A Functional and Performance-Oriented Comparison of Transition Mechanisms for Internet Transition from IPv4 to IPv6 Protocol

A dissertation presented

by

Nejc Škoberne

to

The Faculty of Computer and Information Science

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the subject of

Computer and Information Science

Ljubljana, 2013

#/viris[ ]

Investing in your future
Operation Part Financed by The European Union
European Social Fund

A Functional and Performance-Oriented Comparison of Transition Mechanisms for Internet Transition from IPv4 to IPv6 Protocol

A dissertation presented

by

Nejc Škoberne

to

The Faculty of Computer and Information Science

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

in the subject of

Computer and Information Science

Ljubljana, 2013

APPROVAL

I hereby declare that this submission is my own work and that to the best of my knowledge, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made.

— Nejc Škoberne —

December 2013

The submission has been approved by

dr. Mojca Ciglarič

Assistant Professor of Computer and Information Science

ADVISOR AND EXAMINER

dr. Andrej Kos

Associate Professor of Electrical Engineering

PRESIDENT OF THE EXAMINATION COMMITTEE

dr. Zoran Bosnić

Assistant Professor of Computer and Information Science

EXAMINER

Dr Olaf Maennel

Lecturer of Computer Science

EXTERNAL EXAMINER

Loughborough University

PREVIOUS PUBLICATIONS

I hereby declare that the research reported herein was previously published/submitted for publication in peer reviewed journals or publicly presented at the following occasions:

[1] N. Škoberne, O. Maennel, I. Phillips, R. Bush, J. Žorž and M. Ciglarič. IPv4 Address Sharing Mechanism Classification and Tradeoff Analysis. *IEEE/ACM Transactions on Networking*, volume PP, number 99, 2013. doi: 10.1109/TNET.2013.2256147

[2] N. Škoberne and M. Ciglarič. Practical Evaluation of Stateful NAT64/DNS64 Translation. *Advances in Electrical and Computer Engineering*, volume 11, number 3, pages 49-54, 2011. doi: 10.4316/aece.2011.03008

I certify that I have obtained written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work I have completed during my registration as a postgraduate student at the University of Ljubljana.

To my darling wife Mica, the true master of patience.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Nejc Škoberne

Zmogljivostna in funkcionalna primerjava prehodnih mehanizmov pri prehodu interneta s protokola IPv4 na IPv6

POVZETEK

V času, ko je bil internet osnovan, je ideja o računalniškem omrežju, ki bi povezovalo milijarde naprav po celotnem planetu, sodila bolj v znanstveno fantastiko kot kam drugam. Tudi to je eden izmed razlogov, zakaj je na internet v zadnjih 20 letih nastala precejšnja gneča. Glavni razlog je v tem, da je bil protokol IP eden ključnih protokolov za delovanje Interneta, razvit kot eksperimentalni protokol, ki je "pobegnil iz laboratorija". Manj kot desetletje zatem je internetna skupnost ugotovila, da bo naslovov glede na trend porabe kmalu zmanjkalo in začele so nastajati prve rešitve, ki bi težave naslovile. Prehod na brezrazredno usmerjanje med domenami (angl. Classless Inter-Domain Routing, CIDR) je bil zelo pomemben za nadaljnjo rast interneta, vendar kljub temu ni zadostoval. Dejstvo, da je možno nasloviti le 2^{32} internetnih naprav, je predstavljalo razlog za resno zaskrbljenost, ki je bila začasno odpravljena z agresivno uporabo mehanizmov za prevajanje omrežnih naslovov. Žal je ta tehnologija hkrati tudi resno ogrozila enega od ključnih načel interneta – povezljivost od točke do točke.

Kot edina možna dolgoročna rešitev za izčrpanost naslovnega prostora IPv4 *in* za povezljivost od točke do točke se kaže prehod interneta s "starega" protokola IPv4 na "novi" protokol IPv6. Prehod je potreben, ker sta protokola IPv4 in IPv6 nezdružljiva. Izvedba prehoda tako velikega in kompleksnega sistema kot je internet, ni enostavna naloga. Posebno ne v primeru, če izvajanje začasnih "grdih" rešitev vedno znova (začasno) odpravlja najbolj pereče težave in če prava motivacija za prehod izgleda zelo oddaljena in zamegljena. Za izvedbo prehoda je bilo od nastanka protokola IPv6 predlaganih mnogo različnih prehodnih mehanizmov, ki naj bi prehod poenostavili in ga naredili manj bolečega za uporabnike. Znanstveni pristop k razvoju, analizi in vrednotenju teh mehanizmov je ključen, če se želimo izogniti negativnim učinkom med prehodom kot tudi med obdobjem sobivanja protokolov IPv4 in IPv6.

V tej disertaciji smo se temeljito poglobili v problem prehoda Interneta na protokol

IPv6. Čeprav je bil protokol IPv6 standardiziran več kot desetletje in pol nazaj, smo pokazali, da se prehod ni še niti dobro začel. Ozrli smo se tudi nazaj in preučili, kako se je prehod na protokol IPv6 začel in izvajal skozi čas ter identificirali glavne zaviralne in pospeševalne dejavnike. Nadalje smo identificirali dve glavni skupini prehodnih mehanizmov: mehanizmi za umestitev IPv6, ki omogočajo uvedbo protokola IPv6 v omrežja, ki uporabljajo izključno protokol IPv4; ter mehanizme za deljenje naslovov IPv4, ki zagotavljajo dolgoročno sobivanje protokolov IPv4 in IPv6 na Internetu tako, da omogočajo ponudnikom internetnih storitev dodeljevanje enega naslova IPv4 več naročnikom hkrati. Analizirali smo tudi trenutno stanje prehoda in pripravili sistematičen pregled velikega števila prehodnih mehanizmov, ki so bili kdaj koli predlagani oz. objavljeni.

Mehanizme za deljenje naslovov IPv4 smo obravnavali še posebej temeljito. Na podlagi ugotovitve, da je zelo težko znanstveno vrednotiti dejanske predloge različnih mehanizmov, smo razvili klasifikacijo mehanizmov na podlagi petih dimenzij. Obstoječe mehanizme za deljenje naslovov IPv4 smo nato klasificirali v devet razredov ter analizirali lastnosti mehanizmov tako, da smo preučili možne vrednosti, ki jih posamezni mehanizmi zasedajo v vsaki dimenziji. Da bi omogočili praktikom, da bi razumeli kompromise med različnimi razredi mehanizmov, smo v okviru analize kompromisov ovrednotili ključne prednosti in slabosti posameznih razredov.

Z določenjem dimenzij klasifikacije smo hkrati skonstruirali 5-dimenzionalni prostor možnih prehodnih mehanizmov za deljenje naslovov IPv4. Tako smo lahko identificirali tudi kombinacijo lastnosti, ki so postale podlaga za nov uporaben mehanizem za deljenje naslovov IPv4, imenovan AP64. Zanj smo definirali arhitekturo in vse potrebne funkcije ter obrazložili vse tehnične podrobnosti za implementacijo.

Nazadnje smo predlagali ogrodje za teoretično analizo oz. vrednotenje zmogljivosti mehanizmov za deljenje naslovov IPv4. Spoznali smo, da vrednotenje zmogljivosti mehanizmov nasploh predstavlja velik izziv, saj so ti kompleksni in razmeroma zahtevni za implementacijo. Poleg tega je kvalitetne implementacije teh mehanizmov razomeroma težko pridobiti. Da bi se temu izognili, smo razbili posamezne razrede mehanizmov na osnovne operacije nad paketi in eksperimentalno ovrednotili njihovo zmogljivost. Ukvarjali smo se s prostorsko in časovno zahtevnostjo mehanizmov, kar omogoča, da končno ogrodje lahko uporabimo kot orodje za vrednotenje zmogljivosti prihodnjih mehanizmov tako, da jih najprej klasificiramo glede na našo klasifikacijo, nato pa identificiramo osnovne operacije, ki v njih nastopajo.

Ključne besede: Prehod na protokol IPv6, prehodni mehanizmi na IPv6, deljenje naslovov IPv4, NAT na ravni ponudnika (CGN), naslov plus vrata (A+P), vrednotenje zmogljivosti, prevajanje naslovne družine, prevajanje omrežnih naslovov (NAT), tuneliranje

University *of Ljubljana*
Faculty *of Computer and Information Science*

Nejc Škoberne
A Functional and Performance-Oriented Comparison of Transition Mechanisms for Internet Transition from IPv4 to IPv6 Protocol

ABSTRACT

A computer network that would cover the entire planet and connect billions of devices was beyond imagination when the Internet was first conceived. Consequently, over the last 20 years, the Internet has become a crowded place. This is because the IP protocol that makes the Internet possible was an experimental protocol that "escaped from the lab". The Internet address exhaustion problem was identified less than a decade later, when the first proposal solutions also began to emerge. Switching to the Classless Inter-Domain Routing (CIDR) scheme was crucial for continued Internet growth, but insufficient to resolve the resulting problematic. Being able to address only 2^{32} Internet nodes started becoming a serious concern that was subsequently temporarily dismissed due to the aggressive use of Network Address Translation (NAT). As collateral damage, NAT heavily endangered one of the core principles of the Internet–end-to-end connectivity.

The only possible long-term solution to the IPv4 address exhaustion *and* end-to-end connectivity problem appears to be transitioning the Internet from the "old" IPv4 protocol to the "new" IPv6 protocol. This must be done because IPv4 and IPv6 are incompatible. However, transitioning such a large and complex system is not an easy task, particularly if applying "hacks" continues to solve the most difficult problems and if motivation appears to be far off and misty. Many transition mechanisms have been proposed in order to solve this task and make the transition less painful for users. A scientific approach to the development, analysis and evaluation of these mechanisms is crucial in order to avoid as much hassle as possible during the transition as well as during the coexistence period of the IPv4 and IPv6 protocols.

In this thesis, we thoroughly address the IPv6 transition problem. Even though the IPv6 protocol was standardized more than a decade and a half ago, we show that the transition has not yet gained its momentum. We examine how the transition to the

IPv6 was approached and executed through time and what are the major drawbacks and drivers in this ongoing process. We identify two major groups of transition mechanisms: IPv6 deployment mechanisms, which are used to introduce IPv6 connectivity into IPv4-only networks, and IPv4 address sharing mechanisms, which are used to assure the long-term co-existence of the IPv4 and IPv6 protocols on the Internet by enabling ISPs to allocate a single IPv4 address to multiple subscribers. We analyse the current state of the transition and systematically review a large number of the IPv6 transition mechanisms that have been proposed thus far.

We then go one step further in approaching IPv4 address sharing mechanisms. In recognition that it is difficult to scientifically examine the actual proposals of various mechanisms, we develop a classification system by defining five dimensions. We classify existing IPv4 address sharing mechanisms into nine distinct classes and analyse mechanism properties by systematically analysing possible values along the dimensions. To enable practitioners to understand the tradeoffs between different mechanism classes, we asses the key advantages and disadvantages of individual classes in the tradeoff analysis.

By defining the dimensions for the classification, we construct a 5-dimensional space of all possible IPv4 address sharing mechanisms. We identify a combination of properties that result in another useful IPv4 address sharing mechanism–AP64. We define the architecture and function of this mechanism and provide all the technical details for its implementation.

Finally, we propose a theoretical performance analysis framework for IPv4 address sharing mechanisms. While conducting this research, it has become apparent that it is indeed challenging to generally evaluate the performance of different mechanisms because they are complex and difficult to implement. Also, the implementations of whole mechanisms are difficult to obtain. To alleviate this problem, we decompose the mechanism classes into basic packet operations and experimentally evaluate their performance. We tackle space and time requirements for the mechanisms and establish a resulting framework that may be used as a tool to evaluate the performance of future mechanisms by classifying them using the proposed classification system and identifying their basic packet operations.

Key words: IPv6 transition, IPv6 transition mechanisms, IPv4 address sharing, carrier grade NAT (CGN), address plus port (A+P), performance evaluation, address family

translation, network address translation (NAT), tunneling

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerest thanks to my mentor, Assistant Professor Mojca Ciglarič, PhD, for the exchange of ideas, for her ongoing guidance through tough problems, for sharing her knowledge, patience, and most of all, for directing me towards having my work published.

I would like to extend my gratitude to Olaf M. Maennel, PhD, for being my unofficial co-mentor for the last two years of my research work, and to Iain W. Phillips, PhD, for his committed work on joint projects and papers on IPv4 address sharing. I am also very grateful to them and their wives, Kaie and Thomasina, for the unselfish help and kindness they extended to me and my family during both our stays in Loughborough, UK. Thanks also to Randy Bush, Jan Žorž, Jozef Woods, Roshan Mosaheb, Simon Perreault, Ibrahim Rashid and the team at Sky, Ahmed Abubakar, Dan Wing, Gang Chen and Ole Trøan for their collaboration, testing, reviews and more.

I could not even have engaged in my PhD study without the support of my employer, Milan Gabor, who was very tolerant about my needs and wishes throughout this study. Thanks also go to Polona Vodopivec and Tanja Plavec for their dedicated office work and sincere support.

I am also extremely grateful to Miha Štajdohar, my friend and cofounder of Genialis, our startup. He has really helped me with PhD tips, has demonstrated a lot of patience throughout this process and performed a lot of work for the startup when I was otherwise engaged.

My deepest gratitude goes to my wife Mica and our three sons Jakob, Bine and Peter, as well as to my mom Dijana, my dad Renato and my brother Jure, who have all been my loyal supporters in their own way. I would also like to thank to Janja Rejec and Franci Kovačič, Mica's parents, for doing a lot of babysitting so that I could write this thesis. I am especially grateful to my little boys for their stress-relieving interruptions in the form of play requests,

bed-fighting, screaming, laughing, crying, diaper changing, etc.

— Nejc Škoberne, Ljubljana, December 2013.

