Reflections on Middlebox Detection Mechanisms in IPv6 Transition

Aaron Yi Ding*[†], Jouni Korhonen[‡], Teemu Savolainen[§], Yanhe Liu*, Markku Kojo*, Sasu Tarkoma*, Henning Schulzrinne[†]

*University of Helsinki [†]Columbia University [‡]Broadcom [§]Nokia

Abstract. Middleboxes performing Network Address Translation (NAT) have gained a positive role in the IPv6 transition by enabling interoperability of different IP versions, i.e., IPv4 and IPv6. Although the NAT-based mechanisms pave the way for a smooth adoption of IPv6 in the transition, we shall be prudent towards such middlebox-oriented approach due to its complexity and overhead for end users and network services. In this position paper, we discuss the middlebox detection mechanisms designed for IPv6 transition, focusing on NAT64 in specific, and share our hands-on experience in protocol design and standardization. Based on our experimental findings, we identify issues in the existing approaches and offer suggestions to protocol designers and researchers. The goal is to improve our understanding of the middleboxes and hence leading to a more robust design of protocols for the evolving IPv6 Internet.

1 Introduction

In the transitional phase to IPv6, ensuring the interoperability between IPv4 and IPv6 is the key to a smooth adoption of IPv6 in the Internet [1]. Among various proposals, the once criticized Network Address Translation (NAT) mechanism, typically deployed on middleboxes, has made a positive impact to bridge the gap between those two incompatible IP versions.

As indicated in recent studies [2, 3], the IPv6 adoption has grown steady on a global scale. A lot of this growth can be accredited to the transitional technologies deployed by network operators, such as NAT64 [4] combined with DNS64 [5]. As shown in Figure 1 on an exemplary scenario in cellular access, middleboxes deploying NAT64 and DNS64 can enable end hosts in an IPv6-only network to access services in an IPv4-only network.

Despite the fast growth, middleboxes deploying such NAT-based solution have created connectivity issues for several services such as gaming and peer-to-peer applications that rely directly on IPv4-literal rather than the addresses from DNS responses [6]. Although 464XLAT [7] provides a solution on the architecture level, there is a strong need for detection mechanisms to discover the presence of middleboxes and learn the corresponding IPv6 context so that end hosts can conduct address synthesis by themselves. To solve the challenge, we start the WiBrA project [8] to investigate this topic together with industrial partners,

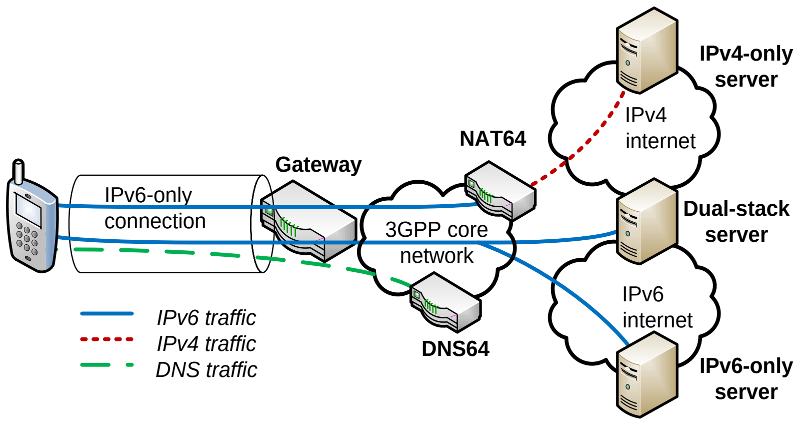

Fig. 1: NAT64 and DNS64 in Cellular Access [9]

focusing on mechanisms to detect NAT middleboxes and derive the IPv6 prefixes used in the access network.

In this position paper, we present our research outcome with the following contributions:

1. We present a study of middlebox detection mechanisms through our standardization experience to show how the design principle and practice actually work together in the real world.
2. We identify issues in the existing approaches and share our experimental findings. By implementing enhanced security features for our proposal [10], we run performance tests over four popular open-source NAT64 platforms. We spot a performance issue that provides valuable hints for protocol designers, middlebox vendors and researchers.
3. Based on our observations, we provide suggestions on how to cope with middleboxes in the transitional phase and beyond.

