perf++ for DNS64 [20] and siitperf for SIIT [6].

One of the most important transition technologies that sit in the encapsulation category is the Lightweight 4over6 (lw4o6) technology [21], which is also considered an IPv4-as-a-Service (IPv4aaS) technology that gives the IPv6-only network operators a practical solution to continue providing customers with IPv4 services [22]. This paper aims at designing the world's first RFC 8219 compliant lw4o6 Tester.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the lw4o6 transition technology. Section 3 presents the basic operational requirements for the Tester based on RFC 8219. Section 4 discloses the most important design considerations. Section 5 gives our plans for implementing the Tester. Section 6 concludes our paper.

II. Lightweight 4over6 (lw4o6) technology

Lw4o6 [21] helps in the incremental deployment of IPv6 by decoupling it in the operator network and makes it possible to share IPv4 addresses by combining two popular technologies: IP in IP, more specifically IPv4 in IPv6, and stateful Network Address and Port Translation (NAPT).

Lw4o6 operates by deploying two different devices: The lightweight Basic Bridging BroadBand (lwB4) and the lightweight Address Family Transition Router (lwAFTR). The lwB4 device can be either a directly connected host device or a Customer Premise Equipment (CPE), which acts as a home gateway for customers and is supplied with a WAN interface provisioned only with IPv6 by the service provider. In addition, the lwB4 device represents one of the IPv6 tunnel endpoints. It encapsulates the customer's IPv4 traffic into the service provider's IPv6 traffic before transmitting it to the lwAFTR device. In contrast, the lwAFTR device represents the other IPv6 tunnel endpoint. When it receives the IPv4 embedded IPv6 traffic, it decapsulates it and then routes it to its intended destination. The reply packets will also traverse these devices, but now the devices execute reverse encapsulation/decapsulation processes. Fig. 1 depicts the architecture of lw4o6 technology.

To manage the traffic activity with the distributed lwB4 devices, the lwAFTR device maintains the so-called softwires (i.e. binding entries of two different IP versions) in a particular address binding table. Each entry in the table is formed on a per-subscriber basis and belongs to a particular lwB4 device. The lwAFTR device uses this entry to perform two tasks: the IPv6 encapsulation of ingress IPv4 packets destined to a customer connected to the related lwB4 device and the validation of egress IPv4-in-IPv6 packets received from the related lwB4 to decapsulate them and then forward the decapsulated IPv4 packets to their intended destinations.

It should also be said that Lw4o6 is actually an improvement of the DS-Lite technology [23]. In contrast to DS-Lite, lw4o6 relocates the stateful NAPT function from the centralized lwAFTR device to the distributed lwB4 devices. This procedure remarkably reduces the overhead of maintaining traffic states from per-flow to per-subscriber and thus logging overhead. It noticeably relieves lwAFTR from being overloaded by translation tasks as it rather has other tasks to accomplish like encapsulation/decapsulation, software maintaining and lookup, and A+P routing.

Finally, the lw4o6 technology also maintains two important mechanisms: provisioning and hair pinning. The first one is used to assign a specific IPv4 public address and a port set for each lwB4 device. This assigned information should also be synchronized with the corresponding information stored in the lwAFTR binding table. The other mechanism (i.e. hair pinning) is used to enable direct communication between two different lwB4 devices that are associated with the same lwAFTR device.

III. Operational requirements and scope decisions

Testing under different operational conditions is important for benchmarking IPv6 transition technologies, as it emulates, to some extent, the condition of a production network environment [2]. In this section, a high-level overview of the requirements of the Tester is given and the considerations behind the scope decision are disclosed.

