Acta Technica Jaurinensis

Vol. 14, No. 2, p. 186-211, 2021
DOI: 10.14513/actatechjaur.00577

A survey on the performance analysis of IPv6 transition technologies

A. Al-hamadani[1,*], G. Lencse[1]

[1]Department of Networked Systems and Services, Budapest University of Technology and Economics, Magyar tudósok körútja 2, 1117 Budapest, Hungary
***e-mail: aalhamadani@uomosul.edu.iq**

Submitted: 24/11/2020; Accepted: 28/01/2021; Published online: 03/02/2021

Abstract: As the public IPv4 address space has already been depleted, the full deployment of IPv6 became indispensable, especially for service providers, as it offers a sufficient address pool. However, the ongoing IPv6 transition seems to be a lengthy task because of the numerous challenges it faces. Therefore, it is expected that IPv4 and IPv6 will coexist for a long time. Consequently, many transition technologies have been developed for this purpose. Several research papers have conducted performance analysis for a number of these transition technologies and even compared them based on some measuring metrics like RTT, throughput, jitter, packet loss, and so on. This paper reviews the results of these papers, discusses their findings, and gives some guidelines fora feasible benchmarking methodology.

Keywords: IPv6 Transition Technologies; 464XLAT; Dual Stack; Tunneling; MAP-T/E

1. Introduction

The rapid growth of the Internet over the past few years has led to the actual depletion of IPv4 addresses in 2011 [1], which makes the adoption of IPv6 a necessity more than ever before. In addition to solving this problem, transitioning from IPv4 to IPv6 can help in solving many other problems such as the burden of NAT translation, the need for scalable routing tables, and the delay of the defragmentation process in the intermediary devices [2]. However, migrating the entire Internet from IPv4 to IPv6 is not an easy task and can take a considerably long

time as this will need to modify many technologies, protocols, services, and so on to work properly in the new IPv6 environment. Consequently, IPv4 and IPv6 will need to coexist for some time before IPv4 can be completely excluded. As IPv4 and IPv6 are not compatible with each other, numerous transition technologies have been developed by the IETF to help in the gradual yet smooth full adoption of IPv6 [3].

Many research papers have been conducted in literature aiming to compare IPv6 transition technologies and executed a significant analysis of their performance. This paper surveys the work done by some of these research papers to highlight the feasibility and effectiveness of the most important IPv6 transition technologies, which may depend on several network conditions and parameters.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the most important IPv6 transition technologies. Section 3 surveys the performance analysis of these IPv6 transition technologies accomplished by several research papers in the literature. Section 4 discusses the results of the work done by these surveyed research papers. Section 5 outlines future work. Section 6 concludes this paper.

2. IPv6 Transition Technologies

Many different IPv6 transition technologies have been developed to make communication between IPv4 and IPv6 networks possible and powerful. This paper follows the classification presented in [4] for these technologies, namely Dual Stack, encapsulation, single translation, and double translation technologies. It considers that the production network transitioning to IPv6 as being consisted of three IP domains, namely, domain A, core domain, and domain B. Domains A and B are IPvX-specific domains, while the core domain can be either IPvY-specific or Dual Stack (IPvX and IPvY) domain, where X and Y are part of the set (4,6) and X is not equal to Y.

2.1. Dual Stack

RFC 4213 [5] describes Dual Stack as a mechanism that supports both IPv4 and IPv6 by including both stacks at the same time in the network nodes, but one of them could be disabled and allow the other to communicate. However, this mechanism can result in significant communication delays. For instance, when a host wants to connect to a remote server, it initiates a DNS request asking for all available addresses of that remote server. The DNS server replies with both IPv4 and IPv6 address information (i.e. A and AAAA records). Then, the host may choose, by default, the IPv6 address to start the TCP session with. Once the IPv6 communication fails with the remote server, the host will switch to IPv4. Consequently, this causes latency in communication.

To alleviate the effects of this problem, a method of connection establishment called “Happy Eyeballs” has been proposed in [6] and improved in [7]. Its algorithm targets a better user experience by selecting the proper IP version for packet delivery after initiating multiple asynchronous connection attempts and then establishing the first successful one and cancelling all other attempts.

However, Dual Stack, even with “Happy Eyeballs”, introduces unnecessary complication as both IPv4 and IPv6 protocols should be maintained and makes the network nodes more vulnerable to security threats. Not to say that it does not participate in solving the problem of the depletion of IPv4 addresses [3].

2.2. Encapsulation

The transition mechanisms under this category encapsulate packets coming from IPvX-specific domain by the header of IPvY at the edge between IPvX-specific domain and IPvY-specific domain. Then, the IPvY encapsulated packets are de-encapsulated at the edge between the IPvY-specific domain and another IPvX-specific domain before being received by the nodes of the latter one [4].

2.2.1. Manual Tunneling

Manual Tunneling [5], also called Explicit Tunneling or Configured Tunneling, is the very basic idea of this type of transition technologies, where IPv6 hosts/sites are linked together throughout an IPv4 infrastructure by encapsulating the IPv6 traffic into IPv4 packets (in this case the value of the protocol type field of the IPv4 headershould be 41) at one tunnel endpoint and then decapsulating them at the other tunnel endpoint. Both endpoints have to be Dual Stack and configured manually by the network administrator. Thus, configuring this type of point-to-point tunnels requires additional administration efforts and doesn’t scale well. 6in4 technology is considered some sort of this type of tunneling.

2.2.2. 6to4

6to4 [8] overcomes the problems of 6in4 manual tunneling by allowing IPv6 hosts/sites, which have at least one public IPv4 address, to communicate with each other over an IPv4 network without explicit tunnel setup. It also supports the communication between IPv6 hosts/sites and native IPv6 domains via relay routers. To apply the automatic setup of one-to-many tunnels, 6to4 includes a predefined prefix (2002::/16) to distinct its packets over the public IPv4 network and embeds the public IPv4 address into the IPv6 address. However, as [9] mentioned, several problems with 6to4 were documented in [10] and [11].

2.2.3. Teredo

Teredo[12]enables IPv4 hosts that do not have a public IPv4 address, but rather are sitting behind NAT, to connect to IPv6 nodes by tunneling the IPv6 packets over UDP. To accomplish this task, Teredo deploys two types of devices: Teredo server and Teredo relay. The Teredo server is responsible for configuring the Teredo tunnel while the Teredo relay acts as an IPv6 router and is responsible for forwarding traffic to/from Teredo clients. Each Teredo host is assigned an IPv6 address starting with a special service prefix (2001:0000:/32). Teredo is mainly developed to provide a "last resort" option for nodes that want to connect to IPv6 Internet and no other IPv6 transition technology is rather deployed.