In the rest of this paper, Section 2 covers the NAT64 detection mechanisms. Section 3 presents our analysis, implementation and standardization experience. Section 4 illustrates the experimental findings. We discuss open issues and provide suggestions in Section 5, and conclude the paper in Section 6.

2 Detecting NAT64 Middleboxes during the IPv6 Transition

Following the fast migration to IPv6 in the access infrastructure [11], the deployed NAT64 middleboxes have enabled the connectivity to IPv4-only services from IPv6 networks. This trend encourages more users to adopt IPv6 as their first choice. Meanwhile, due to the inherited limitation of NAT, several services such as gaming still suffer from connection issues [6]. Because the problem comes from the lack of IPv6 context used by the NAT64 middleboxes, i.e., IPv6 prefixes [9, 12], it is thereby important to detect the presence of middlebox and derive such context from the access network to improve service experience.

As shown in Table 1, we highlight the IETF proposals for detecting NAT64 and learning IPv6 prefixes.

Table 1: Proposals for Detecting NAT64 Middlebox for IPv6 Prefix

Proposal	Network layer	Proposed date	Last update	Iteration times	Current status
Heuristic	Independent	09.2010	11.2013	20	RFC 7050
EDNS0 option	Application	07.2010	02.2011	2	expired draft
DNS A64	Application	10.2009	09.2010	2	expired draft
DNS TXT	Application	10.2008	10.2009	4	expired draft
DNS U-NAPTR	Application	03.2009	10.2009	3	expired draft
PCP	Application	09.2012	05.2014	10	RFC 7225
DHCPv6	Network	05.2009	12.2009	1	expired draft
RA	Network	05.2009	07.2009	1	expired draft

The heuristic discovery [10] and EDNS0 option [13] are proposals made in the WiBrA project [8]. The heuristic discovery is a generic approach that is independent of the incumbent implementations. The procedure is straight forward: a node makes a DNS query for an AAAA record against a well-known IPv4 only name, i.e., *ipv4only.arpa*, and by receiving an IPv6 address in the response, the node is able detect the presence of NAT64/DNS64 middlebox in its access network.

The EDNS0 approach [13] is based on Extension Mechanisms for DNS (EDNS0) [14]. It proposes a new option to indicate the presence of middleboxes and convey the IPv6 context. To support this mechanism, DNS64 servers shall implement the protocol and insert a new EDNS0 option in the synthesized AAAA Resource Records. An end host should also implement this protocol.

The DNS A64 approach [15] proposes a new Resource Record, namely A64, to store the synthetic IPv6 address. When an end host sends a query asking for this dedicated A64 record, by receiving a positive response, it can distinguish the record from other native IPv6 addresses and hence detecting the middlebox. This approach requires both the DNS servers and end hosts to support the protocol.

The DNS TXT and DNS U-NAPTR [16] are both DNS based solutions. The DNS TXT uses the existing TXT Resource Record to convey a string that contains the IPv6 unicast address and the prefix length used by NAT64. The DNS U-NAPTR uses a proposed U-NAPTR application to send a U-NAPTR query similar to the reverse DNS query. This query contains the IPv6 address of the host in the *.ip6.arpa* tree, and the response to a successful query will contain the unicast IPv6 address and the prefix length of NAT64. These two approaches demand no changes on the network side or the end host network stack.

The PCP-based mechanism [17] relies on Port Control Protocol (PCP) [18] and defines a new PCP option to learn IPv6 prefixes used by PCP enabled NAT64 middleboxes. Thus, PCP needs to be deployed for this approach to work.

The DHCPv6 mechanism [19] proposes a new DHCPv6 option to learn the IPv6 context used by middleboxes in the general IP configuration phase. It requires modifications to DHCP servers and end hosts.