CONTENTS

LIST OF ABBREVIATIONS

- AAAA: *DNS record type specific to the Internet class that stores a single IPv6 address*
- AICCU: Automatic IPv6 Connectivity Client Utility *(tunnel broker software client with AYIYA and TIC support)*
- AIIH: Assignment of IPv4 Global Addresses to IPv6 Hosts *(translation-based transition scheme proposed as an IETF Internet Draft document)*
- AIX: *series of proprietary Unix operating systems developed by IBM*
- ALG: Application Layer Gateway *(translation function that manipulates application-layer protocol headers)*
- AMS-IX: Amsterdam Internet Exchange *(internet exchange point located in Amsterdam, Netherlands)*
- AP64: *scalable address-plus-port IPv4 address sharing mechanism for IPv6 networks*
- A+P: Address Plus Port *(technique for sharing single IPv4 address among several users without using NAT in the ISP network)*
- APNIC: Asia-Pacific Network Information Centre *(regional internet registry for Asia-Pacific region)*
- ARPANET: Advanced Research Projects Agency Network *(one of the world's first operational packet switching networks)*
- AS: Autonomous System *(collection of connected Internet Protocol routing prefixes under the control of one or more network operators)*

- ATM: Asynchronous Transfer Mode *(telecommunications concept defined by ANSI and ITU standards for carriage of a complete range of user traffic, including voice, data and video signals)*
- AYIYA: Anything In Anything *(tunneling protocol for connecting separated areas of IP traffic with each other)*
- BDMS: Bi-Directional Mapping System *(mechanism for IPv4/IPv6 address mapping transition)*
- BGP: Border Gateway Protocol *(standardized exterior gateway protocol designed to exchange routing and reachability information between autonomous systems on the Internet)*
- BIA: Bump-in-the-API *(transition mechanism which allows for the hosts to communicate with other IPv6 hosts using existing IPv4 applications)*
- BIH: Bump-in-the-Host *(host-based IPv4 to IPv6 protocol translation mechanism)*
- BIS: Bump-in-the-Stack *(technique for non-IPv6 compliant applications on an IPv4-only host to communicate with IPv6 hosts)*
- BMR: Basic Mapping Rule *(in AP64 mechanism, a rule to derive the AP64 IPv6 address)*
- BSD: Berkeley Software Distribution *(family of Unix-like operating systems)*
- CATNIP: Common Architecture for the Internet *(convergence protocol that integrates CLNP, IP and IPX; one of the IPng proposals)*
- CGN: Carrier-Grade NAT *(transition mechanism where NAT function resides in the ISP core network)*
- CIDR: Classless Inter-Domain Routing *(method for allocating IP addresses and routing IP packets)*
- DE-CIX: German Internet Exchange *(internet exchange point located in Frankfurt, Germany)*

- CPE: Customer Premises Equipment *(any terminal network equipment located at the subscriber's premises connected to the ISP's network)*
- DF: Don't Fragment *(bit in IPv4 header that forbids the intermediate routers to fragment the packet)*
- DHCP: Dynamic Host Configuration Protocol *(protocol for automatic configuration of networking parameters to hosts)*
- DHCPv6: Dynamic Host Configuration Protocol for IPv6 *(protocol for automatic configuration of IPv6 networking parameters to hosts)*
- DNAT66: Destination Network Address Translation from IPv6 to IPv6 *(destination IP address translation function from IPv6 to IPv6)*
- DNS: Domain Name System *(hierarchical distributed naming system for devices connected to the Internet)*
- DNS4: IPv4 DNS server *(IPv4-only DNS component of BDSM mechanism architecture)*
- DNS46: IPv4-IPv6 DNS server *(IPv4-to-IPv6 DNS component of BDSM mechanism architecture)*
- DNS6: IPv6 DNS server *(IPv6-only DNS component of BDSM mechanism architecture)*
- DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers *(DNS server in Stateful NAT64 mechanism architecture)*
- DoS: Denial-of-Service *(attempt to make a machine or network resource unavailable to its intended users)*
- DS-Lite: Dual-Stack Lite *(transition mechanism where IPv4 packets are tunneled within IPv6 packets)*
- DSTM: Dual-Stack Transition Mechanism *(transition mechanism that allows to assign temporary global IPv4 addresses to IPv6 nodes)*

- DTI: Dynamic Tunnel Interface *(transition mechanism that leverages IPv4-in-IPv6 tunneling by using DNS service to obtain tunnel end-point addresses)*
- DTTS: Transparent Scalable Solution for IPv4 to IPv6 Transition *(transition mechanism that employs the dual stack approach with dynamic tunneling technique and is similar to DSTM)*
- EA: Embedded Address *(IPv4 address (or part thereof) embedded into IPv6 address)*
- FAITH: *implementation of TRT transition mechanism in the WIDE KAME IPv6 stack*
- FMR: Forward Mapping Rule *(in AP64 mechanism, optional additional mapping rules used for mapping functions operation)*
- FTP: File Transfer Protocol *(traditional application-layer protocol for file transfers)*
- GATEVI: *transition mechanism in IPv4-to-IPv6 space operating in a stateless manner*
- 3GPP: 3rd Generation Partnership Project *(initiative to make a globally applicable third-generation mobile phone system specification)*
- GRE: Generic Routing Encapsulation *(tunneling protocol developed by Cisco Systems that can encapsulate a wide variety of network layer protocols)*
- HTTP: Hypertext Transfer Protocol *(application protocol for distributed, collaborative, hypermedia information systems)*
- IANA: Internet Assigned Numbers Authority *(department of ICANN, a nonprofit private American corporation, which oversees global Internet number allocations)*
- ICMP: Internet Control Message Protocol *(protocol for exchanging error message between IP network nodes)*
- ICMPv6: Internet Control Message Protocol version 6 *(implementation of ICMP for IPv6 protocol)*
- IDS: Intrusion Detection System *(device or software application that monitors network or system activities for malicious activities or policy violations)*

- IETF: Internet Engineering Task Force *(nonprofit organization for development and promotion of Internet standards)*
- IID: Interface ID *(bits in AP64 IPv6 address that identify a specific network interface on a segment)*
- IOS: Internetwork Operating System *(proprietary operating system used on most Cisco Systems routers and current Cisco network switches)*
- IP: Internet Protocol *(principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries)*
- IPng: IP Next Generation *(pre-IPv6 name for the new Internet Protocol)*
- IPS: Intrusion Prevention System *(network security appliance that monitors network and/or system activities for malicious activity)*
- IPsec: Internet Protocol Security *(protocol suite for securing IP communications by authenticating and encrypting each IP packet of a communication session)*
- IPv4: Internet Protocol version 4 *(traditional Internet Protocol published in early 1980s)*
- IPv6: Internet Protocol version 6 *(new Internet Protocol published in 1996)*
- ISATAP: Intra-Site Automatic Tunnel Addressing Protocol *(transition mechanism meant to transmit IPv6 packets between dual-stack nodes on top of an IPv4 network)*
- ISP: Internet Service Provider *(organization providing Internet access to its subscribers)*
- IVI: *stateless IPv4/IPv6 translation mechanism that allows hosts in different address families to communicate with each other*
- IX: Internet Exchange point *(physical infrastructure through which ISPs exchange Internet traffic between their networks)*
- KAME: *joint effort of six organizations in Japan which aimed to provide a free IPv6 and IPsec protocol stack implementations*

- LAN: Local Area Network *(computer network that interconnects computers in a limited local area)*
- LDAP: Lightweight Directory Access Protocol *(application protocol for accessing and maintaining distributed directory information services over an IP network)*
- MAP: Mapping of Address and Port *(Cisco IPv6 transition proposal which combines A+P port address translation with the tunneling of legacy IPv4 protocol packets over an ISP provider's internal IPv6 network)*
- MIME: Multipurpose Internet Mail Extensions *(Internet standard that extends the format of email to support new features)*
- MPLS: Multiprotocol Label Switching *(mechanism in high-performance telecommunications networks that directs data from one network node to the next based on short path labels)*
- MSS: Maximum Segment Size *(the largest size of data (in bytes) that a computer or communications device can handle in a single, unfragmented piece)*
- MTU: Maximum Transmission Unit *(the largest size of a protocol data unit (in bytes) that can be sent in a packet network such as the Internet)*
- NAPT: Network Address and Port Translation *(IP address translation method that uses transport-layer identifiers to multiplex privately addressed hosts to a public IPv4 address)*
- NAPT44: NAPT from IPv4 to IPv4 *(IP address and port translation method for IPv4)*
- NAPT64: NAPT from IPv6 to IPv4 *(IP address and header translation method from IPv6 to IPv4)*
- NAPT66: NAPT from IPv6 to IPv6 *(IP address and port translation method for IPv6)*
- NAT: Network Address Translation *(any IP address translation method)*

- NAT-PT: Network Address Translation - Protocol Translation *(transition mechanism that enables for IPv4 to IPv6 and IPv6 to IPv4 address and header translation)*
- NAPT-PT: Network Address and Port Translation - Protocol Translation *(transition mechanism that enables for IP address and header translation from IPv6 to IPv4 and IPv4 to IPv6 using port translation)*
- NAT444: Network Address Translation from IPv4 to IPv4 to IPv4 *(transition mechanism, using double IP address and port translation method for IPv4)*
- NAT46: Network Address Translation from IPv4 to IPv6 *(any IP address and header translation method from IPv4 to IPv6)*
- NAT64: Network Address Translation from IPv6 to IPv4 *(any IP address and header translation method from IPv6 to IPv4)*
- NBMA: Non-Broadcast Multiple Access Network *(computer network to which multiple hosts are attached, but data is transmitted only directly from one computer to another single host)*
- NetBSD: *open-source Unix-like operating system descended from BSD with focus on portability*
- OMNeT++: *extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators*
- OpenBSD: *open-source Unix-like operating system descended from BSD with focus on security*
- OpenVMS: Open Virtual Memory System *(computer server operating system that runs on VAX, Alpha and Itanium-based families of computers)*
- OpenVPN: *open source software application that implements virtual private network techniques*
- OPNET Modeler: *suite of protocols and technologies for modelling various network types and technologies*

- OSI: Open Systems Interconnection *(effort to standardize computer networking that was started in 1977 by the International Organization for Standardization)*
- PacketCable: *industry consortium founded by CabelLabs with the goal of defining standards for the cable television modem access industry*
- PCP: Port Control Protocol *(protocol that allows an IPv6 or IPv4 host to control how incoming IPv6 or IPv4 packets are translated and forwarded by a NAT)*
- DHCPv6-PD: DHCPv6 – Prefix Delegation options *(additional DHCP options that provide a mechanism for automated delegation of IPv6 prefixes using DHCPv6)*
- PET: Prefixing, Encapsulation and Tunneling *(framework for IPv4-IPv6 interconnection that integrates tunneling, translation and prefix-related control plan operations)*
- NAT-PMP: NAT Port Mapping Protocol *(port forwarding configuration protocol for LAN networks)*
- PMTUD: Path MTU Discovery *(standardized technique in computer networking for determining the MTU size on the network path between two IP hosts)*
- PoP: Point of Presence *(artificial demarcation point or interface point between communicating entities)*
- PPP: Point-to-Point Protocol *(data link protocol commonly used in establishing a direct connection between two networking nodes)*
- PPPoE: Point-to-Point Protocol over Ethernet *(network protocol for encapsulating PPP frames inside Ethernet frames)*
- PR-NAPT66: Port-Restricted Network Address and Port Translation from IPv6 to IPv6 *(in AP64 mechanism, a translation function in the CPE)*
- PRL: Potential Router List *(set of entries about potential routers, used to support router and prefix discovery in ISATAP)*
- PSID: Port-Set ID *(in AP64 mechanism, a set of bits, uniquely identifying a port-set allocated to an AP64 CPE)*

- pTRTd: Portable Transport Relay Translator Daemon *(an implementation of TRT protocol)*
- RFC: Request for Comments *(an IETF memorandum on Internet standards and protocols)*
- RIPE-NCC: Réseaux IP Européens Network Coordination Centre *(RIR for Europe, the Middle East and parts of Central Asia)*
- RIR: Regional Internet Registry *(organization that manages the allocation and registration of Internet number resources within a particular region of the world)*
- ROI: Return On Investment *(concept of an investment of some resource yielding a benefit to the investor)*
- RouterOS: *operating system based on Linux kernel developed by MikroTik*
- RTT: Round-Trip Time *(the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received)*
- SCTP: Stream Control Transmission Protocol *(transport-layer protocol, serving in a similar role to the popular protocols TCP and UDP)*
- SIIT: Stateless IP/ICMP Translation *(stateless IP address and header translation method from IPv6 to IPv4 and IPv4 to IPv6)*
- SIP: Session Initiation Protocol *(signaling communications protocol, widely used for controlling multimedia communication sessions such as voice and video calls over IP networks)*
- SIPP: Simple Internet Protocol Plus *(one of the candidates that was considered in the IETF for IPng)*
- SixXS: Six Access *(free, non-profit, non-cost IPv6 tunnel broker service for Local Internet Registries and endusers)*
- SLAAC: Stateless Address Autoconfiguration *(process that IPv6 nodes use to automatically configure IPv6 addresses for interfaces)*

- SNAT66: Source Network Address Translation from IPv6 to IPv6 *(source IP address translation function from IPv6 to IPv6)*
- SOCKS: Socket Secure *(Internet protocol that routes network packets between a client and server through a proxy server)*
- SOCKS64: SOCKS from IPv6 to IPv4 *(transition mechanism providing an IPv4-IPv6 intercommunication gateway using SOCKS5 protocol)*
- STUN: Session Traversal Utilities for NAT *(standardized set of methods and a network protocol to allow and end host to discover its public IP address if it is located behind a NAT)*
- TC: Traffic Class *(8-bit field in IPv6 packet header that is used for packet classification and congestion control)*
- TCP: Transmission Control Protocol *(Internet protocol providing a reliable, ordered, error-checked delivery of a stream of octets between programs running on computers connected to an IP network)*
- TIME-WAIT: *(one of the states of TCP protocol that represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request)*
- ToS: Type of Service *(field in the IPv4 header, defined in different ways by five RFCs)*
- TR-069: Technical Report 069 *(Broadband Forum technical specification entitled CPE WAN Management Protocol)*
- TRT: Transport Relay Translation *(transport-layer transition mechanism relying on DNS translation between AAAA and A records)*
- TSP: Tunnel Setup Protocol *(networking control protocol used to negotiate IP tunnel setup parameters between a tunnel client host and a tunnel broker server)*
- TUBA: TCP and UDP with Bigger Addresses *(one of the candidates that was considered in the IETF for IPng)*

- TUN: *virtual-network kernel device*
- UDP: User Datagram Protocol *(Internet protocol that allows sending messages (datagrams) to other hosts on an IP network without prior communications to set up special transmission channels)*
- UPnP: Universal Plug and Play *(set of networking protocols that permit networked devices to seamlessly discover each other's presence on the network)*
- VoIP: Voice over Internet Protocol *(methodology and group of technologies for the delivery of voice communications and multimedia sessions over IP networks)*
- VPN: Virtual Private Network *(virtual network that enables a computer to send and received data across shared or public networks as if were directly connected to the private network)*
- WAN: Wide Area Network *(network that covers a broad area using private or public network transports)*
- WIDE: Widely Integrated Distributed Environment *(Internet project in Japan that runs a major backbone of the Japanese Internet and used to run the .jp top-level domain)*
- 464XLAT: *transition mechanism that provides limited IPv4 connectivity across an IPv6-only network by combining existing and well-known stateful protocol translation*

1

Introduction

When an Internet user reads his or her e-mail, surfs a news web site or talks to a friend over live video chat, he or she usually unknowingly uses a vast number of different communication technologies and protocols [1]. For a user, the Internet "just works". In fact, the end-user experience has improved significantly over the last three decades. The Internet has become faster, mobile, always-on, more reliable and cheaper. This has happened even though the number of hosts connected to the Internet has grown exponentially from a few hundred to about a billion [2] over the last 30 years.

In the seventies, Robert E. Kahn and Vint Cerf from ARPANET began working on an inter-network protocol [3] to overcome the differences that were inherent in the protocols of various types of networks. This would make the networks more interoperable and individually less responsible for delivering packets. The idea was to shift the responsibility for communications from the networks to hosts. This gave birth to the Internet Protocol [4], which enabled hosts to send a packet to other hosts using only one destination address. However, the Internet Protocol was never designed to scale to millions or to even billions of hosts. When the 32-bit fields were chosen for source and destination addresses, most of the engineers believed this would be more than enough for "an experiment". If the experiment proved to be successful, they would build "production" versions of protocols afterwards. However, the Internet Protocol soon got out of control and took on a life of its own.

In the late 80s, just a few years following deployment, it became clear that address conservation methods would need to be implemented. The number of connected hosts soon surpassed 100,000 and by 1992, following migration to the classless network model [5, 6], IETF formalized the process of finding a more scalable solution by creating a temporary, ad-hoc IP Next Generation (IPng) Directorate to deal with the issues of the next Internet Protocol. Consequently, the Directorate issued a white paper solicitation [7] for the Next Generation Internet Protocol. The following final candidates were analysed [8]: CATNIP [9], TUBA [10], SIPP [11]. The latter (SIPP) was then selected as a basis for the development of IPv6, which was published in a series of RFCs, commencing with RFC 1883 [12] in 1996.

IPv6 [13] is in many aspects very different from IPv4. Firstly, its address space is 128 bits in length, compared to 32 bits in IPv4, which eliminates addressing scalability problems for some time to come. Secondly, multicasting is part of the base specification in IPv6. Moreover, SLAAC (Stateless Address Auto-Configuration) enables IPv6 hosts to automatically configure themselves when connected to a routed IPv6

network. Finally, its headers are simplified for router processing and in-stack support for network-level security (IPsec) is encouraged. IPv6 offers other features that are non-existent in IPv4.

In fact, IPv6 is so different from IPv4 that IPv6-only and IPv4-only hosts cannot directly communicate with each other and are thus incompatible on-the-wire. This means that the Internet must be transitioned from IPv4 to IPv6. When the first IPv6 RFC was published, there were about 15 million hosts connected to the Internet [2]. Initially, the transition from the IPv4 to the IPv6 protocol was envisioned [14] so that all of existing (and all new) hosts would gradually become IPv6-ready while retaining thei rIPv4 connectivity and reachability. When all Internet nodes could finally speak IPv6, a phase-out of IPv4 could be initiated.