2.2.4. ISATAP

Intra-Site Automatic Addressing Protocol (ISATAP) [13] is mainly designed to connect Dual Stack nodes to IPv6 nodes via IPv4 networks and it deploys an automatic tunneling mechanism for this purpose. The ISATAP host forms a well-defined IPv6 address format from a predefined 64-bit prefix obtained from an ISATAP server, followed by a reserved interface identifier (::5efe), and ended with the 32-bit IPv4 address. It then uses this address for communication over the ISATAP network[14].

2.2.5. 6rd

IPv6 Rapid Deployment (6rd) [15] is derived from 6to4 technology. Instead of using the 6to4's well-known IPv6 prefix 2002::/16, rather it permits for each ISP to deploy its own prefix. This gives ISPs more control over their network. 6rd can be considered stateless as it relies upon an algorithmic mapping between IPv6 and IPv4 addresses, which enables IPv4 tunnel endpoints to be automatically determined from IPv6 prefixes.

2.2.6. Tunnel Broker

Tunnel Broker [16] is not considered a technology by itself, but rather a way of managing IPv4 tunnels for IPv6 hosts/sites by deploying dedicated servers for automatically processing tunnel requests coming from these IPv6 hosts/sites. It, therefore, aims to stimulate expanding the number of IPv6 interconnected hosts and allowing IPv6 network providers to supply easy access to their IPv6 networks.

2.2.7. DS-Lite

Dual Stack-Lite (DS-Lite) [17] facilitates the incremental deployment of IPv6 by decoupling it in the broadband service provider network and enables IPv4 address sharing by combining two well-known technologies: IP in IP, more specifically IPv4 in IPv6, and stateful Network Address and Port Translation (NAPT).

DS-Lite operates by deploying two different devices: The Basic Bridging BroadBand (B4) device and the Address Family Transition Router (AFTR). The B4 device can be either a directly connected host device or a Customer Premise Equipment (CPE), which acts as a home gateway for customers and it is supplied with a WAN interface that is provisioned only with IPv6 by the service provider. The B4 device encapsulates the customer's IPv4 traffic into the service provider's IPv6 traffic before sending it to its destination. When the AFTR device receives this traffic, it decapsulates the IPv4 embedded IPv6 traffic and performs a stateful NAPT44 [18] function to translate the decapsulated IPv4 payload into packets with public IPv4 source address before sending it to its intended destination. The reply packets will also traverse these devices, but now the devices perform reverse encapsulation/decapsulation processes. Fig. 1 depicts the architecture of DS-Lite.

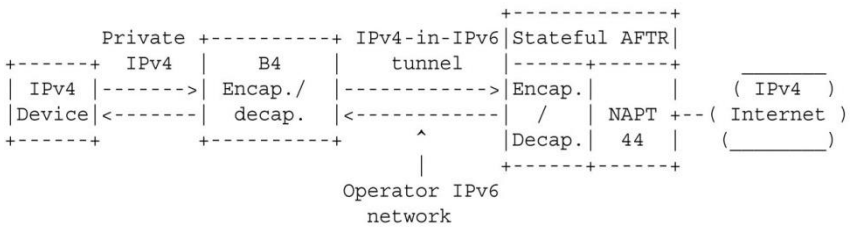

Figure 1. DS-Lite Architecture[19]

2.2.8. MAP-E

Mapping of Address and Port with Encapsulation (MAP-E)[20] uses a stateless algorithm which is based on designated rules that target configuring IPv4 address information (and also port information in case of shared IPv4 addresses) and embedding this information into the IPv6 prefix assigned to an end-user Customer Edge (CE) device when constructing the IPv6 packets during encapsulation at CE and then validate this information when decapsulating the arrived IPv6 packets at a Border Relay (BR) that belongs to the same MAP domain. However, these MAP rules are also used when encapsulating the reply packets at BR before forwarding them in the opposite direction.

A MAP domain virtually connects one or more CE and BR devices that share the same mapping rule set. A service provider can control one or more MAP domains. On the other hand, a single CE may be connected to more than one MAP domain if it has more than one IPv6-enabled interface. However, each one of these MAP domains utilizes its own mapping rule set that is different from those of other

domains. The communications inside one particular MAP domain (e.g. between two different CEs) can be either according to "mesh mode" (i.e. direct communication without the support of a BR) or according to "hub-and-spoke mode" (i.e. all traffic must be first forwarded to a BR belongs to the same MAP domain). In contrast, all communications with nodes from outside one particular MAP domain should be accomplished through one of the domain's BR devices. Fig. 2 depicts MAP-E architecture.

All CE devices typically perform stateful NAPT44 before encapsulating the IPv4 traffic into IPv6 packets using MAP rules and then send these IPv6 packets to their intended destination through the related MAP domain. When a BR receives traffic intended for a destination outside its MAP domain, it decapsulates the IPv6 header and validates IPv4 address (and port if necessary) information using the related MAP rules before sending the IPv4 payload to its intended destination.

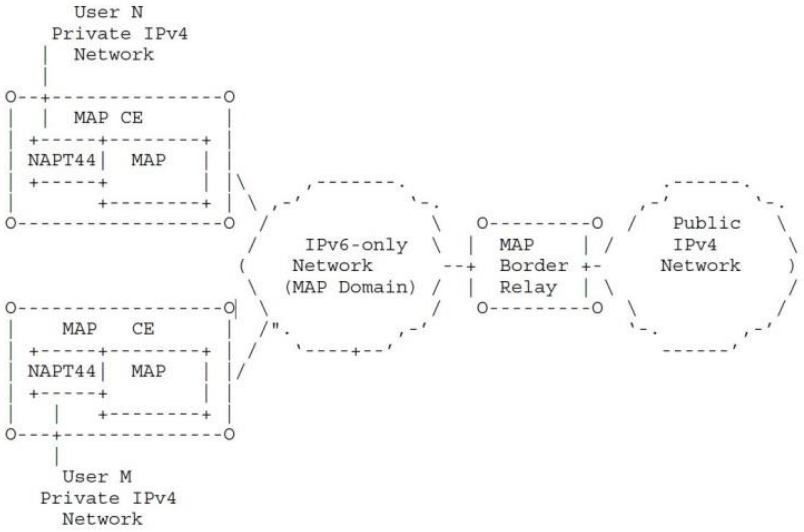

Figure 2. MAP-E Architecture[20]

2.2.9. Lw4o6

Lightweight 4over6 [21] is an extension to DS-Lite. In contrast to DS-Lite, lw4o6 relocates the NAPT function from the centralized AFTR (here it is rather called

lwAFTR) device to the distributed B4(here it is rather called lwB4) devices. This solution noticeably reduces the overhead of maintaining traffic states from per-flow to per-subscriber and thus logging overhead and also considerably relieves lwAFTR from being overloaded by translation tasks as it already has other tasks like encapsulation/decapsulation, softwire maintaining and lookup, and A+P routing. Fig.3 shows the architecture of lw4o6.