A. Test and Traffic Setup

As lw4o6 is considered an encapsulation technology, the test setup of this technology may, in general, follow the dual DUT test setup described in section 4.2 of RFC 8219 and shown in Fig. 2. Here, the lwB4 device can act as DUT 1 and the lwAFTR device can act as DUT 2. However, both have asymmetric behavior. Therefore, as RFC 8219 recommends [2], they should be tested separately using the single DUT test setup, which is shown in Fig. 3. In this case, the Tester should have encapsulation/decapsulation capabilities the same as the two DUTs.

```
         Private +-------------+            +----------+
         Private |    lwB4     | IPv4-in-IPv6| Stateless|
+------+  IPv4   |------+------|   tunnel    |  lwAFTR  |    ________
| IPv4 |------->|      |Encap.|------------>|(encap/A+P|   ( IPv4   )
|Device|<-------| NAPT |  /   |<------------|bind. tab +--( Internet )
+------+        |  44  |Decap.|      ^      | routing) |   (________)
                +------+------+      |      +----------+
                              Operator IPv6
                                 network
```

Fig 1. The architecture of lw4o6 technology [8]

```
                 +------------------+
                 |                  |
   +-------------|IPvX  Tester  IPvX|<-------------+
   |             |                  |              |
   |             +------------------+              |
   |                                               |
   |    +----------------+    +----------------+   |
   |    |                |    |                |   |
   +--->|IPvX  DUT1  IPvY|--->|IPvY  DUT2  IPvX|---+
        |                |    |                |
        +----------------+    +----------------+
```

Fig 2. Dual DUT Test Setup [2]

Both test setups (i.e. single DUT and dual DUT) adhere to the following test specifications which comply with RFC 8219:

- Although unidirectional arrows are used, testing with bidirectional traffic is required and testing with unidirectional traffic is optional.
- The two different IP versions are deployed, and they are expressed as IPvX and IPvY, where X=4 and Y=6.
- Ethernet is the media type being relied on even though other media types can also be deployed.
- Frame sizes should be based on the recommendations of RFC 5180 [5]. However, RFC 8219 recommends, besides that, considering Maximum Transmission Unit (MTU) in the context of frame size overhead to avoid frame loss due to MTU mismatch between the virtual encapsulation interfaces and the physical network interface controllers (NICs). Therefore, the larger MTU between them should be set for all interfaces of the DUT and the Tester.
- The selected IPv6 addresses should meet the specifications of Section 5 of RFC 5180 [5], while the selected IPv4 addresses should meet the specifications of Section 12 of RFC 2544 [4].
- UDP is used as the transport layer protocol.
- Tests should also include native IPv6 traffic besides the IPv4 traffic that is encapsulated and different proportions of the two types of traffic must be used.

```
           +--------------------+
           |                    |
+----------|IPvX   Tester   IPvY|<---------+
|          |                    |          |
|          +--------------------+          |
|                                          |
|          +--------------------+          |
|          |                    |          |
+--------->|IPvX     DUT    IPvY|----------+
           |                    |
           +--------------------+
```

Fig. 3 Single DUT Test Setup [2]

To make things more organized, we decided to perform three different types of testing.

1) lwB4Testing: this test follows the single DUT test setup, in which the lwB4 device acts as the DUT. Both the Tester and the DUT should have two interfaces configured as IPv4 and IPv6 respectively. The Tester should be able to send IPv4 packets from its IPv4 interface. The DUT receives these packets from its IPv4 interface, performs NAPT translation, encapsulates IPv4 packets with an IPv6 header, and then forwards the encapsulated packets to the Tester from its IPv6 interface. When the Tester receives these packets from its IPv6 interface, it decapsulates them and gets its original IPv4 traffic. To perform testing in the opposite direction, the Tester should first encapsulate its IPv4 packets with an IPv6 header before sending them from its IPv6 interface. When the DUT receives these packets from its IPv6 interface, it decapsulates them and gets the IPv4 traffic, performs NAPT translation based on the information available in its local NAPT table, and then forwards the IPv4 packets to the Tester from its IPv4 interface. The Tester, in turn, receives these packets from its IPv4 interface.