However, as of 2013, the transition of the Internet to IPv6 has not been completed. In fact, it could be argued that it has not even yet begun. For example, by March 2013, the percentage of Internet hosts which access Google services barely surpassed 1% [15]. On the other hand, at 24% globally [16] (March 2013), IPv6 penetration at content providers (top 500 web sites) is substantially stronger. It is nevertheless obvious that the transition to IPv6 has been very slow over the past 17 years, when the first implementations began to appear.

In the first years after the next generation Internet Protocol was standardized as IPv6, implementations were rare and far from production quality. Among the first operating systems with IPv6 support were IBM's AIX (1997), DEC's Tru64 and OpenVMS (1997). By 2000, all BSD operating systems had production-quality IPv6 support via the KAME project. In 2001, Cisco introduced IPv6 support for its Cisco IOS routers and Layer-3 switches. In 2002, Microsoft included IPv6 into its Windows Server 2003 as a core networking technology, suitable for commercial deployment. In 2003, Apple Mac OS X also joined the club. It was not until 2005 that Linux kernel version 2.6.12 removed experimental status from its IPv6 implementation. Finally, in 2007, more than 10 years after the first IPv6 RFC was published, Microsoft added production-quality IPv6 implementation into Windows Vista. Before all of these major operating system vendors added IPv6 support into their products, it was impossible to achieve high levels of IPv6 deployment in edge networks.

For a long time, IPv6 proponents have also been struggling to nail down a set of drivers for the IPv6 transition in order to find a "IPv6 killer app" [17]. Decision-makers were eager to see how their investments into migration would fare on returns but en-

gineers failed to come up with a business model for IPv6. Unfortunately, the most serious "driver" – IPv4 address exhaustion – was also the most evasive [18]. Mainly, this was because of address usage slowdown. In the early 1990s, this slowdown was caused by migration to the CIDR network model. But more importantly and to a greater extent, it has also been caused by the massive deployment [19] of Network Address and Port Translation (NAPT) [20] devices in enterprises and home networks over the last two decades. This phenomenon repeatedly pushed forecasted IPv4 address exhaustion dates into the future, so that many people stopped believing the transition would ever even occur.

And yet, it did. On February 3rd 2011, the Internet Assigned Numbers Authority (IANA) announced that the pool of public IPv4 Internet addresses had been depleted. Consequently Regional Internet Registries (RIRs) were left with only the addresses they had been assigned prior to this date. On April 15th 2011, the Asia-Pacific Network Information Center (APNIC) activated its "last /8 address policy" [21]. Similarly, on September 14th 2012 Réseaux IP Européens Network Coordination Centre (RIPE NCC) also activated its last /8 policy. This means that from then on, any organization applying to these RIRs for IPv4 address space will receive a maximum allocation of one and only one /22 prefix (1024 IPv4 addresses). Such allocations are too small to satisfy current growth rates.

The motivation for this work is as follows. Since the birth of the IPv6 protocol, a vast number of IPv6 transition mechanisms have been proposed. Firstly, there is no complete and systematic overview of these technologies, which is the first step towards identifying the various design spaces of these architectures. Secondly, a taxonomy of all the relevant existing mechanisms and relevant new proposals would be beneficial for the further study and development of these technologies. Thirdly, there is no well-defined analysis and therefore no comparison of these mechanisms, which would enable practitioners to establish more informed decisions about which mechanism to deploy in certain environments. Also, by identifying the design space of IPv6 transition mechanisms, researchers and engineers would be able to investigate new potential mechanisms and technologies, which would display better characteristics over existing mechanisms and would benefit the IPv4-coexistence phase of the IPv6 transition.

1.1 Thesis Overview

We begin with an overview of IPv6 transition mechanisms in Chapter 2. We then provide an outline of the historical landscape of the IPv6 transition, which provides a solid base for understanding the motivation for developing new mechanisms. A high-level overview of traditional, IPv6 deployment mechanisms (tunneling, translation, dual-stack) is outlined. Following this, we provide a detailed review of IPv4 address sharing mechanisms along with a proposed classification system.

In Chapter 3 we shift focus to IPv4 address sharing mechanisms, which are the main topic of this thesis. We first describe an important building block of these mechanisms, the Network Address and Port Translation (NAPT) function. Next, we provide a classification of IPv4 address sharing mechanisms by proposing and explaining five dimensions, each with multiple properties. This classification is then used in the remaining sections of this thesis. As opposed to treating specific mechanisms, we discuss and analyse their abstractions – the classes of the classification. Further, we classify the mechanisms into nine classes and review them. The purpose of this exercise is to find general similarities that facilitate greater understanding and discussion of such technologies. We also examine the mechanisms' properties based on the proposed classification. We further identify and describe the most important practical technical issues that are related to the specific properties of each approach. Finally, we present a qualitative analysis that describes the tradeoffs between the classification dimensions. The analysis aims at better guiding Internet Service Providers (ISPs) in their decisions and provide a firm mgrounding for future research in this area. The material in this chapter is based on the paper (and inclusive research) that was recently published in IEEE/ACM Transactions on Networking [22].

We propose a new scalable IPv4 address sharing mechanism for IPv6-only networks in Chapter 4. First, we outline the reasons for defining a mechanism with a new set of properties. Second, we classify the new mechanism using the classification system outlined in Chapter 3, while we also compare the new class with existing classes of the classification. Third, we define mechanism functions that perform packet manipulations and provide step-by-step descriptions of the operation of every function.

In Chapter 5 a theoretical performance analysis and comparison framework for IPv4 address sharing mechanisms is proposed that enables the comparison of all classes outlined in the classification system. We begin by describing the methodology and metrics

used within the framework. We then we provide a functional decomposition of mechanism classes, which dissects each of the classes into a series of packet manipulation functions. These are in turn evaluated independently in order to present a grounding for performance analysis. Finally, using the framework, we analyse and compare the performance for each of the classes of taxonomy.

1.2 Contributions to Science

This dissertation includes the following original contributions to science:

- A systematic review of the IPv6 transition mechanism area, consisting of IPv6 deployment and IPv4 address sharing mechanisms.
- An IPv4 address sharing mechanism classification and theoretical performance analysis and comparison framework for IPv4 address sharing mechanisms and performance comparison of all classes included in the proposed classification.
- An IPv4 address sharing mechanism class property analysis and tradeoff analysis.
- AP64: a scalable address-plus-port IPv4 address sharing mechanism for IPv6-only networks.

2

Overview of IPv6 Transition Mechanisms

Before it was even decided which of the potential candidates [8] would eventually become the next-generation Internet Protocol (IPng) specification, the first IPng transition considerations were documented [23] by Carpenter in 1994. He made several interesting points, which we may effectively evaluate nearly twenty years later:

- *The transition to IPng will take years.* In fact, after more than two decades since the inception of the next-generation Internet Protocol, the great majority of Internet hosts are still equipped with only IPv4 connectivity [15].
- *Every site will need to decide its own staged transition plan.* In the recent past, several influential individuals have made unsuccessful attempts that suggested a more orchestrated approach to IPv6 transitioning [24].
- *Once the IPng decision is taken, the next decade (or more) of activity on the Internet and by all private networks using the Internet suite will be strongly affected by the process of IPng deployment.* By November 2013, there are still several working groups within IETF that engage in IPv6-related activities: 6lo, 6lowpan, 6man, 6tisch and v6ops [25].
- *It may not be a foregone conclusion that what they (i.e., user sites) change to will be IPng. Their main concern will be to minimise the cost of the change and the risk of lost production.* We may consider today's Internet, which is full of various NAPTs and tomorrow's Internet, which will be full of CGNs, as something very different from the "original" Internet, where end-to-end connectivity without middle boxes was granted. Not eventually transitioning to IPv6 means adopting this new kind of Internet, as IPv4 address depletion has already become a reality in 2011 [26].
- *It is expected that users will continue running their old equipment and software. Therefore, any combination of IPv4 and IPng hosts and routers must be able to interwork (i.e., participate in UDP and TCP sessions). An IPv4 packet must be able to find its way from any one IPv4 host to any other IPv4 or IPng host, or vice versa, through a mixture of IPv4 and IPng routers. In addition, an application package which is "aware" of IPv4 but still "unaware" of IPng must be able to run on a computer system that is running IPv4 but communicating with an IPng host.* This

compatibility requirement for IPv4 and IPng was not satisfied by IPv6 specification. This essentially means that IPv4 and IPv6 are incompatible on the wire [27].

- *Any transition scenario that requires dynamic header translation between IPv4 and IPng packets will create nearly insurmountable practical difficulties. Unless the translation mechanism is completely blind and automatic, the administration of translators will be quite impracticable for large sites.* This proved not to be true. Although NAT-PT [28] was indeed a failure [29], IETF later standardized Stateful NAT64 [30], which has already been widely implemented [31, 32]. The reason is most likely the advancement of hardware microprocessing technologies.

- *All hosts running IPng must still be able to run IPv4. IPng-only hosts are not allowed during transition.* This is an interesting requirement, which has already been broken. As of today, there already exist IPv6-only hosts (mostly clients), which are able to initiate connections to most of the IPv4 world (only using DNS [33], indirectly via IPv4 address literals) [34].

We can see that the transition of the Internet to the successor of IPv4 was considered very early by prominent engineers who were active within IETF. However, today we see that this has not yet been implemented. Indicators about this are discussed in the next section. Some people wonder as to what exactly is the reason that IPv6 has not to gained its momentum until just recently? Bush [35] questions: "*Was it the vendors? Lazy operators? Lack of content? Applications support? The CPE? End-user host stack? Not enough transition mechanisms?*" He concludes that the main problem is that IPv6 transition depends on the *simultaneous existence of these factors, which is a recipe for failure*.

Carpenter was not the only one to envision the long-term coexistence of IPv4 and IPv6. Bradner notes that *because the Internet is too large for all of its users to quickly cutover to IPng, IPng must coexist well with IPv4. Furthermore, IPv4 users will not spend the time, effort and money required to upgrade to IPng without a compelling reason. Access to new services will not be a strong enough motivation, since new services will want to support both IPng and the IPv4 users.* He argues that there could obviously be no such thing as "IPv6 killer app", which many have been looking for. He therefore concludes that *there will be a long period of coexistence between IPng and IPv4, so this coexistence*

needs to be quite painless, and not be based on any assumption that IPv4 use will rapidly diminish, if at all [36].

Robert Hinden [11], who co-authored IPv6 specification RFCs with Steve Deering, strongly emphasized the importance of backwards compatibility of IPng with IPv4 more than a year before IPv6 RFCs were published: *The Internet has a large installed base. Features need to be designed into an IPng to make the transition as easy as possible. As with processors and operating systems, IPng must be backwards compatible with IPv4*. Hinden also stated that *IPng must have a great transition strategy and new features*. For some reason, the backwards compatibility and transition strategy parts were omitted in the IPv6 specification. The same critique has been repeated many times by many different engineers. For example, Bernstein argues [37] that *the IPv6 designers made a fundamental conceptual mistake: they designed the IPv6 address space as an alternative to the IPv4 address space rather than as an extension to the IPv4 address space. In other words: The current IPv6 specifications do not allow public IPv6 addresses to send packets to public IPv4 addresses*. He also accused the IETF of not having an effective transition plan for IPv6. Finally, the authors themselves admitted that *the lack of real backwards compatibility for IPv4 was the single critical failure* [27] of IPv6.

However, the Internet community must play the cards it is dealt. To make IPv6 predominant over IPv4, the Internet will require a transition to IPv6. Žorž [38] provides an interesting insight into the motivation of those individuals, who deploy IPv6 today: enthusiasm, friendliness to new technologies, encouragement by progressive government regulation and procurement requirements (like RIPE-501 document), fear (of not being ready if and when the IPv6 transition actually takes place), new opportunities (possible advantages) and simplicity (greenfield deployments, where IPv4 is not really needed). Some authors envisioned an "S"-shaped three-phase transition plan with the following phases:

- *Preparation Phase*: This phase includes activities such as testing equipment, applications, IPv6 network services etc., and performance of pilot deployments in controlled environments.

- *Transition Phase*: In this phase, ISPs should offer production IPv4 and IPv6 services to their Internet customers while end-site organizations should provide Internet-facing services in a production manner. The goal is to enable IPv6 on most Internet hosts.

- *Post-Transition Phase*: Most Internet hosts are now IPv6-ready, so IPv4 sunsetting can begin. Eventually, most of the Internet will become IPv6-only.

One such transition plan is roughly described by Curran [24]. In this plan, each phase is also time-constrained with specific months and years. It must be noted that this RFC is not a standard of any kind and is informational only in nature. This means that noone is really obligated to follow it. Consequently, this specific plan has been completely ignored by the Internet community, which emphasizes Carpenter's point when he argues that *every site will need to decide its own staged transition plan*. However, it seems that there is consensus regarding the "S"-shape of the Internet transition [39].

Over the last twenty years, we managed to break one of the most fundamental concepts of the Internet: end-to-end connectivity. This is the ability of any host to send packets to any other host on the Internet. With the rise of Network Address Translation in edge networks, Internet connections are mostly client-initiated. Putting aside firewall issues, today's packets cannot get routed to most Internet-connected hosts. This negatively affects many Internet services: peer-to-peer networking, FTP, some VPN protocols as well as many others. This is why some Internet users experience slow file transfers when using Skype or are unable to play peer-to-peer games.

This semantic change of the Internet was also caused by delaying the IPv6 transition. Furthermore, in the last couple of years, we are experiencing a shift of ISPs locating translation devices from customers to ISP core networks. This is in contrast to the "stupid core, smart edge" philosophy which differentiates the packet-switched Internet from circuit-switched classic telecommunications networks. This was an issue also emphasized by Žorž [38]: *the risk in this transition phase is that the Internet will head off in a completely different direction*, from the *transition phase* in which we are currently. However, there is still hope that transitioning the Internet to IPv6 will to some extent restore, or at least improve, the Internet end-to-end connectivity principle. To achieve that, ISPs must be careful when selecting IPv6 transition mechanisms, or more accurately, IPv4 address sharing mechanisms. These must encourage the deployment of IPv6 as much as possible as opposed to blocking it.

To be able to perform the transition to IPv6, the tools and recommendations for their usage in specific scenarios are required. We call these tools IPv6 transition mechanisms–architectures, methods and technologies that facilitate the transition of the Internet from IPv4 to IPv6. IPv6 transition mechanisms enable hosts on either net-

work to participate in networking with the opposing network. By their nature, mechanisms are designed as a means to an end – to eventually transform the Internet to IPv6-only.

Internet Engineering Task Force (IETF) is an organization that develops and promotes Internet standards, in particular standards of the Internet protocol suite (TCP/IP). However, its scope also covers some higher layer protocols (LDAP, MIME, etc.) and standardization efforts in network management and operations. Thus, most transition mechanisms are designed and standardized within IETF. However, some are first proposed elsewhere, e.g., as scientific publications, and are later rewritten in the form of Internet Drafts and possibly published as Requests For Comments (RFCs).

Two overlapping phases of IPv6 transition can be identified. First, the goal was to connect IPv6-enabled nodes and networks over predominant IPv4-only infrastructure, which involved numerous tunneling mechanisms. The goal was to bring IPv6 connectivity into edge networks, which could not obtain native IPv6 connectivity. This was mostly due to the fact, that a great majority of ISPs would not upgrade their infrastructure to provide such connectivity. The mechanisms of the first phase are called *IPv6 Deployment Mechanisms*. The second phase is no longer about connecting IPv6 islands but rather, about ensuring long-term IPv4 and IPv6 coexistence. Until a great majority of Internet hosts is IPv6-enabled, it will be impossible to "turn off" IPv4. This clashes with the IPv4 address exhaustion problem, which leads the second transition phase to focus on providing various *IPv4 Address Sharing Mechanisms*. Some of the mechanisms fall into both groups. In this thesis, we focus on these technologies. We do so predominantly because IETF has been actively working on these mechanisms over the last years and most of the existing research and engineering work is directed into this group of mechanisms.