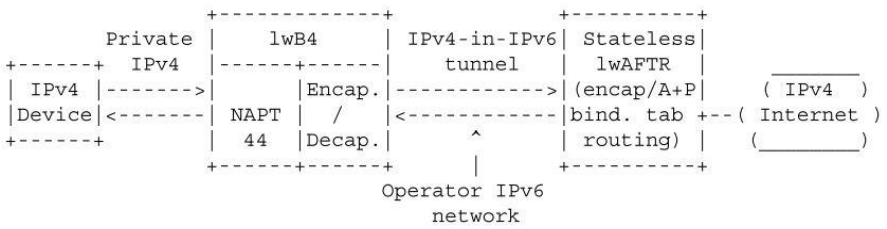

Figure 3. The architecture of lw4o6 [19]

In addition to creating a tunnel to a lwAFTR, the B4 device supports port-restricted IPv4 address allocation and performs stateful NAPT44 function, whereas lwAFTR represents the other endpoint of the tunnel and maintains the so-called softwires (i.e. two different IP versions binding entries) in an address binding table. Each entry in the binding table is constructed on a per-subscriber basis and belongs to a particular lwB4. The lwAFTR uses each entry of the table to implement two functions: 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 destination.

Lw4o6 also supports hair-pinning to enable direct communication between two lwB4s which are associated with the same lwAFTR.

Finally, lw4o6 maintains a provisioning mechanism for assigning specific IPv4 public address and port set for each lwB4 device. This assigned information should also be synched with that of the lwAFTR binding table.

2.3. Single Translation

The transition mechanisms under this category translate packets coming from IPvX-specific domain to packets that go to IPvY-specific domain and vice versa (i.e. reverse translation) at the edge between these two domains [4]. By this translation,

the communication between IPvX-only nodes and IPvY-only nodes becomes possible and feasible.

2.3.1. Stateful NAT64 along with DNS64

Although DNS64 [22] is not considered a single translation transition technology, it is discussed here along with stateful NAT64 [23] as they are commonly used together to enable IPv6-only clients to communicate with IPv4-only servers. It is quite appropriate to deploy this communication scenario instead of using the regular NAT444 (also called Carrier Grade NAT [24]) and practically transfer hosts to the IPv6 usage to shorten the transition period [9].

RFC 6146 [23] gives a behavior walk-through of NAT64 and DNS64 via a detailed example. To summarize the operation of this technology, consider the scenario depicted in Fig. 4. Here, an IPv6-only client wants to communicate to an IPv4-only server. It first sends a request to the DNS64 server asking for the IPv6 address (i.e. the AAAA record) of the **www.hit.bme.hu** web server. The DNS64 server asks the DNS system for this address. Since the DNS system has no such record, it will eventually respond with the corresponding IPv4 address (i.e. the A record), which is in this case 152.66.248.44. The DNS64 server uses the 64:ff9b::/96 NAT64 Well-Known Prefix [25] to synthesize the so-called *IPv4-embedded IPv6 address* by appending the received 32-bit IPv4 address (i.e. 152.66.248.44) and responds to the client with this synthesized address. The IPv6-only client starts the TCP session and sets the received *IPv4-embedded IPv6 address* (i.e. 64:ff9b::9842:f82c where the last 32-bit 0x9842f82c represents exactly the 152.66.248.44 IPv4 address) as the destination address of the IPv6 packet. As a prerequisite to this technology, all packets destined to a 64:ff9b::/96 prefixed address must be routed to the NAT64 gateway. Thus, the NAT64 gateway receives this packet via its IPv6 interface, constructs an IPv4 packet using the last 32-bit of the address (i.e. 152.66.248.44) as the destination IPv4 address and its IPv4 interface address as the source address of the IPv4 packet, registers the connection into its connection tracking table, and then sends the IPv4 packet to the IPv4-only web server. The server responds to the NAT64 gateway. The NAT64 gateway receives the IPv4 reply packet from the server, constructs an IPv6 packet from the relevant information using those registered in the state table, and sends this IPv6 packet to the IPv6-only client [26].

However, this technology still needs to implement Dual Stack at NAT64 gateways and DNS64 servers to work properly. Also, according to [27], it seems that this technology doesn’t work well with peer-to-peer applications.

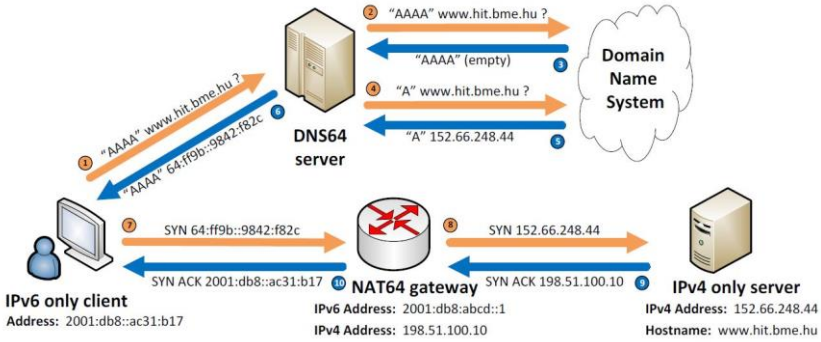

Figure 4. NAT64 and DNS64 scenario: An IPv6-only client communicates with an IPv4-only server [26]

2.3.2. SIIT

The Stateless IP/ICMP Translation (SIIT) technology [28] translates IPv4 to IPv6 packet headers (including ICMP headers) and vice versa by implementing an address mapping algorithm and based on the configuration of the translator and the being translated packet information. Additionally, the translator doesn't maintain any dynamic session or binding state. Therefore, packets in a single session or flow can traverse more than one translator instead of a single one as stateful translators do.

2.3.3. NAT-PT

RFC 2766 [29] defined this technology in 2000. But, then, it was moved to historic status for many reasons stated by authors of RFC 4966 [30]. Therefore, it is out of the interest of this paper even though its performance has been analyzed by many research papers.