The RA mechanism [20] adds a new option to the IPv6 Router Advertisement (RA) to convey IPv6 context including IPv6 unicast prefix, IPv6 any-source multicast prefix, source-specific multicast prefix, and the prefix length used by the middlebox. It requires routers and end hosts to implement the protocol.

Besides the drafted proposals submitted to IETF, the Session Traversal Utilities for NAT (STUN) [21] can also be used for prefix detection owing to its capacity to learn the external IP addresses, as covered in [9, 12]. It requires applications to be modified.

We also note that there are many research studies and proposals for middlebox detection [22–34] and we discuss them in Section 5.

3 Hands-on Experience and Standardization

In this section, we analyze the existing proposals and share our standardization experience. By referring to the well-known protocol design principle [35], we show how those principles and practice actually work together in reality.

Through the WiBrA project, we conducted an extensive comparison study of existing solutions [9, 12] and proposed two competing mechanisms [10, 13] to IETF for standardization. We implemented both solutions including GNU C library implementation and EDNS0 patches for DNS BIND9. After several iterations based on the feedback from the IETF community, our heuristic discovery protocol becomes a standard. We highlight the key features of the proposals in Table 2.

Table 2: Analysis of Proposals

Proposal	Active detection	Adaptation to changes	Impact on network entities	Host system changes	Secure detection
Heuristic	Yes	Moderate	Yes if DNSSEC	No	With DNSSEC
EDNS0 option	Yes	Fast	Yes	Yes	No
DNS A64	Yes	Fast	Yes	Yes	No
DNS TXT	Yes	Fast	Yes	No	No
DNS U-NAPTR	Yes	Fast	Yes	No	No
PCP	Yes	Fast	Yes if DNSSEC	Yes	With DNSSEC
DHCPv6	No	Moderate	Yes	Yes	No
RA	No	Fast	Yes	Yes	No

Regarding to the key factors identified by RFC 5218, all the candidate proposals meet a real need to drive the IPv6 transition. However, only a few meet the requirement of incremental deployability, i.e., with minimum impact to the existing network infrastructure. The heuristic approach, owing to its independence of the existing implementations in the network, stands out in this aspect during the competition. Many appealing features such as efficiency of detection and adaptation are of less impact in the standardization process in reality.

Meanwhile, extensibility also plays a crucial role. When the heuristic discovery proposal entered the final stage, we received strong requests from the IETF community to add authentication support. Owing to the flexible design, we are able to meet the request and hence pushing it to the final version. In retrospective, if our proposal were not flexible or extensible, its standardization path could have been protracted even longer.

Another indication is that the push from vendors and operators can affect the proposal standardization. The PCP based solution [17] is a good example. Being proposed in late 2012 as an individual draft, the space for such transitional mechanism is already crowded given all other candidates available. Because the Port Control Protocol (PCP) [18] offers good support to control and manage middleboxes, it is backed by major vendors and operators. After PCP becoming a standard in 2013, the PCP based approach benefit from it and made its way to a standard in May 2014.

4 Experimental Findings

To investigate the detection mechanisms and the behavior of middleboxes, we select four open source platforms to test our heuristic discovery protocol. Table 3 summarizes the platforms used in our tests.

Table 3: Open Source NAT64 Platforms

NAT64 Platform	Type	Last Update	OS	License
Ecdysis [36]	Stateful	04.2014	BSD, Linux	GNU GPLv3
Tayga [37]	Stateless	12.2010	Linux	GNU GPLv2
Jool [38]	Stateful	10.2014	Linux	GNU GPLv3
WrapSix [39]	Stateful	07.2013	Linux	GNU GPLv3

We conduct a set of experiments in our lab at the University of Helsinki to validate the detection mechanism against those four selected platforms. We installed the NAT64 implementations on our Linux desktop machines to act as middleboxes, running kernel 3.13 with 1 Gbps Ethernet connections. As expected, we are able to detect the IPv6 prefixes used by the middlebox, although the DNSSEC operation adds up latency due to the extra round trips and the computation overhead from authentication.