2.1 IPv6 Deployment Over Time

In this section, we present a historical view on IPv6 transition and discuss the current transition status. In order to further discuss transition mechanisms, we need to understand what has been happening with the Internet over the past two decades. First we examine various metrics that provide insight into different aspects of the IPv6 transition. Next, we review World IPv6 Day and World IPv6 Launch events which encouraged many ISPs and content providers to turn to IPv6. Finally, we discuss some of the most relevant IPv6 deployment surveys.

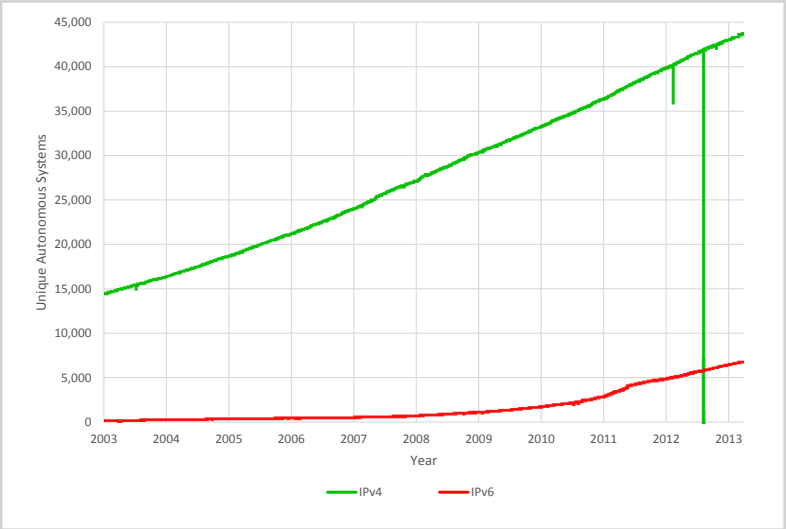


Figure 2.1

10-year graph of unique IPv6 and IPv4 autonomous systems.

2.1.1 IPv6 Deployment metrics

There are multiple possible metrics for measuring IPv6 deployment. In this section, we cover only the most interesting examples – those that enable us to get a sense of how the transition to IPv6 has been historically progressing and what the current status of the transition is. We consider IPv6 autonomous systems and announced prefixes, IPv4 vs. IPv6 traffic ratios, IPv6-ready web users and IPv6-ready web servers.

IPv6 Autonomous Systems and Announced Prefixes

The number of IPv6-ready autonomous systems that participate in BGP routing reveals us how many ISPs and organizations with provider-independent IP addresses have already enabled IPv6 on their border routers and are announcing IPv6 prefixes. The gradual growth of both IPv6 and IPv4 AS counts is depicted in Figure 2.1 [40]. From this graph, it becomes apparent that the number of autonomous systems which are IPv6-enabled is only 15.76% of IPv4 autonomous systems (April 2013).

The number of announced IPv6 prefixes (routes) in the global BGP routing table should be lower than the number of IPv4 prefixes, as IPv6 prefixes are much more spacious than are IPv4 prefixes. Figure 2.2 [40] indicates that the number of IPv6

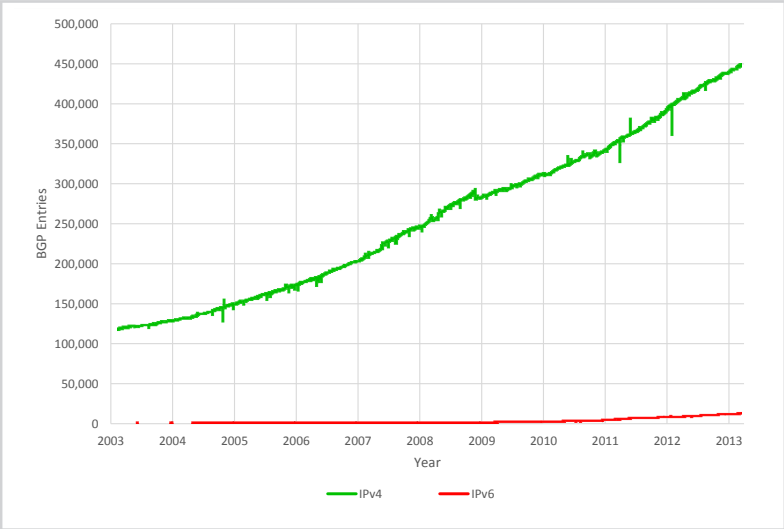


Figure 2.2

10-year graph of announced IPv6 and IPv4 prefixes.

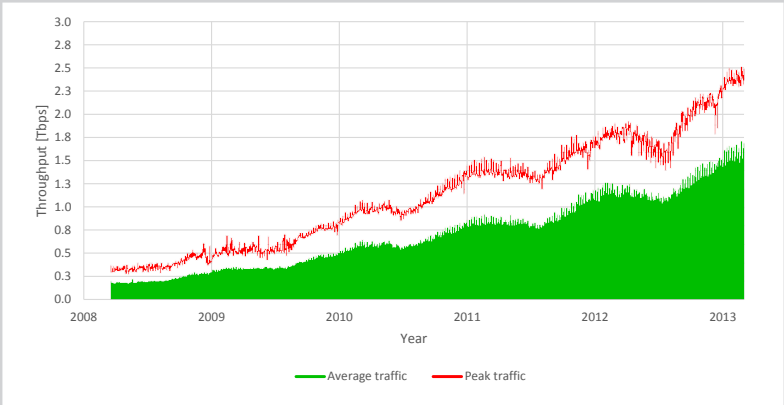


Figure 2.3

5-year graph of IPv4 traffic exchanged at DE-CIX.

prefixes (12,471) is approximately 2.76% of IPv4 prefixes (451,148) (April 2013).

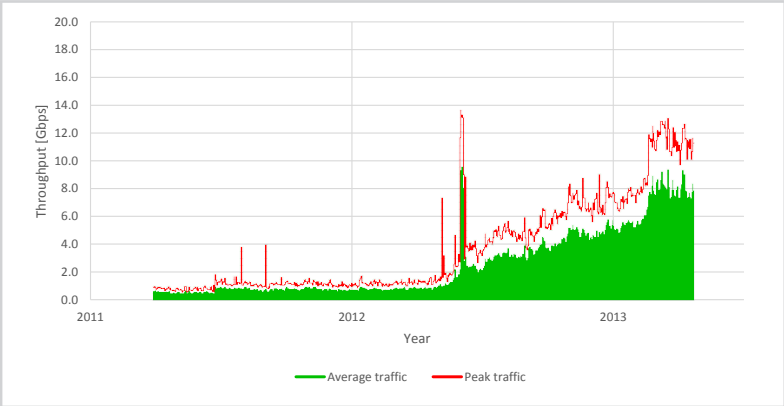


Figure 2.4

2-year graph of IPv6 traffic exchanged at DE-CIX.

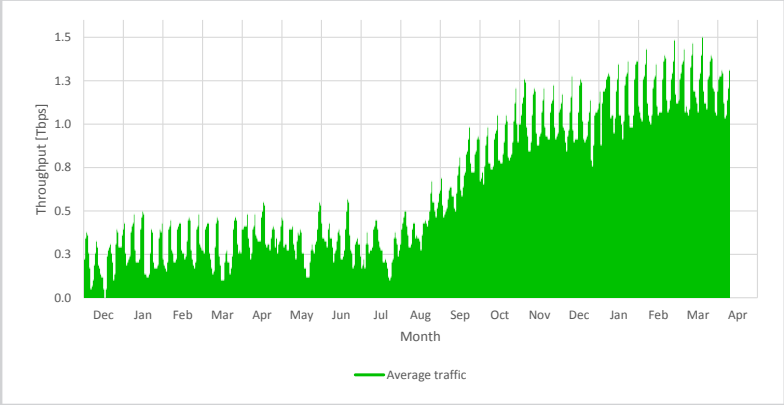


Figure 2.5

1-year graph of IPv4 traffic exchanged at AMS-IX.

IPv4 vs. IPv6 Traffic Ratio

Another metric of the IPv6 transition is the ratio of IPv6 traffic compared with IPv4. We consider the traffic exchanged at the two largest Internet exchanges (IX) in the world, DE-CIX (Frankfurt) and AMS-IX (Amsterdam), with more than 2.5 and 2.1 Tbps of peaking traffic, respectively.

Figure 2.3 and Figure 2.4 [41] show IPv4 and IPv6 traffic volumes exchanged at DE-CIX, respectively, while Figure 2.5 and Figure 2.6 [42] show the same for AMS-

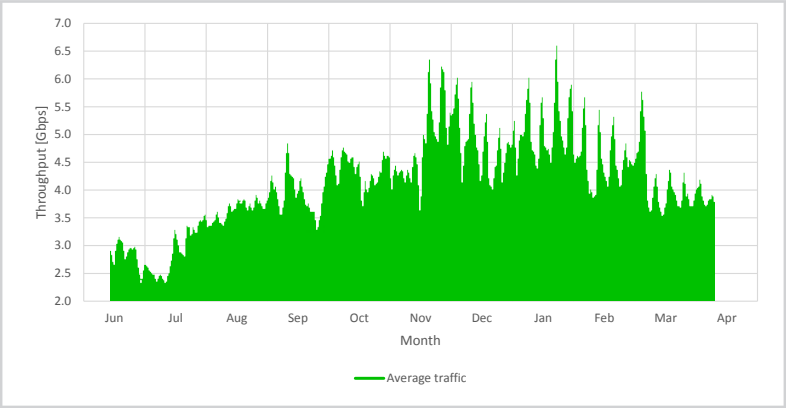


Figure 2.6

1-year graph of IPv6 traffic exchanged at AMS-IX.

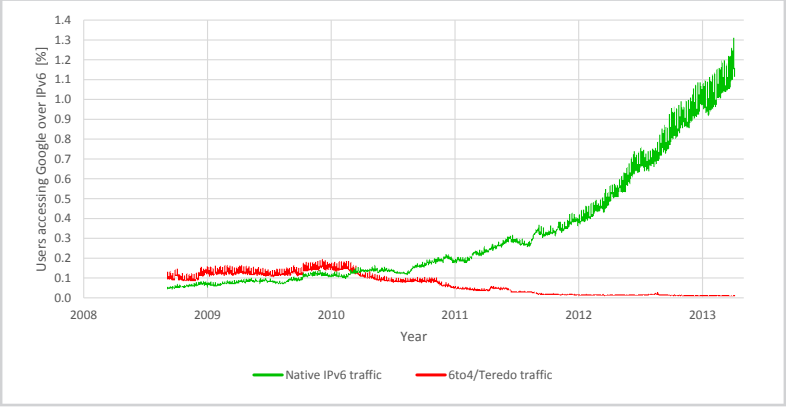


Figure 2.7

4.5-year graph of percentage of users that access Google over IPv6.

IX. The two-day average at DE-CIX is 1.591 Tbit/s for IPv4 traffic and 8 Gbit/s for IPv6 traffic, while at AMS-IX it is 1.378 Tbit/s for IPv4 traffic and 3.9 Gbit/s for IPv6 traffic (April 2013). The ratio between IPv6 and IPv4 traffic is therefore about 0.5% at DE-CIX and 0.28% at AMS-IX.

IPv6-Ready Web Users

As one of the largest Internet companies (Google Search service's market share was 66.7% in December 2012 [43]), Google's user base statistics can (effectively) be used

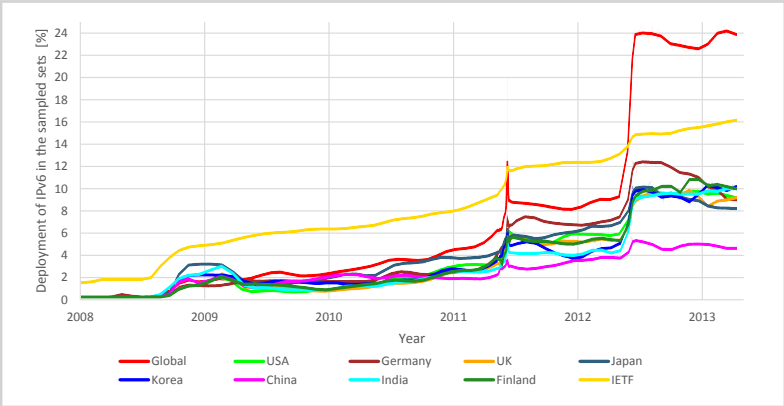


Figure 2.8

5.5-year graph of percentages of IPv6 accessible web sites from multiple Alexa top 500 lists.

to analyse the growth of IPv6-enabled web users.

Figure 2.7 [15] indicates that the number of IPv6 clients of Google's services has been growing steadily over the past few years and currently averages at 1.13% (April 2013). We can also observe, that clients using 6to4 and Teredo transition mechanisms have virtually disappeared in the last two years.

IPv6-Ready Web Sites

Alexa is a web site analytics organization that provides various lists of top web sites. Periodically testing IPv6 readiness of such lists is useful as it provides a sense of IPv6 adoption among content providers.

Figure 2.8 [16] depicts 5.5-year growth of percentages of IPv6 accessible web sites from multiple Alexa top 500 lists. The percentages of IPv6 accessible Alexa global top sites are 24% for top 500 web sites, 10.8% [44] for top 1,000 web sites and 5.27% [45] for top 1,000,000 web sites (April 2013).

2.1.2 World IPv6 Day and IPv6 Launch Events

On 8 June 2011, more than 1,000 Internet company web sites, content delivery networks, ISPs, vendors and other organizations participated at the World IPv6 Day event. Major participants included Comcast, Google, Yahoo, Facebook, Youtube, Akamai, Microsoft, AOL, etc. The goal of this 24-hour test-drive was to evaluate the "brokenness" of IPv6 clients, i.e., to analyse the issues that arise when major web sites enable

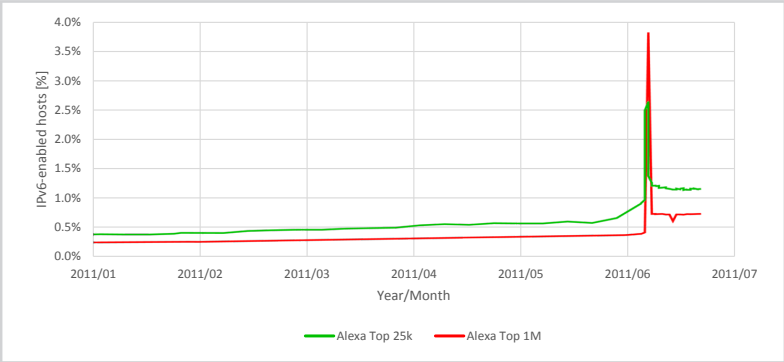


Figure 2.9

World IPv6 Day spike in percentage of IPv6-enabled web sites in Alexa top 25,000 and 1,000,000.

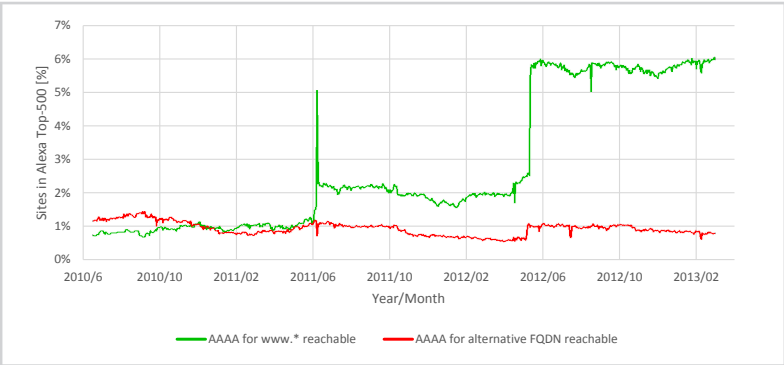


Figure 2.10

World IPv6 Day and Launch steps in percentage of IPv6-enabled web sites in Alexa top 500.

DNS AAAA (IPv6 address) records for their domains. The organizers of this influential event also wished to motivate other members of the Internet ecosystem to ready their services for IPv6 transition.