2.4. Double Translation

The transition mechanisms under this category translate packets coming from IPvX-specific domain to packets that go through the IPvY-specific core domain (of the network operator) and then to another IPvX-specific domain. The first translation is accomplished at the edge between the first IPvX domain and the IPvY core domain, while the second translation is accomplished at the edge between the IPvY core domain and the second IPvX domain [4]. Subsequently, the reply packets can also be double translated in the reverse direction.

2.4.1. 464XLAT

This technology is clearly described in [31]. Since then, it was deployed by several network operators [32] such as the US mobile telecommunication company T-Mobile [33]. It enables the service providers to deploy IPv6-only devices for their infrastructure while keeping customers still running applications that use socket APIs and literal IPv4 addresses like Skype, Google Talk, and so on[32]. Thus, this technology is also considered an *IPv4aaS* technology [9].

464XLAT uses two different devices, namely, the CLAT and PLAT devices. The CLAT is located at the client-side and it performs stateless NAT46 translation for the private IPv4 packets received from IPv4 clients to global IPv6 packets sent over the ISP's IPv6 core network and later translates back the reply packets (i.e. it performs SIIT). The PLAT device is located at the ISP side and it performs stateful NAT64 translation for the global IPv6 packets received from the IPv6 core network to global IPv4 packets sent to the IPv4 server [31]. For a clearer picture of the architecture of this technology, see Fig.5.

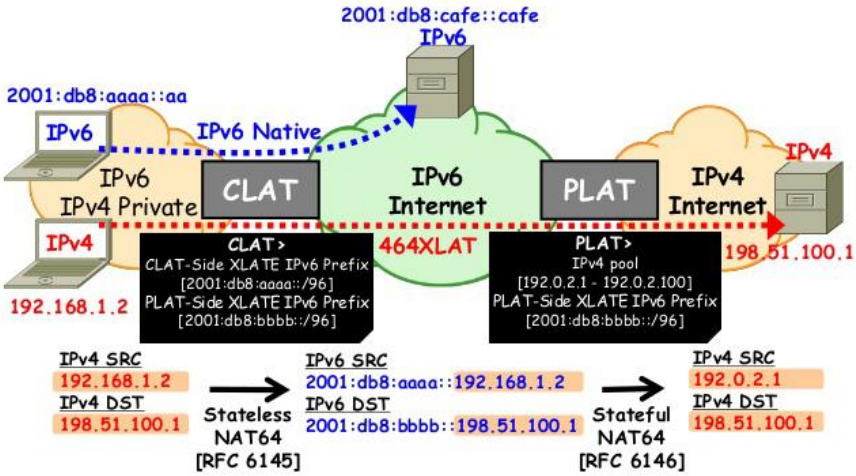

Figure 5. 464XLATArchitecture[34]

Although this technology is listed here as a double translation technology, that is when CLAT and PLAT translations are executed one after another, it can also

operate as a single translation technology. This happens when an IPv6 client wants to communicate to an IPv4 server. In this case, the only translation needed is the PLAT one. However, this means that the PLAT's stateful NAT64 will need DNS64 for name resolution. Here, the CLAT acts as a DNS proxy [9]. One more case is the no translation case, which can be happened when an IPv6 client wants to communicate to an IPv6 server. Here, the CLAT acts as a router for IPv6 traffic [9]. The three case scenarios are listed in Table 1.

Table 1. 464XLAT Traffic Treatment Scenarios[31]

Application and Host	Server	Traffic Treatment	Location of Translation
IPv6	IPv6	Native IPv6	None
IPv6	IPv4	Stateful NAT64 + DNS64	PLAT
IPv4	IPv4	464XLAT (i.e. SIIT/Stateful NAT64)	CLAT/PLAT

2.4.2. MAP-T

Mapping of Address and Port using Translation (MAP-T) [35] can be compared to MAP-E as it uses mapping rules and can be compared to 464XLAT as it uses a stateless double NAT64 translation rather than encapsulation. This can give MAP-T the advantage of using the strength of mapping in scenarios where encapsulation is being ruled out such as those presented in [35]. Thus, it targets diminishing the overhead of encapsulation and allowing IPv4 and IPv6 traffic to be treated as the same as possible.

MAP-T deploys the same technique as MAP-E in terms of using MAP rules. As shown in Fig. 6, its architecture is so identical to that of MAP-E and it uses similar terminology as MAP-E does (e.g. MAP domain, Customer Edge (CE) device, Border Relay (BR) router, and so on). However, it uses, of course, a double translation instead of encapsulation to enable IPv6-only connectivity to its operator network.

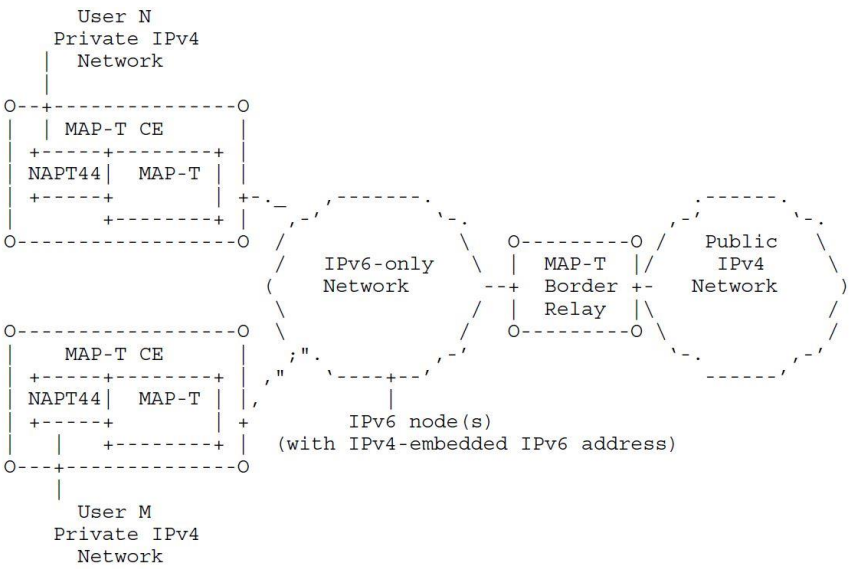

Figure 6. MAP-T Architecture[35]

3. Performance Analysis of IPv6 Transition Technologies

Several research papers have analyzed the performance of different IPv6 transition technologies in the literature. This section is intended to give a clear summary of their findings. However, there are some others which are not covered in this paper because they are either relatively old (e.g. [36], [37], and [38]) or out of the scope of this paper such as those measuring or comparing the performance of different implementations of a single technology (e.g. [39] for DNS64 implementations, [40] for NAT64 implementations, and [41] for 6to4 implementations).