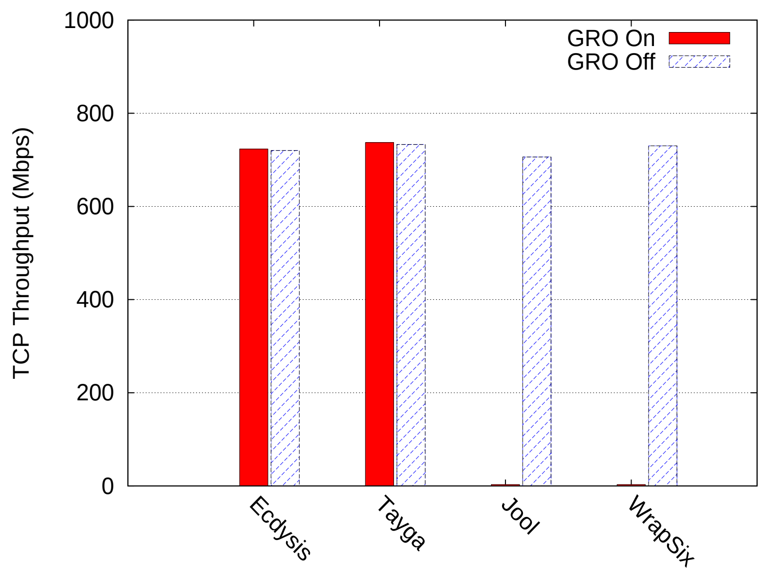

Fig. 2: Performance issue on NAT64 platforms

More interestingly, we spot a performance issue during the TCP throughput tests and plot the results in Figure 2. Among the chosen platforms, half of the NAT64 implementations, namely Jool and WrapSix, exhibit performance degradation. The Ecdysis and Tayga do not exhibit such problem. This experimental finding reveals an issue in the existing implementations that are hard to probe nor diagnose by the end host along. With our debugging efforts, the cause of the trouble turns out to be the optimization functionality generic receive offload (GRO) used by the Linux system. Both Jool and WrapSix implementations can not properly handle the GRO operation that merges the arriving packets to reduce the load of CPU. This failure leads to massive drop of packets on the arriving interface and thus bringing down the TCP throughput from over 700 Mbps down to around 220 Kbps, i.e., 3000+ times difference. To solve this problem, we manually turned off GRO on the receiving interface.

This issue spotted shows that the client-side detection mechanism is limited and can not detect or prevent all the problems generated by middleboxes. One quick alternative for the end host is to use other access or interface, if available, to circumvent the middlebox interference. We provide detailed discussion in the following section.

5 Discussions and Suggestions

In this section we discuss open issues based on our observations from the standardization experience and experimental findings. We further offer suggestions how to cope with middleboxes in a more general sense.

The problem spotted in our experiment leads to a research question: *how to detect and debug hidden issues of middleboxes and further solve them in a cost-effective manner?*

Because existing standardized solutions often target specific scenarios or depend on other protocols, they may not be suitable for general debugging or diagnosing unpredictable problems. To our knowledge, research communities have been investigating this domain for decades. The lessons and findings offer us good insights.

Given TCP's end-to-end design frequently disrupted by middleboxes, a good modeling of middleboxes can simplify our interaction and configuration with middleboxes by using abstraction [40]. Direct enhancement on TCP itself can also help end users to diagnose expected problems more efficiently [34].

At the same time, measurement studies can improve our understanding on the behavior of existing middleboxes, as done for the Internet and Web [2,3,6,24, 28,32,41], for mobile networks [30], for home networks and gateways [26,27,31], and for transport layer such as TCP options [29,42,43].

To complement standardized solutions, general-purpose tools and novel detection mechanisms are the key to spot and cope with unpredictable issues. The Netalyzr [25] and Tracebox [33] are good examples.

From the standards perspective, the Port Control Protocol (PCP) offers a clean and direct solution by allowing the end host to manage the behavior of NAT devices following the predefined specification. It also enables a more active communication channel between end hosts and middleboxes to get rid of various workaround mechanisms such as outgoing keep-alive messaging. There are also ongoing discussions among the operators to use IKEv2 [52]. For mobile networks, RFC 6888 [53] identifies a set of common requirements for Carrier-Grade NAT devices to improve sevice stability and robustness.