One of the notable results of World IPv6 Day is indicated in Figure 2.9 [46], where a spike in the percentage of IPv6-enabled web sites is apparent. Some of the sites have not disabled IPv6 access following the end of the test. An increase in IPv6 was also reported by Internet exchange points, e.g., one of Europe's large IX points noted a 30% increase of IPv6 traffic [47] on that day.

One year later, on 6 June 2012, a similar event occurred, called World IPv6 Launch. This time, the goal was to permanently enable IPv6 on the web sites of participat-

ing organizations. As a result, over 3,000 web sites made themselves reachable via IPv6. AMS-IX Internet exchange point observed a 50% increase in IPv6 traffic on that day [48]. The graph of the percentage for IPv6-enabled Alexa top 50 global web sites for all countries is depicted in Figure 2.10 [49]. From this graph, the "steps" for both the World IPv6 Day and World IPv6 Launch are clearly apparent. Obviously, these events were successful, as the increase in the percentage of IPv6-enabled web sites in the Alexa top 500 list was about 600% after both events.

2.1.3 *IPv6 Deployment Surveys*

First, we examine the survey of IPv6 deployment, published in June 2012 by BT Connect, which questioned 876 IT or Operations professionals about IPv6 deployment status and the future plans of their organizations regarding IPv6 deployment across various industries [50]. The key findings of this survey are the following:

- 57% of respondents have either deployed, are deploying or plan to deploy IPv6 within two years, of whom 13% have already deployed IPv6 on all or on a portion of their networks and while 44% are in the process of deploying or plan to initiate deployment within 2 years.
- Respondents still believe that stronger business case justification is needed to demonstrate a sufficient Return On Invest (ROI). 22% of respondents state that the largest hurdle to IPv6 deployment is the inability to demonstrate a clear business case, while 17% believe that the complexity of the infrastructure upgrade is the most difficult component.
- Interestingly, in a 2011 survey conducted by the same organization, 35% of the respondents expressed "huge concern" regarding IPv4 depletion, but in the latest survey, this figure dropped to 25%. This is probably because the 2011 survey was conducted only two months following the IANA IPv4 address depletion announcement.
- 55% of the respondents agree with the statement that "*IPv6 is required for deployment across the entire network*".

Next, we may review a related survey that was published by June 2012, also by GNKS Consult [51]. The numbers of respondents varied from question to question.

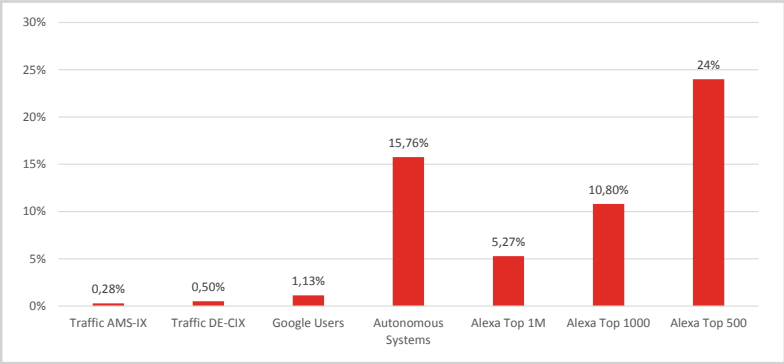


Figure 2.11

Summarized metrics of IPv6 deployment.

As a similar survey was also conducted in 2010 and 2011, year-to-year comparisons are possible. The following are the main findings of the 2012 survey:

- In 2012, 35% of ISPs stated that their IPv6 customer base is zero, while in 2011, the same was stated by 44% and in 2010 by 60% of the ISPs (n=531).
- More than 50% of respondents stated that *lack of user demand* is the greatest problem with IPv6 in production, followed by *technical problems* (less than 40%) and *no experience yet* (more than 30%) (n=1017).
- 76.5% of respondents stated that their organization has some level of IPv6 presence (48.5% both within internal and external networks, 19.5% only on the Internet, 8.5% only within internal networks) (n=1045).

2.1.4 IPv6 Deployment Summary

The number of announced IPv6 prefixes is not considered in this summary, as IPv6 address space is completely independent of IPv4 space, which means that there is no way to expect a bijective mapping between the announced IPv4 and IPv6 prefixes. Figure 2.11 depicts all metrics with their respective percentages in one graph for comparison.

From this graph, we can see that content providers (in our case, web sites) are currently doing best at transitioning their services to IPv6. Conversely, end-users (clients)

are progressing slowly, which can also be seen by very low IPv4/IPv6 traffic ratios at Internet exchanges.

We argue that the most relevant metrics for IPv6 deployment are IPv6/IPv4 traffic ratios and the percentage of hosts that access popular Internet services over IPv6. Even if IPv6 deployment at peering level (autonomous systems) were 100% and if all of the Internet services were IPv6-enabled, without IPv6-enabled hosts (end-users, clients), the IPv6 traffic flowing through Internet exchanges would be insubstantial. This effectively means that most of the Internet would still be IPv4-only.

Nonetheless, content providers achieved an important task in breaking the "chicken-or-egg-problem", as it is now obvious what part of the Internet ecosystem is most responsible for taking the next step in transitioning the Internet to IPv6. It is ISPs, bringing IPv6 connectivity into edge networks.

2.2 Review of IPv6 Deployment Mechanisms

Despite the fact that this thesis focuses mostly on IPv4 address sharing mechanisms, it is important to analyse the historical landscape of transition mechanism research and development because this enables us to better understand the reasoning behind IPv6 transition engineering. It also helps open up the transition mechanism design space, which needs to be understood when designing new mechanisms.

In this section, we review IPv6 deployment mechanisms [52]. The common denominator of these technologies is the goal to introduce IPv6 connectivity into networks which have been IPv4-only. More than a decade ago, these were very important, as only small and very scattered parts of the Internet infrastructure were IPv6-enabled and it was consequentially very difficult or even impossible to obtain native IPv6 connectivity in most places of the world. IPv6 deployment mechanisms enabled "connecting IPv6 islands in the IPv4 sea". These are still used today but practitioners will more often try to ensure native IPv6 connectivity after a decision to adopt IPv6 has been taken [50].

For the sake of completeness, we discuss all IPv6 deployment mechanisms that have been standardized (as Proposed Standard, Draft Standard, Best Current Practice) or otherwise published as RFC (as Experimental, Informational or Historic). The exception are those that can also be used for IPv4 address sharing. These will be discussed in the following chapters. We group the mechanisms into three groups:

- *Dual Stack:* a host is configured with an IPv4 and an IPv6 address and can establish and receive connections with any other host.
- *Tunneling:* IPv4 packets are tunneled within IPv6 or IPv6 packets are tunneled within other network or transport-layer protocols.
- *Translation:* IPv4 packets are translated into IPv6 packets or vice versa.

As IPv4 address sharing mechanisms are to be used by ISPs and not by edge networks, we also provide sample topologies related to ISPs environment for IPv6 deployment mechanisms. Throughout this thesis, we use the following terms for the topology of an ISP, as outlined in Figure 2.12:

- *ISP network*: the network of an Internet service provider, which also contains the access network.

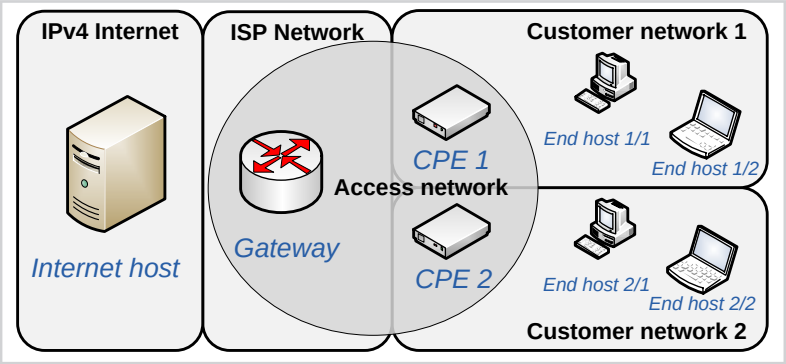


Figure 2.12

An ISP topology. For the purpose of simplicity, only two customer networks are shown, although there could realistically be thousands or millions of them.

- *Gateway*: a device in the core of the ISP network that processes customer traffic to and from the Internet.
- *Access network*: the network connecting CPEs and gateways in the ISP network.
- *CPE* (Customer Premises Equipment): a device located at the customer's premises that processes the traffic between the customer's network and the access network.
- *Customer network*: the network behind the customer's CPE, for which the ISP provides Internet access.
- *End-host*: a device which desires access to the IPv4 (and possibly IPv6) Internet residing in the customer's network.

For each mechanism we first provide a short description of the main concept. Next, we provide a timeline of IETF RFCs and Internet Drafts related to the mechanism, excluding those documents that introduce completely new ideas. Furthermore, we discuss to what extent the mechanism is adopted by relevant parties and its relevance today. We also provide a visualisation of the operation for the mechanism. It should be noted that the visualisation displayed is only one of several possible scenarios. Finally, we discuss additional points or issues that are related to the mechanism.

2.2.1 *Dual Stack*

Overview. The idea of the "dual stack" transitioning scheme [14, 53, 54] (which is not the same as the Dual-Stack Transition Mechanism (DSTM [55]) is straightforward:

Every host in a dual stack network must implement both the IPv4 and IPv6 stack, have both IPv6 and IPv4 addresses configured and have its networking applications support both IPv6 and IPv4. Only such hosts can seamlessly establish and receive connections to/from IPv6 or IPv4 hosts. In this sense, the dual stack concept is also present in many other tunneling mechanisms. Deploying native IPv4 and IPv6 connectivity is the most recommended approach in the transition to IPv6. However, in some cases, some parts of the network infrastructure cannot be IPv6-enabled and therefore other transition mechanisms are required.

IETF Timeline. Figure 2.13 shows a timeline of published IETF RFC and Internet Draft documents related to the dual stack transition mechanism. From this graph, we can see that the dual stack approach had already been defined in 1996. It has inspired several other RFC documents over the past 18 years [56–64].

Penetration. The dual-stack is the leading transition/co-existence mechanism among other strategies. 21.5% of ISPs and 28.5% of questioned enterprises, stated that they have used or plan to implement the dual stack mechanism throughout their networks. 12.5% of ISPs have deployed or plan to deploy dual stack to customer-facing networks only and 7% of ISPs have implemented or consider implementing dual stack in their backbone only. 16.5% of enterprises have deployed or plan to deploy dual stack to Internet-facing servers only [50].

Operation. Figure 2.14 depicts the difference between IPv4 single stack and IPv4/IPv6 dual stack host configurations. Every application running on such hosts can leverage both IP stacks.

Miscellanea. There are nevertheless some caveats with dual stack deployments. For example, an intelligent algorithm for dual stack hosts connecting to dual stack servers is required. If a dual stack server has (possibly temporary) issues with IPv6 connectivity, dual stack hosts will experience significant delays before "failing back" to IPv4 due to TCP retransmission and timeout delays. To resolve this, IETF has proposed a "Happy Eyeballs" algorithm [64] which establishes two simultaneous TCP connections from client to server, one over IPv4 and the other over IPv6. Most web browser implementations (Chrome, Opera, Firefox) prefer IPv6 connection to IPv4, but some (Mac OS X Lion) prefer whichever is fastest. After the preferred transport connection is established, it is used for higher-layer protocol operation.

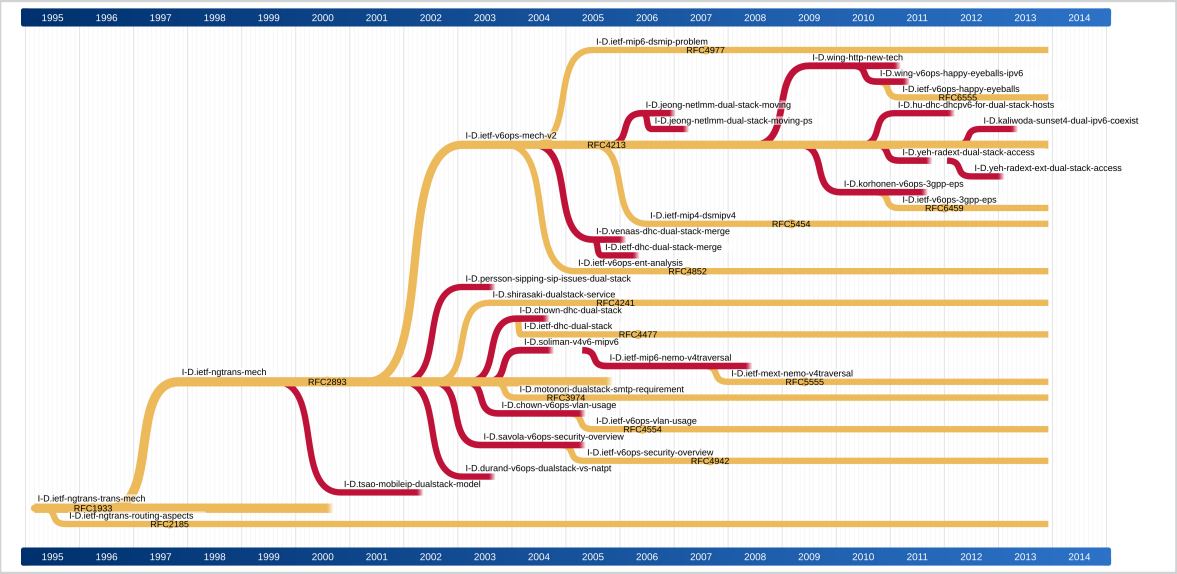


Figure 2.13

Timeline of published IETF RFC and Internet Draft documents related to dual stack mechanism.

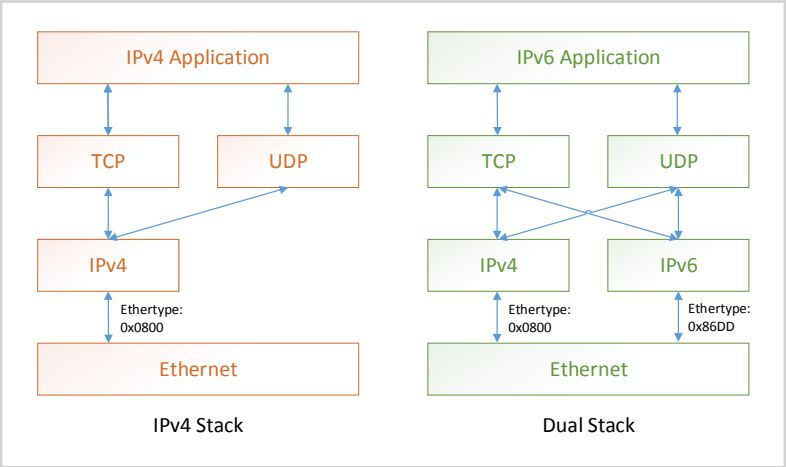


Figure 2.14

IPv4 single stack vs. IPv4/IPv6 dual stack host configuration on an Ethernet link.

If both IPv4 and IPv6 connections get established simultaneously, the non-preferred one gets torn down and is not used for that specific higher-layer session.

2.2.2 *Tunneling*

Tunneling is the process of encapsulating, transporting and decapsulating a packet in order for a packet to cross heterogeneous networks. The "outer" protocol is referred to as the tunneling protocol while the "inner" protocol is referred to as the tunnelled protocol. Normally, the tunneled protocol header remains unchanged (hop counts are not incurred) while being transported over non-native networks. In the context of IPv6 transition, we consider IPv4-in-IPv6 and IPv6-in-IPv4 tunneling types. Traditionally, tunnels are either configured or automatic [65]. Configured tunnels require that administration–tunneling interfaces be configured at both tunnel endpoints. In contrast, automatic tunneling infers tunnel endpoints from the addresses themselves. An IPv6 packet is encapsulated within an IPv4 packet using protocol number 41 and an IPv4 packet is encapsulated within an IPv6 packet using the Next Header 4. In this section, we consider the tunneling mechanisms proposed by IETF.