Altangerel et al. [2] performed performance analysis on some IPv6 transition technologies, namely Dual Stack, manual tunneling, ISATAP, and 6to4. Their experiments were implemented in a real network environment by transmitting 30,000 TCP segments of size of approximately 31,000 bytes generated by Poisson distribution in MATLAB. They applied these different technologies to a single user and also to a varying number of users. Then, they calculated the average results of both RTT and throughput obtained from the two cases. They also used GNS3 and OPNET simulators to complete their comparison and analysis of the before mentioned technologies. The experiments showed that both manual tunneling and 6to4 proved higher throughput and lower RTT whereas Dual Stack was the worst as it resulted in the minimum throughput and the maximum RTT.

Georgescu et al. [42] proposed an IPv6 Network Evaluation Testbed (IPv6NET) as a measuring tool to test the feasibility of transition technologies in a series of scenario-based network situations. The article took one of the enterprise network scenarios, which has been introduced by the IETF in [43], as a case study for their research. The scenario targets enterprises using one of the *IPv4aaS* technologies, where the core network is IPv6-only while the end-user and server nodes are IPv4-capable. Thus, they selected 464XLAT and MAP-T as translation-based technologies and MAP-E and DS-Lite as encapsulation-based technologies to implement this scenario. Additionally, the authors selected the free software *Asamap/Vyatta Distribution* [44]as an implementation, which supports the four chosen technologies. The authors also used the Distributed Internet Traffic Generator (D-ITG) described in [45] to generate traffic for their experiments.

As an empirical methodology, IPv6NET tested two types of environments: closed environments for thorough network performance data and open environments for operational capability data. To quantify closed environment results, well-established metrics such as *round-trip delay*, *jitter*, *throughput*, and *packet loss* have been used. For open environment testing, an operational capability indicator has been proposed, which states how much a certain technology is suitable for a certain environment or how it can successfully pass operational problems. For this purpose, three different metrics have been introduced, *configuration capability*, which measures how much a network implementation is capable of in the context of contextual configuration or reconfiguration, *troubleshooting capability*, which measures how much a network implementation is capable of in terms of isolating and identifying faults, and *applications capability*, which measures how much a device is capable at ensuring compatibility with well-known end-user protocols.

The empirical results of IPv6NET experiments indicated that MAP-E showed the best network performance over other tested technologies. Moreover, the translation-based technologies (464XLAT and MAP-T) had better performance in terms of latency, whereas the encapsulation-based technologies (MAP-E and DS-Lite) had better performance in terms of throughput. However, the authors asserted that the results are greatly dependent on the quality of the deployed implementation. Thus, their results should be coupled with the quality of the Asamap implementation that they have chosen for their experiments.

Hossain et al. [46] executed Dual Stack, 6to4, and NAT-PT in the Packet Tracer simulator and analyzed their performance through three metrics: latency, throughput, and packet loss. They concluded that 6to4 had the best performance as its latency and packet loss were the lowest and its throughput was the highest. In contrast, NAT-PT was the worst as it showed the poorest performance in terms of the three metrics.

Quintero et al. [47] compared the performance of ISATAP, 6to4, and NAT64 in real testbeds executed in various operating systems. They measured the One Way Delay (OWD) and throughput for TCP and UDP traffic of each one of the measured technologies. To calculate OWD, they used the benchmark described in [48], and to calculate throughput, they used Iperf [49]. Their research confirmed that generally, both ISATAP and 6to4 have similar performance, whereas NAT64 showed the best performance in terms of OWD and throughput for both TCP and UDP traffic. One simple exception is the case of measuring OWD for UDP traffic of TAYGA-NAT64 as it showed unexpected high values.

Sookun et al. [14] evaluated the performance of Dual Stack, 6to4, ISATAP, and 6rd upon three measurements: the Round Trip Time (RTT), throughput, and CPU usage. For this purpose, they installed their testbeds using the GNS3 simulator. They found that Dual Stack recorded the best performance in terms of the highest throughput and the lowest RTT, followed by 6rd, whereas 6to4 recorded the worst performance. Additionally, Dual Stack recorded the highest CPU usage as it has to maintain two routing tables while the other technologies had almost the same CPU usage.

Singalar et al. [50]simulated the function of three technologies: Dual Stack, 6in4, and NAT-PT using the packet tracer simulator. They measured their results against RTT and throughput. They concluded that Dual Stack produced the highest throughput and the lowest RTT. Hence, the Dual Stack was the best. Conversely, NAT-PT had the poorest performance in terms of all measured metrics.

Yu et al. [51] analyzed the performance of both NAT-PT and NAT64. For NAT-PT, they used the open-source *naptd* daemon in conjunction with DNS-ALG for DNS64, and for NAT64, they used *Ecdysis* implementation [52] in conjunction with the DNS64 synthesizing resolver [22] for DNS64. The study showed that NAT64 is relatively more efficient than NAT-PT, especially for small networks or when a network has only a native IPv6-only connection to an IPv4-only server and no other transition technology deployed. The only exception is in the case of a network that has a considerable amount of large outbound packets and few simultaneous connections.

Sans et al. [54] compared the performance of ISATAP,6to4,6rd, and Teredo in terms of RTT and the throughput for both TCP and UDP over both Ethernet and Fast Ethernet using a real testbed. To calculate RTT, they used the benchmark described in [48], and to calculate throughput, they used Iperf[49]. It can also be noted that the authors deployed a free implementation of Teredo called Miredo [57]to implement Teredo. Their experiments showed that ISATAP presents the best performance over all other tested technologies.

Table 2. Summary of technologies covered by each research paper

IPv6 Transition Technology		***Surveyed Research Paper***										
		[2]	**[42]**	**[46]**	**[47]**	**[14]**	**[50]**	**[51]**	**[53]**	**[54]**	**[55]**	**[56]**
Dual Stack		✓		✓		✓	✓				✓	
Encapsulation	Manual Tunneling	✓					✓				✓	✓
	6to4	✓		✓	✓	✓			✓	✓	✓	✓
	Teredo									✓		
	ISATAP	✓			✓	✓			✓	✓		
	6rd					✓				✓		
	Tunnel Broker											
	DS-Lite		✓									
	MAP-E		✓									
	Lw4o6											
Single Translation	NAT64+ DNS64				✓			✓				
	SIIT											
	NAT-PT			✓			✓	✓				
Double Translation	464XLAT		✓									
	MAP-T		✓									

Shah et al. [55] examined the performance of Dual Stack, 6to4, and 6in4 through three metrics: throughput, response time, and TCP delay by using the OPNET simulator. The results showed that 6to4 had better performance than the others, while Dual Stack was the worst.