To solve the challenge of managing middleboxes, we also need to rethink on the architecture level. As hinted by the radical design of new IP version, i.e., IPv10 [44], we shall try integrating the network function virtualization (NFV) and software-defined networking (SDN) into our future design. As highlighted in recent work [45–49], the new design could be a possible direction to move forward.

Of course when IPv6 is fully deployed, many transition mechanisms will fade away. However, middleboxes will still find their way in the Internet. We note that our lessons learned from the IPv6 transition can provide hints for general usage: at which layer and what feature is crucial for a detection protocol to make to standard and become widely deployed.

To enlighten the future development, we offer suggestions for each stakeholder in this domain:

1. For protocol designers: A solution with incremental deployability should be a first concern. It implies an application layer or technology independent approach that does not bring too much overhead to the host stack and with minimum impact on other network entities are most likely favored by the standardization community.
2. For end users: Adopting the latest research solution can save time and effort by identifying and eliminating potential pitfalls such in the wireless environment [54]. Upon detecting the middlebox presence, for hosts that have multiple accesses or interfaces including WiFi and cellular deployed on mobile handsets, one quick solution is to switch to another access channel or interface to avoid the path through middlebox. When dealing with uncooperative vendors and operators who manage the error-prone middleboxes, users could take more innovative steps by collaborating with each other and publishing the failure results and findings to the Internet and mass media, hence forcing changes to occur.
3. For vendors and operators: collaborating with end users will lead to a win-win situation. By using their help to debug and upgrade the error-prone devices, it results in saving of the maintaining and operational cost in the long run. It also makes sense to set up a channel with users and researchers to understand their needs and latest findings. To reduce the cost of operation, introducing solid and novel research design such as SDN based proposals can potentially benefit every stakeholders in the domain.
4. For researchers: there is a need for new transport design and new architecture [43, 50, 51] in order to ease the management and upgrade of , therefore our future direction can borrow ideas from the active communities such as software-defined networking (SDN) and network function virtualization (NFV). When a mechanism is proven to be valuable, researchers shall also consider taking their proposal forward to SDO and standardize it [55].

6 Concluding Remarks

This position paper presents our research on the protocol design for middlebox detection through experimental study and standardization. We hope this work can deepen our understanding of the middlebox behaviour and shed light on how to design robust and deployable middlebox detection protocols for the evolving IPv6 Internet.

Acknowledgement

The research of Aaron Yi Ding is supported by the Nokia Foundation.

References

1. P. Wu, Y. Cui, J. Wu, J. Liu, C. Metz, Transition from IPv4 to IPv6: A State-of-the-Art Survey. *IEEE Communications Surveys and Tutorials*, 15(3):1407-1424, July 2013.