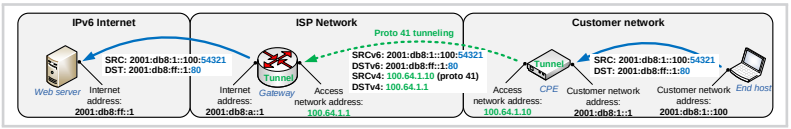


Figure 2.15

In the configured tunnels mechanism, IPv6 traffic is normally tunneled in IPv4 packets using protocol 41 encapsulation.

Configured Tunnels

Overview. As the name indicates, the tunnels must (manually) be configured before any packets can be exchanged between hosts. Configured tunnels offer a straightforward method of using existing IPv4 infrastructure to carry IPv6 traffic to IPv6 islands [14, 53, 54]. Each of the tunnel endpoints must configure four parameters: IPv4 and IPv6 source and destination addresses of the endpoints.

IETF Timeline. Figure 2.16 shows the timeline of published IETF RFC and Internet Draft documents that are related to the configured tunneling transition mechanism. From this, it is apparent that the configured tunneling technique has already been defined in 1996. It has inspired several other RFC documents over the past 18 years [53, 54, 56, 66–74].

Penetration. Configured tunnels are widely implemented because many operating systems, IPv6 routers and home gateways support protocol 41 tunneling. 7.5% of the enterprises questioned, stated that they have already leveraged or are considering leveraging configured tunnels to achieve IPv6 connectivity [50].

Operation. Figure 2.15 shows the required steps for transferring a packet from an IPv6 end-host to an IPv6 web server on the Internet over a configured tunnel. The user's tunnel endpoint is configured on his or her CPE (home gateway), whereas the ISP's tunnel endpoint is located on ISP's core network.

Miscellanea. An important disadvantage of configured tunnels is that paths are normally suboptimal, as they are defined in advance and are static for all packets that are sent through the tunnel. On the other hand, automatic tunnels are established and torn down on-the-fly. Normally, protocol 41 must be allowed on the path between endpoints, which causes problems with traversing NAPT devices given that some only support the traversing of TCP, UDP and ICMP transport-layer protocols. Protocol 41 tunnels are sometimes also referred to as 6in4 tunnels.

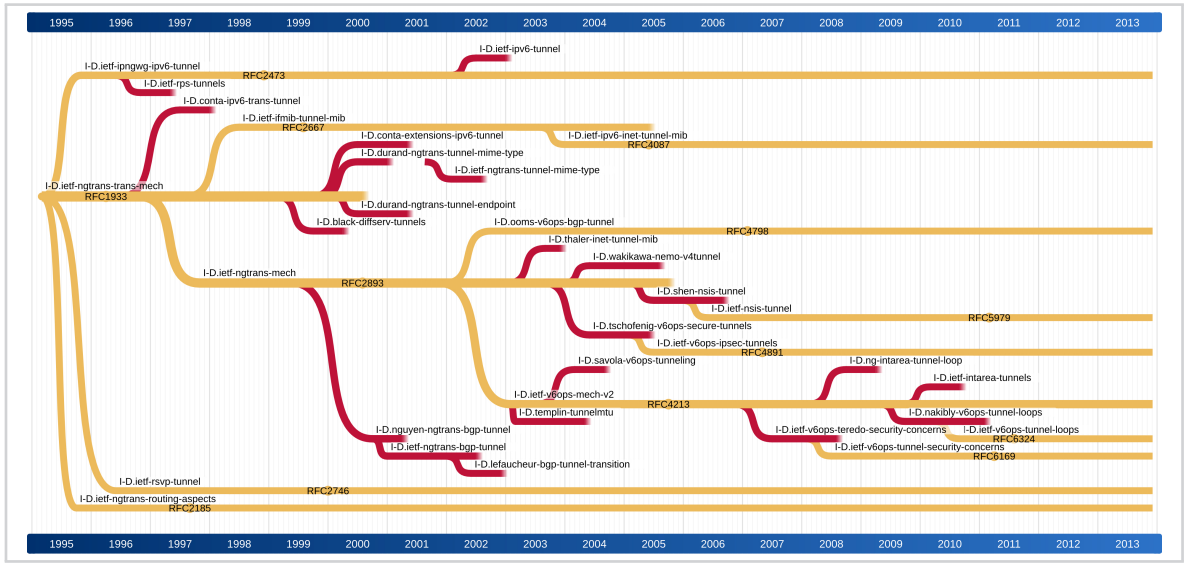


Figure 2.16

Timeline of published IETF RFC and Internet Draft documents related to configured tunneling mechanism.

However, other types of encapsulation can also be used for configured tunnels, like GRE and MPLS. Because of the administration complexity they incur, configured tunnels generally do not scale well.

6to4

Overview. 6to4 [75] is an automatic tunneling mechanism. Every IPv4 host with a public IPv4 address can play the role of a tunnel endpoint for 6to4. A special IPv6 prefix 2002::/16 is reserved for the 6to4 mechanism and it is considered reachable through the tunnel interface of a 6to4 host as a non-broadcast, multiple access [76] (NBMA) network. Every 6to4 tunnel endpoint automatically creates a /48 IPv6 prefix, concatenating the IPv4 address to the 2002::/16 prefix. 6to4 uses protocol 41 for encapsulating IPv6 packets into IPv4 packets. 6to4 uses anycast mechanism [77] for 6to4 relays that provide connectivity between the 6to4 network and IPv6 Internet. Only hosts with private [78] IPv4 addresses cannot use the 6to4 mechanism, as they are not unique and would yield duplicate 6to4 prefixes.

IETF Timeline. Figure 2.17 shows a timeline of published IETF RFC and Internet Draft documents related to the 6to4 tunneling transition mechanism. The timeline indicates that configured tunneling technique had already been defined in 2000. It has inspired several other RFC documents over the past 18 years [79–82].

Penetration. According to Google's users statistics [15], only 0.01% of Teredo *and* 6to4 users access Google services. On the other hand, 7.5% of questioned enterprises stated that they have deployed or will deploy 6to4 tunneling in order to access IPv6 Internet.

Operation. Figure 2.18 shows an end-host behind a CPE with a public IPv4 address that is configured with 6to4 sending an IPv6 packet to IPv6 Internet via 6to4 relay. All public relays are reachable at the anycast address 192.88.99.1, which corresponds with 2002:c058:6301::. Relays advertise the 2002::/16 prefix towards the IPv6 Internet, so that the packets destined to hosts with 6to4 addresses are routed to the nearest 6to4 relay. In the opposite direction, the relay tunnels packets over IPv4 to the IPv4 address of destination 6to4 CPE, which is encoded in the IPv6 address.

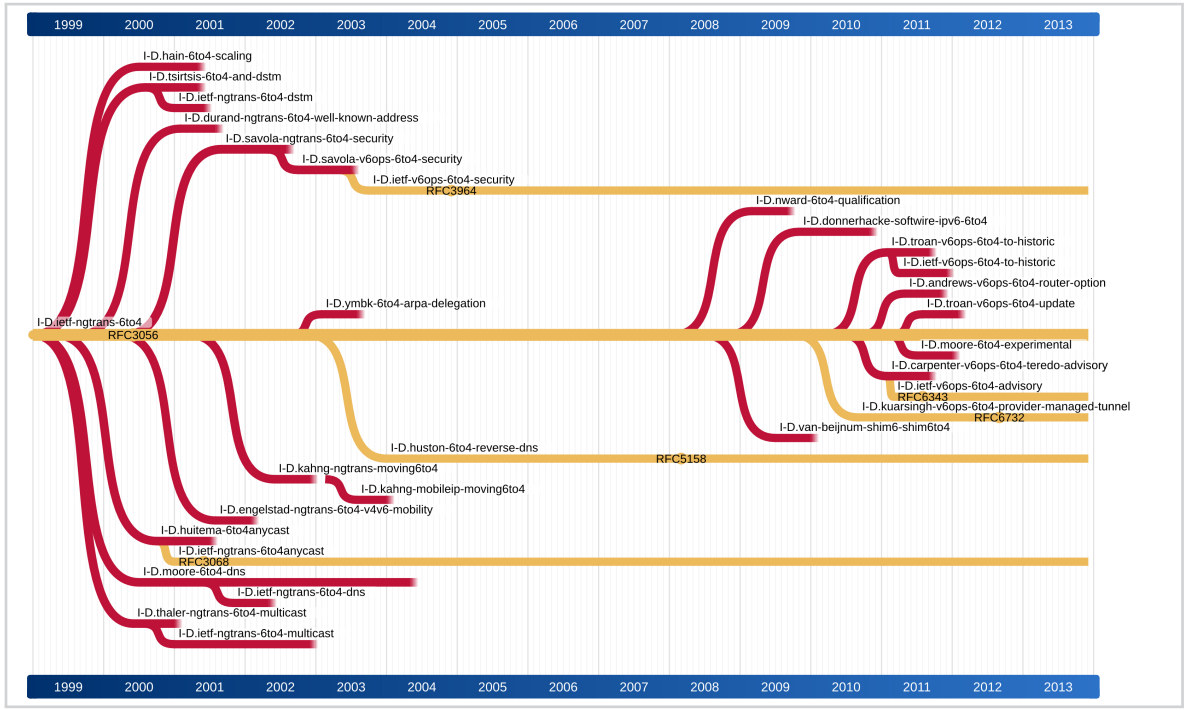


Figure 2.17

Timeline of published IETF RFC and Internet Draft documents related to 6to4 tunneling mechanism.

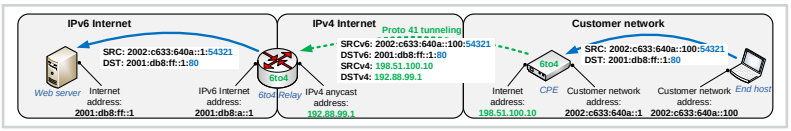


Figure 2.18

In the 6to4 transition mechanism, IPv6 traffic is tunneled to the nearest 6to4 relay using protocol 41.

6to4 hosts have a default route directed to 2002:c058:6301::, which means that they tunnel packets destined to non-6to4 IPv6 hosts to the nearest relay, which in turn decapsulates the packet and sends it off as an IPv6 packet.

Miscellanea. The main issue with 6to4 is that the 6to4 anycast relay is chosen on-the-fly, which introduces a lot of unpredictability. Often, public 6to4 relays are provided by third parties on a best-effort basis, which means they can be very slow. Consequently, asymmetric routing is common, which means that two different 6to4 relays can be used, one for each direction between two hosts. Because of these problems, many people consider 6to4 to be harmful [83, 84], which has led to initiatives to declare 6to4 obsolete [85]. However, the final solution was to deprioritize 6to4. The latest Default Address Selection algorithm for IPv6 [86] mandates that hosts prefer native IPv6 over IPv4 and IPv4 over 6to4 IPv6. Therefore, 6to4 is used if and only if a destination host is not reachable over IPv4 and no other IPv6 connectivity is available. Other 6to4 considerations are documented in RFC 6343 [81]. Recently, IANA has reserved an additional IPv4 prefix for additional shared address space among ISPs. This address space is similar to RFC1918 private address space in the sense that it is not globally unique. This clashes with 6to4 as it is only disabled for RFC1918 addresses.

6rd

Overview. As lack of control over the paths from 6to4 clients to 6to4 relays was discovered to be problematic, IPv6 rapid deployment (6rd) [87, 88] proposes moving the relay into the ISPs network and advertising ISP's own IPv6 prefix rather than a well-known 6to4 prefix 2002::/16. In this way, the operational domain of 6rd is limited to the ISP network and is under its control, which also means that RFC1918 address space can be used in the access network between the CPE's and ISP's core network. Also, the users are not seen by the Internet as "6to4 users" because they are provisioned with ISP's prefix. For IPv6 edge networks and IPv6 Internet, the IPv6 service is equivalent to native IPv6. However, as there is no well-known prefix hardcoded into

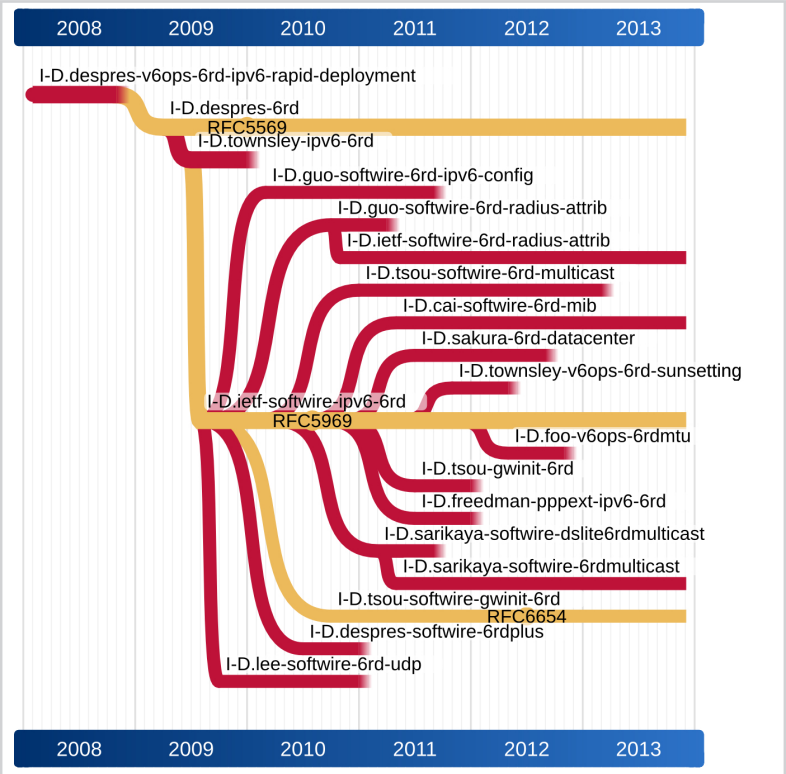


Figure 2.19

Timeline of published IETF RFC and Internet Draft documents related to 6rd tunneling mechanism.

6rd devices, these must be provisioned with one. After the devices are provisioned with a 6rd prefix, subsequent operation is completely stateless, given that IPv4 tunnel endpoints are algorithmically derived from IPv6 prefixes.

IETF Timeline. Figure 2.19 shows a timeline of published IETF RFC and Internet Draft documents related to the 6rd tunneling transition mechanism. The first initiatives to developing a "local" version of 6to4 began relatively late, in 2008. This has inspired one new RFC document [89] and more than a dozen Internet Drafts.

Penetration. *Free*, a major French ISP (then with something less than 20% of the broadband market share) implemented and deployed 6rd in December 2007, even

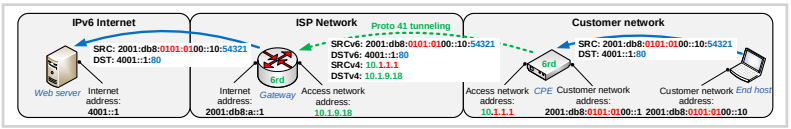


Figure 2.20

In 6rd transition mechanism, IPv6 traffic is tunneled in IPv4 using protocol 41 to ISP's 6rd gateway.

prior to the IETF publishing the first draft. By the end of 2008, 95% of French IPv6 traffic to Google came from Free customers [90]. Since then, several other large ISPs have publicly announced 6rd is being considered for deployment or will definitely be deployed in the near future. However, Comcast, for example, announced plans to deactivate 6rd in favour of the dual stack mechanism. Linux kernel includes a 6rd implementation since version 2.6.33 (December 2010).

Operation. Figure 2.18 shows an end-host behind a CPE with a RFC1918 or public IPv4 address, which is configured with 6rd. This means that it was also provisioned with a 6rd IPv6 prefix, IPv4 mask length and gateway IPv4 address. For incoming packets, gateway constructs the IPv4 address of the destination CPE from the destination IPv6 address of the packet on-the-fly for each packet individually and statelessly. This IPv4 address is then used for encapsulating the IPv6 packet and sending it off to the CPE.

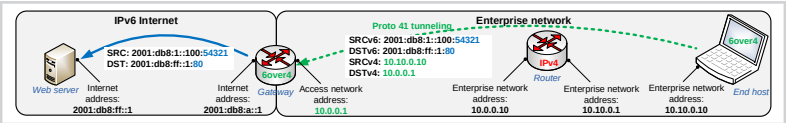
Miscellanea. 6rd is also less prone to traffic anonymization attacks than is 6to4 because all of the end-hosts are under ISP's control [88].