Hadiya et al. [56] setup real testbed experiments to analyze the performance of both 6to4 and configured tunneling and compare them against different metrics like

throughput, jitter, and delay. They found that 6to4 gives better performance than configured tunneling.

Table 2 gives a summary of which IPv6 transition technologies were measured by each one of the surveyed research papers. Table 3 and Table 4 explain the measurement details of each surveyed research paper.

Table 3. Measurement details of the surveyed research papers – part 1

Surveyed research paper	*Test method*	*Tested technology*	*Technology implementation*	*Measured metrics*	*Traffic generation method*
[2]	Real testbed + Simulation (GNS3 & OPNET)	Dual Stack Manual Tunneling ISATAP 6to4	Unspecified for all tested technologies	RTT and throughput	Poison distribution by MATLAB
[42]	Real testbed and by using a proprietary tool (IPv6NET)	464XLAT MAP-T MAP-E DS-Lite	AsamapVyatta Distribution for all tested technologies	RTT, jitter, throughput, and packet loss	D-ITG
[46]	Simulation (Packet Tracer)	Dual Stack 6to4 NAT-PT	Packet Tracer's implementation	Latency, throughput, and packet loss	Packet Tracer's PDU generator
[47]	Real testbed	ISATAP 6to4 NAT64+ DNS64	Daemon Daemon TAYGA & JOOL	One Way Delay (OWD) and throughput	Proprietary for OWD &Iperf for throughput
[14]	Simulation (GNS3)	Dual Stack 6to4 ISATAP 6rd	GNS3's implementation	RTT, throughput, and CPU usage	GNS3's packet generator

Table 4. Measurement details of the surveyed research papers – part 2

Surveyed research paper	*Test method*	*Tested technology*	*Technology implementation*	*Measured metrics*	*Traffic generation method*
[50]	Simulation (Packet Tracer)	Dual Stack	Packet Tracer's implementation	RTT and throughput	Packet Tracer's PDU generator
		6in4			
		NAT-PT			
[51]	Real testbed	NAT-PT	Daemon	RTT	HTTP packets from the client and the server
		NAT-PT's DNS64	DNS-ALG		
		NAT64	Ecdysis		
		NAT64's DNS64	DNS64 Synthesizing Resolver		
[53]	Real testbed	ISATAP	Unspecified	Throughput, EtoE Delay, RTT, and jitter	UDP audio & video streams and ICMP-Ping
		6to4	Unspecified		
[54]	Real testbed	ISATAP	Daemon for PCs and configuration for routers	RTT and throughput	Proprietary for RTT &Iperf for throughput
		6to4	Router configuration		
		6rd	Router configuration		
		Teredo	Miredo		
[55]	Simulation (OPNET)	Dual Stack	OPNET's implementation	Throughput, response time, and TCP delay	OPNET's HTTP traffic generator
		6to4			
		6in4			
[56]	Real testbed	6to4	Unspecified	Throughput, jitter, and delay	D-ITG
		Manual Tunneling	Unspecified		

Discussion

As it may be noticed, no research paper has revealed a remarkable bias that one of the IPv6 transition technologies is adequate for all network environments and situations. It is quite obvious that the results of their performance analysis of IPv6 transition technologies were distinct from each other. This leads us to the fact that their results were highly dependent on many factors including, but not limited to, the followings:

- The selected IPv6 transition technologies have been tested in their experiments.
- The type of their tests (i.e. real testbeds, simulations, etc.).
- The tools they used for testing such as:
  - The type and number of equipment.
  - The chosen simulator, in case of simulations (e.g. GNS3, OPNET, Packet Tracer, etc.).
  - The software implementation of the IPv6 transition technology (e.g. TAYGA, Jool, Asamap, system daemons like *isatapd*, etc.)
  - The type of operating system of the PCs used in the experiments (Linux, Windows 10, etc.).
- The network topology of the experiments.
- The values of configuration parameters of the experiments.
- The number of running iterations of the experiments.
- The form of the applied scenarios in the experiments.

Therefore, deciding which technology is the best for transitioning is a challenging question. The answer may change as the situation or type of network changes. But, at least, a standard benchmarking methodology is essentially needed for testing IPv6 transition technologies. This already has been done through the IETF's RFC 8219 [4], which mainly targets encapsulation and translation technologies. Furthermore, the IETF's RFCs 2544 [58] and RFC 5180 [59] give recommendations for testing Dual Stack. So, they can be used as guides for building compliant testers for various IPv6 transition technologies.

4. Future Work

As stated earlier, the IPv4aaS technologies use IPv6-only devices in the core layer while keep providing IPv4 services for customers at their access layer. This gives the network operators the opportunity to accomplish a gradual yet smooth transition

to IPv6 in their core network while allowing customers to still run applications that use socket APIs and literal IPv4 addresses like Skype, Google Talk, and so on and then easily plug in their devices to IPv6 once it becomes necessary. As [19] stated, the five most prominent technologies that are considered IPv4aaS ones are: 464XLAT [31], DS-Lite [17], lw4o6 [21], MAP-E [20], and MAP-T [35].

Building an RFC 8219 compliant tester for each one of the IPv4aaS technologies could be promising. There are some of the steps that have already been taken in this direction. For instance, siitperf [60] is an 8219 compliant tester for benchmarking stateless NAT64 (also known as SIIT), which is also the CLAT part of 464XLAT. Also, dns64perf++ [61] is also an 8219 compliant tester but for benchmarking DNS64. Moreover, a DS-Lite tester is now being developed under the project of 6transperf [62].

As for future work, we are planning to develop an RFC8219 compliant tester for the other IPv4aaS technologies like lw4o6, MAP-E, and MAP-T in conjunction with typical benchmarking measurements.

5. Conclusion

Many IPv6 transition technologies have been developed over the past few years to mitigate the effect of the problem of IPv4 depletion and help in taking serious steps headed for the full adoption of the successor IP version, IPv6. Several research papers have analyzed the performance of these technologies and compared them in the literature. Their methodology of testing varies in terms of the type of testbed, technology's implementation method, traffic generation method, and number and type of metrics (e.g. RTT, throughput, jitter, packet loss, and so on). This paper surveyed these papers, discussed their results, and gave some guidelines fora standardized benchmarking methodology.