2) lwAFTR Testing: this test also follows the single DUT test setup, in which the lwAFTR device acts as the DUT. Likewise, both the Tester and the DUT should have two interfaces configured as IPv6 and IPv4 respectively. The Tester should be able to send IPv4 packets encapsulated with an IPv6 header from its IPv6 interface. When the DUT receives these packets from its IPv6 interface, it decapsulates them and verifies their information with that of its binding table. If the verification is successful (i.e. matching is found), the DUT forwards the verified packets from its IPv4 interface. The Tester, in turn, receives the IPv4 packets from its IPv4interface.To perform testing in the opposite direction, the Tester should send IPv4 packets from its IPv4 interface. When the DUT receives these packets from its IPv4 interface, it uses their destination address and port for lookup in its binding table. If a match is found, it encapsulates them with the appropriate IPv6 header and then forwards them from its IPv6 interface. The Tester should be able to decapsulate the IPv6 packets and get the original IPv4 traffic upon receiving the IPv6 packets from its IPv6 interface.

3) Overall Testing: this test follows the dual DUT test setup, in which the lwB4 acts as the DUT 1 and the lwAFTR acts as the DUT 2. This test should be done after performing the two beforementioned tests. This test aims to wrap up everything and check the overall performance of the lw4o6 technology. Here, the Tester should have two interfaces configured as IPv4, while each one of the DUTs should have two interfaces configured as IPv4 and IPv6 respectively. The Tester should be able to send native IPv4 packets from one interface and receive also native IPv4 packets from the other interface. In contrast, each one of the DUTs should receive either IPv4 packets from the Tester or IPv6 packets from the other DUT and send either IPv4 packets to the Tester or IPv6 packets to the other DUT and all these activities should, of course, be done after performing their relative tasks like encapsulation/decapsulation.

B. Scope of Measurements

RFC 8219 recommends applying various types of measurement tests. However, in practice, some of them are implemented by some RFC 2544 testers, and others are omitted or rarely used such as back-to-back frames, system recovery, and reset. The first two require the Tester to have the ability to send at the maximum frame rate of the media, which is practically not possible and is not necessarily met by different devices the user would like to use to run the Tester. The third one would need the ability to cause or sense a DUT reset, which would also require supplementary hardware. So, we intend to support those measurement tests that we see are important. The selected types of measurement need different Tester requirements. In this section, we give an overview of the measurement procedures and their requirements.

1) Throughput: This is a crucial measurement as it is important for users and for supporting other measurement procedures. RFC 8219 depends on the RFC 2544's definition of throughput as it is "the fastest rate at which the count of test frames transmitted by the DUT is equal to the number of test frames sent to it by the test equipment" [4], that is, no frame loss occurs. This means that the Tester must have the ability to transmit frames at any constant rate for any given period and count the transmitted and received packets in that period. We can take advantage of the binary search to find this fastest rate and thus properly apply the measurement. RFC 8219 has defined standard frame sizes for performing the throughput test.

2) Latency: This is also an important measurement that practically depends on throughput. To calculate latency, a stream of frames at a particular frame size and the calculated throughput rate should be sent via the targeted DUT. The duration of the stream should be at least 120 seconds and some of the sent frames should be tagged. For the test to be successful, at least 500 tagged frames should be identified after 60 seconds from the start of sending the stream. For each tagged frame, two timestamps must be recorded. The first one is, directly, at the time of fully transmitting the frame, whereas the second one is, directly, at the time at which the frame is received. The latency represents the difference between the values of the two timestamps. Finally, two quantities should be considered after calculating the latency of at least 500 tagged frames, the Typical Latency (TL) and the Worst-Case Latency (WCL). TL represents the median of all these latencies, while WCL represents their 99.9th percentile. The test must be repeated at least 20 times and it then records both the median of all TLs and WCLs.

3) Packet Delay Variation: Two types of tests can be performed for this measurement, Packet Delay Variation (PDV) and Inter Packet Delay Variation (IPDV), and they are considered important, especially for the quality of real-time applications. However, RFC 8219 recommends PDV and keeps IPDV optional for a fine-grained analysis of delay variation. Thus, only PDV will initially be included. Similarly, To calculate PDV, a stream of frames at a particular frame size and the calculated throughput rate should be sent via the targeted DUT. Here, the duration of the stream should rather be at least 60 seconds and the value of the one-way delay of *all* frames should be measured. Then, the PDV represents the difference between the 99.9th percentile value and the minimum value in the stream. Finally, the test must be executed at least 20 times and the final reported value will be the median of all 20 calculated PDVs.