6over4

Overview. The purpose of 6over4 [91], also known as *IPv4 multicast tunneling* or *virtual Ethernet*, is to establish the IPv6 network as an overlay over the existing IPv4 network infrastructure. Any dual stack node implementing IPv6 and 6over4 in addition to IPv4 can leverage Neighbour Discovery [92] and Stateless Address Autoconfiguration [93], sending IPv6 packets to other such nodes in the network or to the IPv6 Internet, if native IPv6 Internet gateway sending Router Advertisements is provided at the border of the network. The main concept of 6over4 is to use IPv4 infrastructure as a link-layer network and then tunnel IPv6 packets on top of that infrastructure using protocol 41 encapsulation. In order to achieve this, the network must fully support IPv4 multicast routing.

Figure 2.21

In 6over4 transition mechanism, IPv6 traffic is tunneled in IPv4 using protocol 41 on top of IPv4 infrastructure using IPv4 multicast.



IETF Timeline. The blue domain in Figure 2.22 shows a timeline of published IETF RFC and Internet Draft documents that are related to the 6over4 tunneling transition mechanism. The first draft was already published in 1997. Only one RFC was published related to 6over4.

Penetration. 6over4 is mostly irrelevant today because it requires an IPv4 infrastructure that fully supports rarely deployed IPv4 multicast routing. It is also relatively easy to deploy local IPv6 communication using IPv6 routing. Finally, 6over4 was replaced by ISATAP [94], which is more complex, but does not require IPv4 multicast.

Operation. Figure 2.21 shows an enterprise network connected to IPv6 Internet using a 6over4-enabled dual stack border router. The IPv6 packets are tunneled using protocol 41 in IPv4 packets. A 6over4-enabled end-host can send packets to IPv6 Internet by encapsulating the packet and sending it off to its' default IPv6 gateway. The IPv4 address of the gateway is discovered using Neighbour Discovery by IPv4 multicasting.

Miscellanea. Hosts can only take advantage of 6over4 if they implement it themselves. This is a tough requirement as changes to hosts' network stacks are required (which means operating system upgrades). Also, 6over4 does not cross over NAPTs within a network because the IPv4 address exchanged through Neighbour Discovery is different from the one that is needed to reach the destination host.

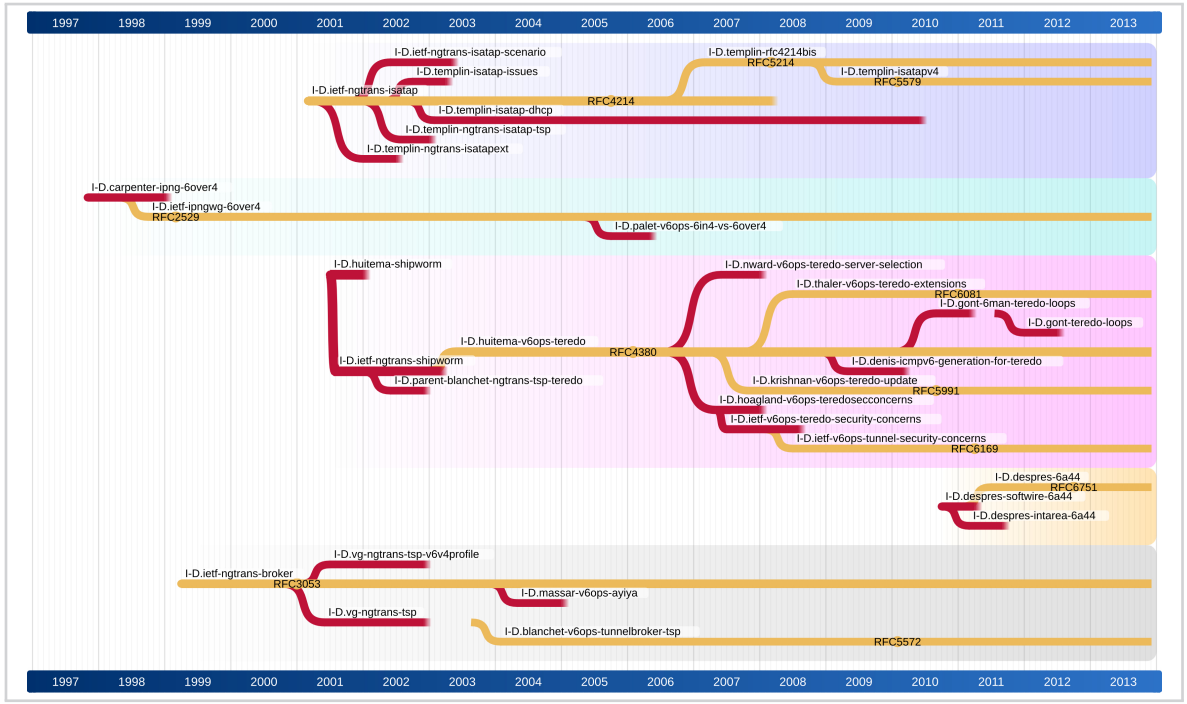


Figure 2.22

Timelines of published IETF RFC and Internet Draft documents related to ISATAP (blue), 6over4 (green), Teredo (red), 6a44 (orange) and tunnel broker (grey) mechanisms.

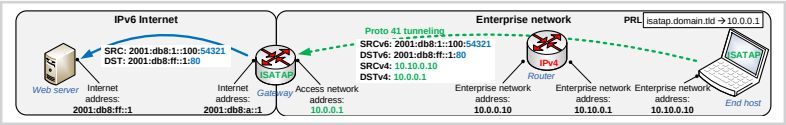


Figure 2.23

In the ISATAP transition mechanism, IPv6 traffic is tunneled in IPv4 using protocol 41 on top of IPv4 infrastructure using a preconfigured Potential Router List (PRL).

ISATAP

Overview. ISATAP [94, 95] is conceptually similar to 6over4, however it does not require IPv4 multicasting. Instead, ISATAP uses an NBMA communication model and requires certain preconfiguration. With ISATAP, hosts are assigned IPv6 addresses whose interface identifier is defined by a unique IPv4 address only. As ISATAP does not rely on IPv4 multicast, it must be provisioned with a Potential Router List (PRL) entirely over IPv4 using DHCP, DNS or manually. The most common and preferred method is DNS – a DNS operator needs to add a new Resource Record into DNS configuration for the domain in question in the form of "isatap.domain.tld". ISATAP hosts will only accept Router Advertisements sourced by any of the servers on the PRL. The IPv6 prefix of size /64, obtained from the router, together with the IPv4 address define the IPv6 address of the host's ISATAP interface. Finally, the IPv4 address encoded in the destination IPv6 address is used to initiate Neighbour Discovery. All IPv6 packets are tunneled in IPv4 packets using protocol 41.

IETF Timeline. The green domain in Figure 2.22 shows a timeline of published IETF RFC and Internet Draft documents related to the ISATAP tunneling transition mechanism. The first draft was already published in 2001. This draft has inspired one new RFC document [96] and a few Internet Drafts.

Penetration. ISATAP is supported by major operating systems like Microsoft Windows, Linux and Cisco IOS, which means it is available for workstations, servers and routers. However, only 2.5% of questioned enterprises stated, that they have already deployed or are considering deployment of ISATAP in their networks[50]. Today, IPv6 native routing is relatively easy to deploy, which reduces the attractiveness of ISATAP.

Operation. Figure 2.23 shows an ISATAP end-host with a configured PRL list, obtained using one of the provisioning mechanisms (usually DNS). After a PRL is con-

structed using the mechanism of choice, the end-host sends a Router Solicitation to receive a unicast Router Advertisement packet with a /64 prefix from an ISATAP router, which enables the end-host to construct its ISATAP IPv6 address. From then on, it is able to participate in IPv6 networking.

Miscellanea. ISATAP works fine with RFC1918 addresses, but cannot cross NAPT devices within the network because IPv4 addresses are encoded in IPv6 addresses. As with 6over4, each host must implement the ISATAP protocol in order to participate in networking. Due to the common practice of building the PRL using DNS, ISATAP is criticized for violating the OSI model [1]–a lower-layer protocol (IPv6) is relying on a higher-layer protocol (DNS), even though IPv4 DNS servers are used and no circularity is induced.

Teredo

Overview. Teredo [97] is an automatic tunneling mechanism that works through NAPTs. It was intended to be a temporary measure in scenarios where native IPv6 connectivity is not yet available. IPv6 packets are not encapsulated using protocol 41 but rather, in UDP packets, which allow multiple Teredo clients behind the same NAPT device. In order to traverse NAPTs, an additional component in addition to a Teredo relay is required. Teredo servers mustbe configured in Teredo clients, which are used for the detection of the types of NAPTs that separate the client from the Internet as well as for "hole punching" technique [98]. For detection, a qualification procedure based on now obsolete NAPT classification [99, 100] is performed by the server. However, Teredo is unable to traverse symmetric NAPTs properly [101]. Teredo clients are allocated an IPv6 address from the IANA-assigned Teredo prefix 2001::/32.

IETF Timeline. The red domain in Figure 2.22 shows a timeline of published IETF RFC and Internet Draft documents related to the Teredo tunneling transition mechanism. The first draft was already published in 2001. It has inspired a few other RFC documents [73, 102, 103].

Penetration. Teredo is strongly present in Microsoft Windows operating systems. Teredo was already available in Windows XP and has been enabled by default since the launch of Windows Vista. Given that the Windows market share is more than

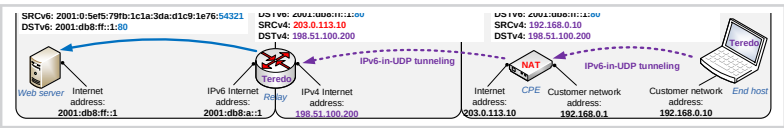


Figure 2.24

In the Teredo transition mechanism, IPv6 traffic is tunneled in UDP packets to a Teredo relay.

90% [104], one would expect most of today's IPv6 traffic to be Teredo. However, according to Google's users statistics [15], only 0.01% of Teredo *and* 6to4 users access Google services, which means that Teredo usage is globally very low. Another report indicates similar results – 0.01% of clients use Teredo to access a dual stack web server with IPv6.

Operation. Figure 2.24 shows a Teredo end-host behind a NAPT sending an IPv6 packet to IPv6 Internet via Teredo relay. In the initialization phase, the type of NAPT in front of the client is determined. Next, a Teredo IPv6 address is assigned to the client. Finally, a list of relay servers is configured by the server. Communication between two NAPT-ed Teredo clients is also possible using the "hole punching" technique with the help of Teredo server.

Miscellanea. As Teredo seems to be relatively a unreliable and poorly performing mechanism [105], it is used as a last resort for connecting to IPv6 destinations. It is reported that Teredo has only a 63% success rate when connecting to native IPv6 destinations. It turns out that Teredo is only used when connecting to IPv6 literals. In such cases, 20% to 30% of clients will opt to use Teredo to access this service [106].

6a44

Overview. While 6rd is a "local version" of 6to4, 6a44 [107] is a "local version" of Teredo. The goal of the 6a44 mechanism is to enable ISPs to provide IPv6 connectivity for their customers even if they are using only IPv4-only NAPT CPE devices. ISP needs to provide a 6a44 relay in the core network and a 6a44 client in the customer's network. A /48 prefix from ISP's address space is used for 6a44 instead of Teredo's well-known 2001::/32. Similar to 6to4, a well-known IPv4 anycast address 192.88.99.2 must be configured on 6a44 relay within ISP's network. In 6a44, it is the relay which designates IPv6 addresses to clients and not the server (because there is no server).

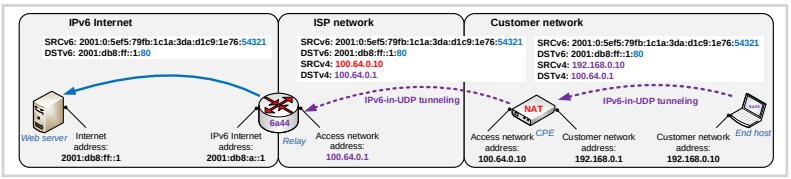


Figure 2.25

In the 6a44 transition mechanism, IPv6 traffic is tunneled in UDP packets to a 6a44 relay in ISP network.

IETF Timeline. The orange domain in Figure 2.22 depicts the timeline of published IETF RFC and Internet Draft documents related to the 6a44 tunneling transition mechanism. 6a44 is a new mechanism and has been published in 2012.

Penetration. As of April 2013, no implementations of the 6a44 mechanism are known to exist. Since the 6a44 RFC does not introduce a standard of any kind, but has rather been published as "Experimental", it is possible that it will be overlooked by the vendor community. The main reason is the requirement of 6a44 clients, which means that major operating system vendors would need to adopt it as well – which is fairly improbable. The community is also much more focused on IPv4 address sharing mechanisms today. Deploying native IPv6 is not as difficult as it was a decade ago because major network hardware vendors already support IPv6.

Operation. Figure 2.25 shows a 6a44 end-host behind a NAPT sending an IPv6 packet to IPv6 Internet via 6a44 relay in the ISP's network. First, the 6a44 client and the 6a44 relay exchange initialization packets that are also known as "6a44 bubbles". IPv6 prefix is sent from the relay to the client, which is used to calculate or confirm its 6a44 IPv6 address. Every 30 seconds, a keepalive packet is sent from the client to the relay in order to ensure that the NAPT mapping(s) on possible NAPT devices is kept between them.

Miscellanea. 6a44 works with endpoint-dependent and endpoint-independent mappings including NAPTs which change the behaviour from one to the other.

Tunnel Broker

Overview. Tunnel broker [108] services have globally been offered by various organizations for a significant period of time. The idea of tunnel broker services is to offer an IPv6-in-IPv4 tunnel provisioning service to end-hosts and routers that are unable to

Figure 2.26

A tunnel broker scenario, where IPv6 traffic is tunneled from the CPE to the tunnel broker's Point of Presence using protocol 41 encapsulation.

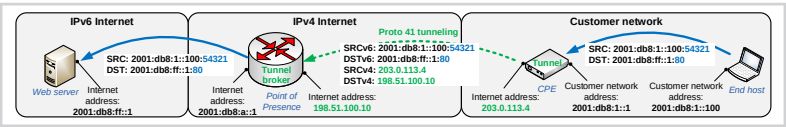
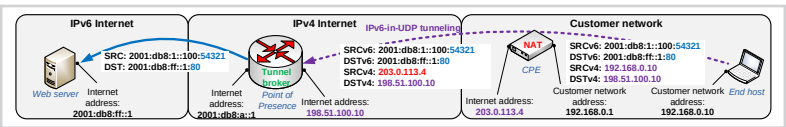


Figure 2.27

A tunnel broker scenario, where IPv6 traffic is tunneled from the end-host to the tunnel broker's Point of Presence using IPv6-in-UDP encapsulation.



connect to IPv6 natively. An administrator can establish an IPv6 tunnel with the broker within minutes without any contracts or dedicated support from the broker. This is the main difference between configured tunnels and the tunnel broker service. Traditionally, tunnel brokers offer protocol 41 tunnels, but some also offer IPv6-in-UDP encapsulation tunnels. This section discusses all kinds of tunnel broker services.

IETF Timeline. The grey domain in Figure 2.22 shows a timeline of published IETF RFC and Internet Draft documents related to the tunnel broker transition mechanism. The first draft was already published in 1999. It has inspired one more RFC document [109].

Penetration. Hurricane Electric is the largest tunnel broker service provider by the number of tunnels hosts (72.022), IPv6-in-IPv4 tunnels, spanning 185 countries as of April 2013 [110]. 44.97% of these are established within the United States. Over the past 20 months, the number of registered users at the tunnel broker service grew from 201.336 to 412.667. Each month, approximately 8.000 new users sign up for this service [111]. The second largest tunnel broker service provider, SixXS, hosts 39.049 tunnels spanning 124 countries and has 36.132 users spanning 145 countries as of April 2013. These figures show that tunnel broker services are still widely used today.