References

[1] A. Al-Azzawi, Towards the security analysis of the five most prominent IPv4aaS technologies, *Acta Technica Jaurinensis* 13 (2) (2020) pp. 85–98. doi: *http://doi.org/10.14513/actatechjaur.v13.n2.530*

[2] G. Altangerel, E. Tsogbaatar, D. Yamkhin, Performance analysis on IPv6 transition technologies and transition method, in: 2016 11th International Forum on Strategic Technology (IFOST), 2016, pp. 465–469. doi: *https://doi.org/10.1109/IFOST.2016.7884155*

[3] A. S. Ahmed, R. Hassan, N. E. Othman, Security threats for IPv6 transition strategies: A review, in: 2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T),2014, pp. 83–88. doi: https://doi.org/10.1109/ICE2T.2014.7006224

[4] M. Georgescu, L. Pislaru, G. Lencse, Benchmarking ethodology for IPv6 transition technologies, IETF RFC 8219 (2017). doi: https://doi.org/10.17487/RFC8219

[5] E. Nordmark, R. Gilligan, Basic transition mechanisms for IPv6 hosts and routers, IETF RFC 4213 (2005). doi: https://doi.org/10.17487/RFC4213

[6] D. Wing, A. Yourtchenko, Happy eyeballs: Success with dual-stack hosts, IETF RFC 6555 (2012). doi: https://doi.org/10.17487/RFC6555

[7] D. Schinazi, T. Pauly, Happy eyeballs version 2: Better connectivity using concurrency, IETF RFC 8305 (2017). doi: https://doi.org/10.17487/RFC8305

[8] B. Carpenter, K. Moore, Connection of IPv6 domains via IPv4 clouds, IETF RFC 3056 (2001). doi: https://doi.org/10.17487/RFC3056

[9] G. Lencse, Y. Kadobayashi, Comprehensive survey of IPv6 transition technologies: A subjective classification for security analysis, *IEICE Transactions on Communications* 102 (10) (2019) pp. 2021–2035. doi: http://doi.org/10.1587/transcom.2018EBR0002

[10] B. Carpenter, Advisory guidelines for 6to4 deployment, IETF RFC 6343 (2011). doi: https://doi.org/10.17487/RFC6343

[11] E. Aben, 6to4 - How bad is it really?, RIPE NCC [cited 2020-11-9]. URL https://labs.ripe.net/Members/emileaben/6to4-how-bad-is-it-really

[12] C. Huitema, Teredo: Tunneling IPv6 over UDP through network address translations (NATs), IETF RFC 4380 (2006).

doi: https://doi.org/10.17487/RFC4380

[13] F. Templin, T. Gleeson, D. Thaler, Intra-site automatic tunnel addressing protocol (ISATAP), IETF RFC 5214 (2008).
doi: https://doi.org/10.17487/RFC5214

[14] Y. Sookun, V. Bassoo, Performance analysis of IPv4/IPv6 transition techniques, in: 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech), 2016, pp. 188–193.
doi: https://doi.org/10.1109/EmergiTech.2016.7737336

[15] W. Townsley, O. Troan, IPv6 rapid deployment on IPv4 infrastructures (6rd) -- Protocol Specification, IETF RFC 5969 (2010).
doi: https://doi.org/10.17487/RFC5969

[16] A. Durand, P. Fasano et al., IPv6 tunnel broker, IETF RFC 3053 (2001).
doi: https://doi.org/10.17487/RFC3053

[17] A. Durand, R. Droms et al., Dual-stack lite broadband deployments following IPv4 exhaustion, IETF RFC 6333 (2011).
doi: https://doi.org/10.17487/RFC6333

[18] P. Srisuresh, M. Holdrege, IP network address translator (NAT) terminology and considerations, IETF RFC 2663 (1999).
doi: https://doi.org/10.17487/RFC2663

[19] G. Lencse, J. P. Martinez et al., Pros and cons of IPv6 transition technologies for IPv4aaS, active Internet Draft, 2020.[cited 2020-11-15].
URL https://tools.ietf.org/html/draft-lmhp-v6ops-transition-comparison-05

[20] E. O. Troan, W. Dec et al., Mapping of address and port with encapsulation (MAP-E), IETF RFC 7597 (2015).
doi: https://doi.org/10.17487/RFC7597

[21] Y. Cui, Q. Sun et al., Lightweight 4over6: An extension to the dual-stack lite architecture, IETF RFC 7596 (2015).
doi: https://doi.org/10.17487/RFC7596

[22] M. Bagnulo, A. Sullivan et al., DNS64: DNS extensions for network address translation from IPv6 clients to IPv4 servers, IETF RFC 6147 (2011).
doi: https://doi.org/10.17487/RFC6147

[23] M. Bagnulo, P. Matthews, I. V. Beijnum, Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers, IETF RFC 6146 (2011).
doi: https://doi.org/10.17487/RFC6146

[24] E. S. Perreault, I. Yamagata et al., Common requirements for carrier-grade NATs (CGNs), IETF RFC 6888 (2013).
doi: https://doi.org/10.17487/RFC6888

[25] C. Bao, C. Huitema et al., IPv6 addressing of IPv4/IPv6 translators, IETF RFC 6052 (2010).
doi: https://doi.org/10.17487/RFC6052

[26] G. Lencse, A. Soós, Design, implementation and testing of a tiny multi-threaded DNS64 server, International *Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems* 5 (2) (2016) pp. 68–78.
doi: http://doi.org/10.11601/ijates.v5i2.129

[27] S. Répás, T. Hajas, G. Lencse, Application compatibility of the NAT64 IPv6 transition technology, in: 2015 38th International Conference on Telecommunications and Signal Processing (TSP), 2015, pp. 1–7.
doi: https://doi.org/10.1109/TSP.2015.7296383

[28] C. Bao, X. Li et al., IP/ICMP translation algorithm, IETF RFC 7915 (2016).
doi: https://doi.org/10.17487/RFC7915

[29] G. Tsirtsis, P. Srisuresh, Network address translation - protocol translation (NAT-PT), IETF RFC 2766 (2000).
doi: https://doi.org/10.17487/RFC2766

[30] C. Aoun, E. Davies, Reasons to move the network address translator - protocol translator (NAT-PT) to historic status, IETF RFC 4966 (2007).
doi: https://doi.org/10.17487/RFC4966

[31] M. Mawatari, M. Kawashima, C. Byrne, 464XLAT: Combination of stateful and stateless translation, IETF RFC 6877 (2013).
doi: https://doi.org/10.17487/RFC6877

[32] J. Palet, Using 464XLAT in residential networks, RIPE 74 (2017) [cited 2020-11-11].
URL https://ripe74.ripe.net/presentations/151-ripe-74-ipv6-464xlat-residential-v2.pdf

[33] A. McConachie, Case Study: T-Mobile US goes IPv6-only using 464XLAT, Internet Society (2014) [cited 2020-11-13].
URL
https://www.internetsociety.org/resources/deploy360/2014/case-study-t-mobile-us-goes-ipv6-only-using-464xlat/