2. J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey, Measuring IPv6 Adoption. In *Proceedings of ACM SIGCOMM 2014.*
3. KC Claffy, Tracking IPv6 Evolution: Data We Have and Data We Need. *ACM SIGCOMM Computer Communication Review*, 41(3):43-48, July 2011.
4. M. Bagnulo, P. Matthews, and I. van Beijnum, Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. Internet RFCs, ISSN 2070-1721, RFC 6146, 2011.
5. M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum, DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. Internet RFCs, ISSN 2070-1721, RFC 6147, 2011.
6. J. Arkko and A. Keranen, Experiences from an IPv6-Only Network Internet RFCs, ISSN 2070-1721, RFC 6586, 2012.
7. M. Mawatari, M. Kawashima, and C. Byrne, 464XLAT: Combination of Stateful and Stateless Translation Internet RFCs, ISSN 2070-1721, RFC 6877, 2013.
8. Wireless Broadband Access (WiBrA) Project: `http://www.cs.helsinki.fi/group/wibra/`
9. Y. Ding, T. Savolainen, J. Korhonen, and M. Kojo, Speeding up IPv6 Transition: Discovering NAT64 and Learning Prefix for IPv6 Address Synthesis. In *Proceedings of IEEE ICC 2012.*
10. T. Savolainen, J. Korhonen, and D. Wing, Discovery of IPv6 Prefix Used for IPv6 Address Synthesis. Internet RFCs, ISSN 2070-1721, RFC 7050, 2013.
11. Comcast Reaches Key Milestone in Launch of IPv6 Broadband Network. `http://corporate.comcast.com/comcast-voices/comcast-reaches-key-milestone-in-launch-of-ipv6-broadband-network`
12. J. Korhonen, and T. Savolainen, Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix. Internet RFCs, ISSN 2070-1721, RFC 7051, 2013.
13. J. Korhonen, and T. Savolainen, EDNS0 Option for Indicating AAAA Record Synthesis and Format. IETF Internet Draft, draft-korhonen-edns0-synthesis-flag-02, 2011.
14. P. Vixie, Extension Mechanisms for DNS (EDNS0). Internet RFCs, ISSN 2070-1721, RFC 2671, 1999.
15. M. Boucadair, and E. Burgey, A64: DNS Resource Record for IPv4-Embedded IPv6 Address. IETF Internet Draft, draft-boucadair-behave-dns-a64-02, 2010.
16. D. Wing, Learning the IPv6 Prefix of a Networks IPv6/IPv4 Translator. IETF Internet Draft, draft-wing-behave-learn-prefix-04, 2009.
17. M. Boucadair, Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP) Internet RFCs, ISSN 2070-1721, RFC 7225, 2014.
18. D. Wing, S. Cheshire, M. Boucadair, R. Penno, and P. Selkirk, Port Control Protocol (PCP) Internet RFCs, ISSN 2070-1721, RFC 6887, 2013.
19. M. Boucadair, J-L. Grimault, T. Savolainen, and G. Bajko, Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions. IETF Internet Draft, draft-boucadair-dhcpv6-shared-address-option-01, 2009.
20. D. Wing, X. Wang, and X. Xu, Learning the IPv6 Prefix of a Networks IPv6/IPv4 Translator. IETF Internet Draft, draft-wing-behave-learn-prefix-03, 2009.
21. J. Rosenberg, R, Mahy, P. Matthews, D. Wing, Session Traversal Utilities for NAT (STUN). Internet RFCs, ISSN 2070-1721, RFC 5389, 2008.
22. A. Medina, M. Allman, S. Floyd, Measuring Interactions Between Transport Protocols and Middleboxes. In *Proceedings of ACM SIGCOMM IMC 2004.*
23. B. Ford, P. Srisuresh, D. Kegel, Peer-to-Peer Communication Across Network Address Translators. In *Proceedings of USENIX ATC 2005.*