4) Frame Loss Rate: This measurement test is similar to the throughput measurement test. Here, the test is also run by sending a specific number of frames at a specific rate via the targeted DUT and then counting the number of received frames at the Tester. The Frame Loss Rate (FLR) is then calculated as in (1):

$$\text{FLR} = ((\textit{sent} - \textit{received}) / \textit{sent}) * 100\%. \qquad (1)$$

What differs from the throughput test is that, here, this test is performed at various frame rates starting from the maximum frame rate for the media and decreased by some percentage values (typically 10%) at each new trial until there are two consecutive trials in which no frames are lost.

IV. DESIGN CONSIDERATIONS

These are some essential design considerations that should be taken into account when implementing the lw4o6 tester.

C. Integration or Separation

It may be a desirable solution to build a fully integrated tester that automatically performs all measurement tests, and this can act as a commodity tester for routine tests. However, this paper proposes a lw4o6 tester that is designed mainly for research purposes. Thus, the goal is to design a flexible tool that runs high-performance programs for the elementary functions and uses input parameters instead of built-in constants even though the RFC 8219 allows using constants like 60 seconds duration or 500 timestamps, and so on. Additionally, it is intended to use modifiable bash scripts to run these programs. All these solutions can help the user to access all intermediate results and experiment easily by executing only certain sub-tasks when needed.

D. Software Architecture and Hardware Requirements

RFC 8219 requires, in general, using bidirectional traffic. To build a clear program structure with high enough performance, a thread pair (i.e. a thread for sending and another thread for receiving) should be used for the forward direction and another thread pair for the reverse direction. If we said that each thread will be executed by one CPU core, then we need four CPU cores for the execution plus an additional CPU core for the main program. It may also be noted that either of the two directions may be primarily inactive. The Tester and the two DUTs need two NICs each for testing purposes and a third one for network communication unless the user prefers to work locally on the console.

E. Input and Output

The decision of separation described earlier offers the possibility for the shell scripts to execute the programs multiple times with two potential forms of parameters, static and dynamic. Those parameters that will not change during execution such as IP addresses, MAC addresses, etc., their values can be statically supplied in a configuration file, while those parameters that change from time to time such as frame size, frame rate, etc., their values can be easily provided as command-line arguments.

The results that are to be used by the shell scripts in making some decisions (e.g. number of frames sent, number of frames received, etc.) are to be printed on the standard output using a clear format to be easily extracted for further processing. In contrast, those results which are big or no longer been needed by the shell script should be written into a separate result file.

V. FUTURE WORK

The most important step to be taken next is to implement the lw4o6 tester. We plan to write it in C++ and follow similar implementation steps as that of siitperf [6]. It is always preferable to use high-performance solutions like Intel's DPDK framework[19] during implementation as it offers fast-packet processing and typical memory, queue, and buffer management. What follows is the validation of the tester through comprehensive benchmarking measurements. Once successfully developed, the lw4o6 tester is planned to be distributed as free software under GPL license for the benefit of the research, benchmarking, and networking communities. Additionally, the developed lw4o6 tester could also be used as a basis for the development of testers of other important IPv6 transition technologies, especially those IPv4aaS technologies that have never been benchmarked yet.

VI. CONCLUSIONS

In this paper, all the necessary details about the operation of the lw4o6 transition technology have been described, high-level details of the operational requirements and the scope of measurements (e.g. throughput, latency, packet delay variation, and frame loss rate) of the RFC 8219 compliant lw4o6 test program are given, and the most essential design considerations and decisions for this test program have been disclosed.

REFERENCES

[1] A. Al-Azzawi, "Towards the security analysis of the five most prominent IPv4aaS technologies", Acta Technica Jaurinensis, vol. 13, no. 2, pp. 85-98, 2020, doi:10.14513/actatechjaur.v13.n2.530.

[2] M. Georgescu, L. Pislaru, and G. Lencse, "Benchmarking methodology for IPv6 transition technologies", IETF RFC 8219, 2017, doi:10.17487/RFC8219.

[3] E. Nordmark, and R. Gilligan, "Basic transition mechanisms for IPv6 hosts and routers", IETF RFC 4213, 2005, doi:10.17487/RFC4213.