[34] M. Mawatari, 464XLAT tutorial, Japan Internet Exchange Co., Ltd. (APNIC 40) (2015) [cited 2020-11-15].
URL https://www.slideshare.net/apnic/464xlat-tutorial

[35] X. Li, C. Bao et al., Mapping of address and port using translation (MAP-T), IETF RFC 7599 (2015).
doi: https://doi.org/10.17487/RFC7599

[36] I. Raicu, S. Zeadally, Evaluating IPv4 to IPv6 transition mechanisms, in: 10th International Conference on Telecommunications, 2003. ICT 2003., 2003, pp. 1091–1098 vol.2.
doi: https://doi.org/10.1109/ICTEL.2003.1191589

[37] M. Shin, H. Kim et al., An empirical analysis of IPv6 transition mechanisms, in: 2006 8th International Conference Advanced Communication Technology, 2006, pp. 6 pp.-1996.
doi: https://doi.org/10.1109/ICACT.2006.206385

[38] S. Narayan, S. Tauch, IPv4-v6 transition mechanisms network performance evaluation on operating systems, in: 2010 3rd International Conference on Computer Science and Information Technology, 2010, pp. 664–668.
doi: https://doi.org/10.1109/ICCSIT.2010.5564141

[39] G. Lencse, S. Répás, Performance analysis and comparison of four DNS64 implementations under different free operating systems, *Telecommunication Systems* 63 (4) (2016) pp. 557–577.

doi: http://doi.org/10.1007/s11235-016-0142-x

[40] S. R. Répás, P. Farnadi, G. Lencse, Performance and stability analysis of free NAT64 implementations with different protocols, *Acta Technica Jaurinensis* 7 (4) (2014) pp. 404–427.
doi: http://doi.org/10.14513/actatechjaur.v7.n4.340

[41] G. Lencse, S. Répás, Performance analysis and comparison of 6to4 relay implementations, *International Journal of Advanced Computer Science and Applications* (IJACSA) 4 (9) (2013)
doi: http://doi.org/10.14569/IJACSA.2013.040903

[42] 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 2 (2) (2015)
doi: http://doi.org/10.4108/inis.2.2.e1

[43] E. J. Bound, IPv6 enterprise network scenarios, IETF RFC 4057 (2005).
doi: https://doi.org/10.17487/RFC4057

[44] M. Asama, Map supported vyatta, Echigo Network Operators' Group [cited 2020-11-7].
URL https://enog.jp/~masakazu/vyatta/map/

[45] A. Botta, A. Dainotti, A. Pescapé, A tool for the generation of realistic network workload for emerging networking scenarios, *Computer Networks* 56 (15) (2012) pp. 3531–3547.
doi: https://doi.org/10.1016/j.comnet.2012.02.019

[46] M. A. Hossain, D. Podder et al., Performance analysis of three transition mechanisms between IPv6 network and IPv4 network: Dual Stack, Tunneling and Translation, *International Journal of Computer* (IJC) 20 (1) (2016) pp. 217–228.

[47] A. Quintero, F. Sans, E. Gamess, Performance evaluation of IPv4/IPv6 transition mechanisms, *International Journal of Computer Network and Information Security* 8 (2) (2016) pp. 1–14.
doi: http://doi.org/10.5815/ijcnis.2016.02.01

[48] K. Velásquez, E. Gamess, A Survey of network benchmark tools, in: Machine Learning and Systems Engineering. Dordrecht, Springer Netherlands, 2010, pp. 465–480.

[49] J. Dugan, S. Elliott et al., Iperf homepage, NLANR/DAST [cited 2020-11-8].
URL *https://iperf.fr/*

[50] S. Singalar, R. M. Banakar, Performance analysis of IPv4 to IPv6 transition mechanisms, in: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1–6.
doi: *https://doi.org/10.1109/ICCUBEA.2018.8697539*

[51] S. Yu, B. Carpenter, Measuring IPv4–IPv6 translation techniques, Department of Computer Science, The University of Auckland, Tech. Rep 1 ((2012)

[52] Ecdysis: open-source implementation of a NAT64 gateway, Viagénie [cited 2020-11-15].
URL *http://ecdysis.viagenie.ca/index.html*

[53] M. Aazam, A. M. Syed et al., Evaluation of 6to4 and ISATAP on a test LAN, in: 2011 IEEE Symposium on Computers & Informatics, 2011, pp. 46–50.
doi: *https://doi.org/10.1109/ISCI.2011.5958881*

[54] F. Sans, E. Gamess, Analytical performance evaluation of native IPv6 and several tunneling technics using benchmarking tools, in: 2013 XXXIX Latin American Computing Conference (CLEI), 2013, pp. 1–9.
doi: *https://doi.org/10.1109/CLEI.2013.6670610*

[55] J. L. Shah, J. Parvez, An examination of next generation IP migration techniques: Constraints and evaluation, in: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014, pp. 776–781.
doi: *https://doi.org/10.1109/ICCICCT.2014.6993064*

[56] D. Hadiya, R. Save, G. Geetu, Network performance evaluation of 6to4 and configured tunnel transition mechanisms: An empirical test-bed analysis, in: 2013 6th International Conference on Emerging Trends in Engineering and Technology, 2013, pp. 56–60.

doi: https://doi.org/10.1109/ICETET.2013.14

[57] R. Denis-Courmont, Miredo: Teredo IPv6 tunneling for Linux and BSD, Remlab (2020) [cited 2020-11-7].
URL https://www.remlab.net/miredo/

[58] S. Bradner, J. McQuaid, Benchmarking methodology for network interconnect devices, IETF RFC 2544 (1999).
doi: https://doi.org/10.17487/RFC2544

[59] C. Popoviciu, A. Hamza et al., IPv6 benchmarking methodology for network interconnect devices, IETF RFC 5180 (2008).
doi: https://doi.org/10.17487/RFC5180

[60] G. Lencse, Design and implementation of a software tester for benchmarking stateless NAT64 gateways, *IEICE Transactions on Communications*, E104-B (2) (2021) pp. 128–140.
doi: http://doi.org/10.1587/transcom.2019EBN0010

[61] G. Lencse, D. Bakai, Design and implementation of a test program for benchmarking DNS64 servers, *IEICE Transactions on Communications* E100-B (6) (2016) pp. 948–954.
doi: http://doi.org/10.1587/transcom.2016EBN0007

[62] G. Lencse, 6transperf: Implementation of a benchmarking program for IPv6 transition technologies, Budapest University of Technology and Economics (2020) [cited 2020-11-5].
URL https://www.hit.bme.hu/edu/project/data?id=19179