24. A. Muller, G. Carle, A. Klenk, Behavior and Classification of NAT Devices and Implications for NAT Traversal. *IEEE Network*, 22(5):14-19, October 2008.
25. C. Kreibich, N. Weaver, B. Nechaev, V. Paxson, Netalyzr: Illuminating The Edge Network. In *Proceedings of ACM SIGCOMM IMC 2010.*
26. S. Hätönen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, M. Kojo, An Experimental Study of Home Gateway Characteristics. In *Proceedings of ACM SIGCOMM IMC 2010.*
27. A. Muller, G. Munz, G. Carle, Collecting Router Information for Error Diagnosis and Troubleshooting in Home Networks. In *Proceedings of IEEE LCN 2011.*
28. G. Halkes, J. Pouwelse, UDP NAT and Firewall Puncturing in the Wild. In *Proceedings of IFIP NETWORKING 2011.*
29. M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, H. Tokuda, Is it Still Possible to Extend TCP?. In *Proceedings of ACM SIGCOMM IMC 2011.*
30. Z. Wang, Z. Qian, Q. Xu, Z. Mao, M. Zhang, An Untold Story of Middleboxes in Cellular Networks. In *Proceedings of ACM SIGCOMM 2011.*
31. L. DiCioccio, R. Teixeira, M. May, C. Kreibich, Probe and Pray: Using UPnP for Home Network Measurements. In *Proceedings of PAM 2012.*
32. A. Muller, F. Wohlfart, G. Carle, Analysis and Topology-based Traversal of Cascaded Large Scale NATs. In *Proceedings of HotMiddlebox Workshop 2013.*
33. G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, B. Donnet, Revealing Middlebox Interference with Tracebox. In *Proceedings of ACM SIGCOMM IMC 2013.*
34. R. Craven, R. Beverly, M. Allman, A Middlebox-Cooperative TCP for a non End-to-End Internet. In *Proceedings of ACM SIGCOMM 2014.*
35. D. Thaler and B. Aboba, What Makes for a Successful Protocol? Internet RFCs, ISSN 2070-1721, RFC 5218, 2008.
36. Ecdysis: open-source implementation of a NAT64 gateway. `http://ecdysis.viagenie.ca/`
37. TAYGA - NAT64 for Linux. `http://www.litech.org/tayga/`
38. Jool Stateful NAT64. `https://www.jool.mx/`
39. WrapSix. `http://www.wrapsix.org/`
40. D. Joseph, I. Stoica, Modeling Middleboxes. *IEEE Network*, 22(5):20-25, October 2008.
41. M. Nikkhah, R. Guerin, Y. Lee, R. Woundy, Assessing IPv6 Through Web Access A Measurement Study and Its Findings. In *Proceedings of ACM CoNEXT 2011.*
42. M. Kühlewind, S. Neuner, B. Trammell, On the State of ECN and TCP Options on the Internet. In *Proceedings of PAM 2013.*
43. B. Trammell, and J. Hildebrand, Evolving Transport in the Internet. *IEEE Internet Computing*, 18(5):60-64, 2014.
44. K. Carlberg, S. Bhatti, J. Crowcroft, IP Version 10.0: A Strawman Design Beyond IPv6. In *Proceedings of ACM ReArch Workshop 2009.*
45. J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, V. Sekar, Making Middleboxes Someone Elses Problem: Network Processing as a Cloud Service. In *Proceedings of ACM SIGCOMM 2012.*
46. Z. Qazi, C. Tu, R. Miao, L. Chiang, V. Sekar, M. Yu, Practical and Incremental Convergence between SDN and Middleboxes. In *Proceedings of ONS 2013.*
47. Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *Proceedings of ACM SIGCOMM 2013.*
48. A. Gember, A. Krishnamurthy, S. S. John, R. Grandl, X. Gao, A. Anand, T. Benson, V. Sekar, A. Akella, Stratos: A Network-Aware Orchestration Layer for Virtual Middleboxes in Clouds. *arXiv:1305.0209v2, 2013.*

49. V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, G. Shi, Design and Implementation of a Consolidated Middlebox Architecture. In *Proceedings of USENIX NSDI 2012.*
50. L. Zhang, A Retrospective View of Network Address Translation. *IEEE Network*, 22(5):8-12, October 2008.
51. M. K. Mukerjee, D. Han, S. Seshan, P. Steenkiste, Understanding Tradeoffs in Incremental Deployment of New Network Architectures. In *Proceedings of ACM CoNEXT 2013.*
52. C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, T. Kivinen, Internet Key Exchange Protocol Version 2 (IKEv2). Internet RFCs, ISSN 2070-1721, RFC 7296, 2014.
53. S. Perreault, I. Yamagata, S. Miyakawa, A. Nakagawa, H. Ashida, Common Requirements for Carrier-Grade NATs (CGNs). Internet RFCs, ISSN 2070-1721, RFC 6888, 2013.
54. K. Kim, H. Nam, and H. Schulzrinne, WiSlow: A Wi-Fi Network Performance Troubleshooting Tool for End Users. In *Proceedings of IEEE INFOCOM 2014*.
55. A. Y. Ding, J. Korhonen, T. Savolainen, M. Kojo, J. Ott, S. Tarkoma, and J. Crowcroft, Bridging the Gap between Internet Standardization and Networking Research. *ACM SIGCOMM Computer Communication Review*, 44(1):56-62, January 2014.