[4] S. Bradner, and J. McQuaid, "Benchmarking methodology for network interconnect devices", IETF RFC 2544, 1999, doi:10.17487/RFC2544.

[5] C. Popoviciu, A. Hamza, G. V. d. Velde, and D. Dugatkin, "IPv6 benchmarking methodology for network interconnect devices", IETF RFC 5180, 2008, doi:10.17487/RFC5180.

[6] G. Lencse, "Design and implementation of a software tester for benchmarking stateless NAT64 gateways", IEICE Transactions on Communications, vol. E104-B, no. 2, pp. 128-140, 2021, doi:10.1587/transcom.2019EBN0010.

[7] D. Raumer, S. Gallenmüller, F. Wohlfart, P. Emmerich, P. Werneck et al., "Revisiting benchmarking methodology for interconnect devices", in Proc. 2016 Applied Networking Research Workshop, Berlin, Germany, 2016, pp. 55–61, doi:10.1145/2959424.2959430.

[8] KeysightTechnologies, "IxNetwork", https://www.keysight.com/zz/en/products/network-test/protocol-load-test/ixnetwork.html, [cited 23-06-2021].

[9] Spirent, "Spirent Testcenter: RFC2544 benchmarking test package", https://assets.ctfassets.net/wcxs9ap8i19s/5Nlwm12wtCKS83GJnlxUko/f7c406269e71a8f7b7490e7060d66aea/STC_RFC-2544_Benchmarking_Test_Package_datasheet.pdf, [cited 24-06-2021].

[10] XenaNetworks, "Quality of service validation", https://xenanetworks.com/whitepaper/quality-of-service-validation/, [cited 24-06-2021].

[11] NetFPGA. https://netfpga.org/.

[12] G. A. Covington, G. Gibb, J. W. Lockwood, and N. Mckeown, "A Packet generator on the NetFPGA platform", in Proc. 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, CA, USA, 2009, pp. 235-238, doi:10.1109/FCCM.2009.29.

[13] M. Ghobadi, G. Salmon, Y. Ganjali, M. Labrecque, and J. G. Steffan, "Caliper: Precise and responsive traffic generator", in Proc. 2012 IEEE 20th Annual Symposium on High-Performance Interconnects, 2012, pp. 25-32, doi:10.1109/HOTI.2012.16.

[14] P. Emmerich, S. Gallenmüller, G. Antichi, A. W. Moore, and G. Carle, "Mind the gap - A comparison of software packet generators", in Proc. 2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Beijing, China, 2017, pp. 191-203, doi:10.1109/ANCS.2017.32.

[15] Iperf. https://iperf.fr/.

[16] J. Sommers, and P. Barford, "Self-configuring network traffic generation", in Proc. 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, 2004, pp. 68–81, doi:10.1145/1028788.1028798.

[17] M. Paredes-Farrera, M. Fleury, and M. Ghanbari, "Precision and accuracy of network traffic generators for packet-by-packet traffic analysis", in Proc. 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006, (TRIDENTCOM 2006), Barcelona, Spain, 2006, pp. 6-37, doi:10.1109/TRIDNT.2006.1649124.

[18] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?", IEEE Communications Magazine, vol. 48, no. 9, pp. 158-165, 2010, doi:10.1109/MCOM.2010.5560600.

[19] D. Scholz, "A look at Intel's dataplane development kit", in Proc. Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), Munich, 2014, pp. 115-122, doi:10.2313/NET-2014-08-1_15.

[20] G. Lencse, and D. Bakai, "Design and implementation of a test program for benchmarking DNS64 servers", IEICE Transactions on Communications, vol. E100-B, no. 6, pp. 948-954, 2017, doi:10.1587/transcom.2016EBN0007.

[21] 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.

[22] G. Lencse, J. P. Martinez, L. Howard, R. Patterson, and I. Farrer, "Pros and cons of IPv6 transition technologies for IPv4aaS", active Internet Draft, 2021, https://tools.ietf.org/html/draft-ietf-v6ops-transition-comparison-00

[23] 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.