Operation. Figure 2.26 shows a scenario where the user's CPE is configured with a tunnel to a tunnel broker service using protocol 41 encapsulation of IPv6 packets. Usually, such tunnels must be configured by the administrator. Figure 2.27 shows a similar scenario, but here it is the end-host which has established a direct tunnel to

the tunnel broker. If the end-host is behind a NAPT, an IPv6-in-UDP (supported by TSP and AYIYA) is suggested, otherwise protocol 41 traversal must be supported by the NAPT device.

Miscellanea. From the tunnel broker service provider point of view, tunnels are configured automatically. An administrator usually registers a new tunnel over a web interface. He can then configure the tunnel in two ways: either manually (usually by running a set of commands on the end-host or router), or automatically (using a client like AICCU [112] which supports one of the tunnel provisioning protocols like Tunnel Setup Protocol [109] or Tunnel Information and Control protocol [113] or Anything in Anything (AYIYA) [114]). Heartbeat [115] protocol enables dynamically addressed IPv4 hosts to use a tunnel broker.

2.2.3 Translation

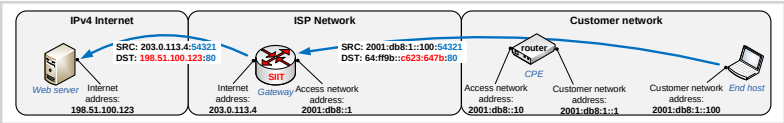
By translation we refer to IPv4-to-IPv6 and IPv6-to-IPv4 header (inter-protocol) translation mechanisms. This does not include mechanisms that only perform address (intra-protocol) translation. The purpose of translation in IPv6 deployment mechanisms is the same as with tunneling: to transport the packets of one address family over the network of a different protocol family. However, different translation mechanisms operate at different layers: network layer (SIIT, NAT-PT, BIH), transport layer (TRT) and application layer (BIH, SOCKS64). Protocol translation is usually considered to be the least effective of the three methods (dual stack, tunneling, translation) given that it breaks numerous concepts of IP networking: hierarchical routing, expanded address space, streamlined packet headers, IPv6 mobility and finally incompatibility with application-layer protocols that in any case use IP address information written in packet headers for their operations.

SIIT

Overview. Stateless IP/ICMP Translation (SIIT) [116, 117], also known as "stateless NAT64", is an algorithm for translating IPv4 packet headers into IPv6 packet headers and vice versa. It is a network-layer translator. Using a SIIT translator between them, an IPv4-only and an IPv6-only host can send packets to each other. The SIIT specification does not itself address routing or DNS considerations, rather, only the translation algorithm is defined. If we want to use SIIT to make IPv4 Internet available to IPv6-

Figure 2.28

In the SIIT transition mechanism, IPv6 traffic is translated to IPv4 traffic by a SIIT translator using one-to-one IPv4-to-IPv6 mappings.



only hosts, then we must to ensure one-to-one mappings between the IPv6 addresses and public IPv4 addresses of the hosts. The reason is that SIIT is stateless and therefore does not support any transport-layer port multiplexing (NAPT) nor does it effectively support IPv4 address sharing. In order to support address sharing, we must use Stateful NAT64 [30], which is discussed in the following chapters. To estabish a usable IPv4 Internet connectivity for IPv6-only hosts, we must also enable a DNS64 [118] server, which provides IPv6-only hosts with synthetic IPv6 addresses of the destination hosts. These addresses are generated from a site-specific or well-known [119] NAT64 prefix (64:ff9b::/96) and an IPv4 address, usually returned by public DNS servers for IPv4-only Internet hosts.

IETF Timeline. The blue domain in Figure 2.29 shows a timeline of published IETF RFC and Internet Draft documents that are related to the SIIT transition mechanism. The first RFC was already published in 2000. It has inspired a few other RFC documents [39, 117, 119, 120].

Penetration. SIIT is made available by major network equipment vendors like Cisco and Juniper. A userland open source implementation is also available [121]. 8.5% of questioned enterprises stated that they have already deployed or are considering deployment of "Network translation (NAT or SOCKS)", which also includes SIIT [50]. According to the testimony of an IT practitioner from a major mobile ISP, SIIT is also popular among mobile providers who have enough public IPv4 addresses.

Operation. Figure 2.28 depicts a scenario where IPv6 traffic coming from an end-host is algorithmically translated to IPv4 traffic by the appropriate replacement of IP headers. The CPE only performs the functions of IPv6 routers. A well-known prefix 64:ff9b::/96 is used in this example.

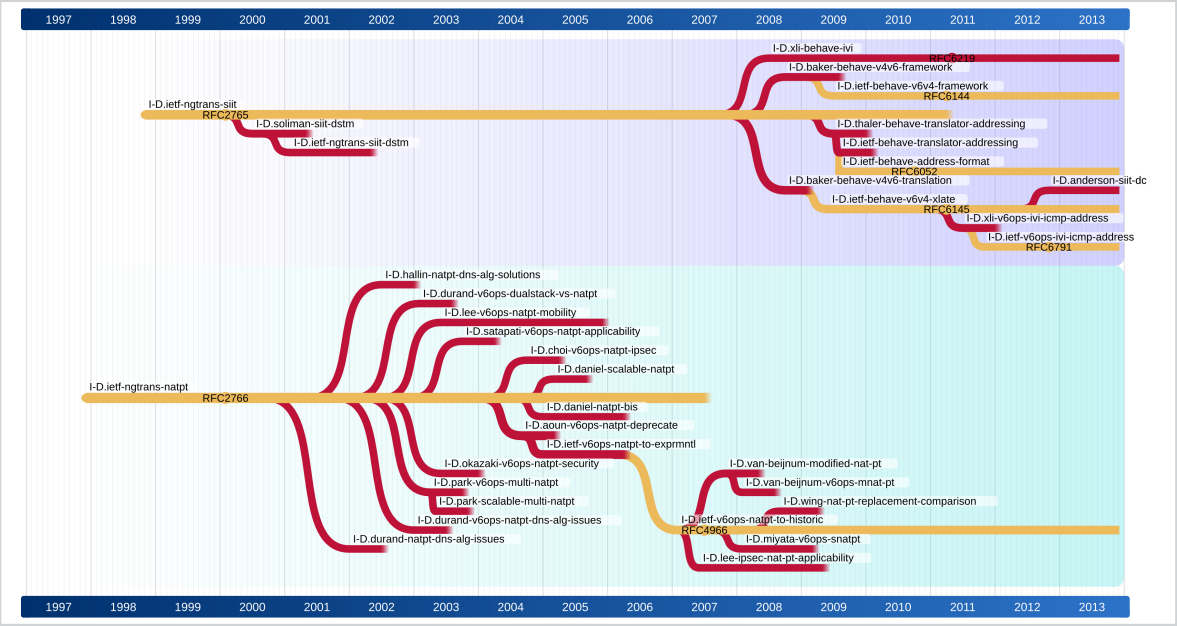


Figure 2.29

Timelines of published IETF RFC and Internet Draft documents related to SIIT (blue) NAT-PT (green) mechanisms.

Miscellanea. It is possible to "simulate" the Stateful NAT64 [30] mechanism by chaining SIIT and ordinary IPv4 NAPT [20], which effectively provides us with an IPv4 address sharing mechanism. The idea is to translate IPv6 headers into IPv4 headers with RFC1918 addresses and then to use a classic IPv4 NAPT translator to multiplex RFC1918 on one or more IPv4 addresses. To avoid transport-layer checksum recalculation, checksum-neutral prefixes can be used for translation. SIIT can be used in four different scenarios: "an IPv6 network to the IPv4 Internet", "the IPv4 Internet to an IPv6 network", "an IPv6 network to an IPv4 network", and "an IPv4 network to an IPv6 network" [39]. A similar mechanism to SIIT, titled IVI, was also developed and proposed to IETF. IVI also includes IVI DNS, which is a DNS A-to-AAAA translator intended for use together with IVI. However, this was never standardized, although it has been used in production [122].

NAT-PT

Overview. RFC2766 [28] defines three functions for IPv6 transition:

- *Network Address Translation/Protocol Translation (NAT-PT)*: provides stateful IPv6-to-IPv4 translation with IPv4 address sharing, which means that it allows for transport-layer port multiplexing of multiple IPv6 addresses on one or more IPv4 addresses.
- *Network Address Port Translation and Protocol Translation (NAPT-PT)*: is similar to NAT-PT, but it also translates transport-layer port numbers to avoid port number collisions which might break applications or cause security issues.
- *DNS Application Layer Gateway (DNS-ALG)*: a A-to-AAAA DNS Resource Record translator which contains various gaps and problems that are hard to overcome.

Although NAT-PT can be used as an IPv4 address sharing mechanism, we discuss it as an IPv6 deployment mechanism because it was envisioned as such back in the late 1990s and because it was deprecated by IETF in 2007 [29]. NAT-PT has since been replaced by various other methods, notably by Stateful NAT64, which is defined for a narrower set of scenarios.

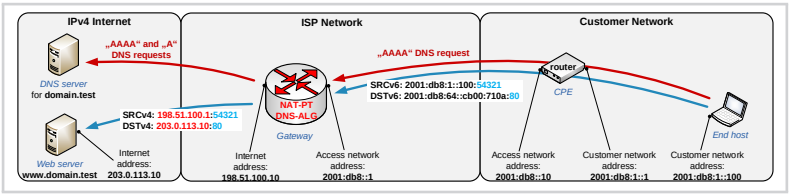


Figure 2.30

In the NAT-PT transition mechanism, IPv6 traffic is translated to IPv4 traffic by a NAT-PT translator using dynamic IPv4-to-IPv6 mappings. DNS-ALG is also used to translate DNS requests.

IETF Timeline. The green domain in Figure 2.29 depicts a timeline of published IETF RFC and Internet Draft documents related to the NAT-PT transition mechanism. The first Internet Draft was already published in 1997. It has inspired many new Internet Draft mechanisms which were intended to somehow analyse and improve upon its shortcomings. However, the only direct successor to NAT-PT RFC is the RFC, which deprecates it [29].

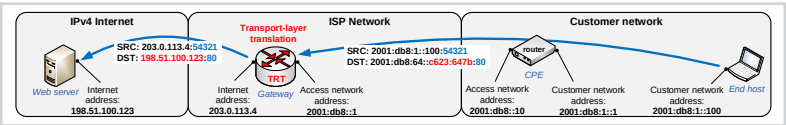
Penetration. NAT-PT was implemented by major network equipment vendors like Cisco and Juniper. It has also been implemented by researchers [123, 124] and as open source software [125]. It may be expected that its deployment will remain very limited due to its deprecation.

Operation. Figure 2.30 demonstrates a scenario where IPv6 traffic that comes from an end-host is statefully translated to IPv4 traffic through the appropriate replacement of IP headers. Multiple IPv6 hosts can be multiplexed on one IPv4 address. DNS-ALG is used to translate DNS requests on-the-fly. The CPE only performs the functions of an IPv6 router. A site-specific prefix 2001:db8:64::/96 is used for NAT-PT in this example.

Miscellanea. NAT-PT was deprecated for numerous reasons. Firstly, one of the missions of NAT-PT was also to solve the NAT46 problem, i.e., to enable IPv6 Internet hosts to access IPv4-only servers. Obviously, it is very difficult to map 128-bit address space to 32-bits. Secondly, NAT-PT has the DNS-ALG element tightly coupled with its IP translator, which means that for every DNS request made by the client, a corresponding entry was made in the translator's binding table. Furthermore, NAT-PT must be located in forwarding path, i.e., it must intercept all IPv4 and IPv6 traffic. Additional reasons for its deprecation are stated in RFC4966 [29].

Figure 2.31

In the TRT transition mechanism, IPv6 traffic is translated to IPv4 traffic by a TRT translator using transport-layer translation. DNS-ALG is also used to translate DNS requests.



TRT

Overview. Transport Relay Translator (TRT) [126] is a stateful transport-layer translator. It does not translate and forward IP packets but rather, terminates the connections with both communicating hosts. It allows IPv6-only hosts to exchange TCP and UDP packets with IPv4-only hosts. When a packet is received, the IP header is discarded and the TCP or UDP header is translated and then encapsulated in a new IP header. IPv4 addresses are embedded in IPv6 addresses and a /64 prefix is reserved for TRT operation. In order to be useful, TRT must be implemented with a DNS ALG [127], which translates AAAA queries into A queries.

IETF Timeline. The blue domain in Figure 2.32 shows a timeline of published IETF RFC and Internet Draft documents related to the TRT transition mechanism. There is only one Internet Draft and RFC published regarding TRT [126].

Penetration. TRT implementations are outdated and its deployments are rare. WIDE KAME [128] IPv6 stack implements TRT for TCP, also known as FAITH. BSD operating systems support `faithd` [129] impementation of TRT. Another implementation of TRT is pTRTd [130]. DNS ALG implementations also exist [131].

Operation. Figure 2.31 shows a typical TRT scenario. Let 2001:db8:64::/64 be a prefix, reserved for TRT operation, i.e., for IPv4 address mapping. This prefix is routed to the TRT translator. An IPv6-only end-host attempts to connect to the IPv4-only host at 198.51.100.123, which means it sends a packet to 2001:db8:64::c623:647b. TRT intercepts the packet, terminates the IPv6 connection and establishes a new IPv4 connection to the IPv4 address, which is contained in the lower 32 bits of the IPv6 address.

Miscellanea. The benefits of TRT are that end-hosts do not need to be modified in any way.

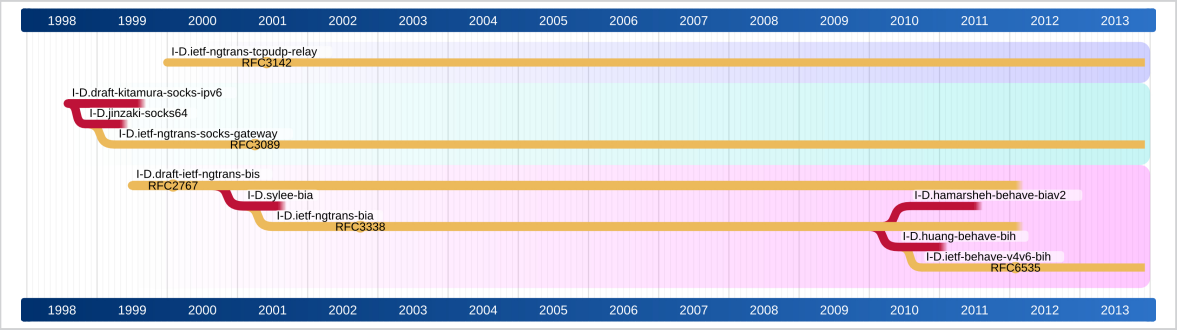
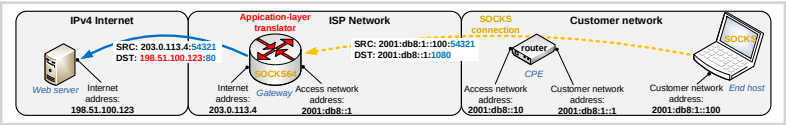


Figure 2.32

Timelines of published IETF RFC and Internet Draft documents related to TRT (blue), SOCKS64 (green) and BIH (red) mechanisms.

Figure 2.33

In SOCKS64 transition mechanism, IPv6 traffic is translated to IPv4 traffic by a SOCKS64 server using application-layer translation. The end-host must have SOCKS library installed.



Also, there are no fragmentation issues (as with translation) or path MTU issues (as with tunneling). However, only connections initiated from IPv6-only end-hosts are possible. Furthermore, TRT's stateful nature introduces complexity and scaling issues. Also, ALGs are required for application-layer protocols, which are not NAPT-friendly.

SOCKS64

Overview. The idea of SOCKS-based IPv6/IPv4 Gateway Mechanism (SOCKS64) [132] is similar to TRT, but in this case, the issue is application-layer translation. SOCKS protocol has been available for IPv4 since 1992 and is a proven application-layer proxy protocol. SOCKS64 allows an IPv6-only and an IPv4-only host to communicate with each other. The initiating host (client) needs to be "socksified", which means that a SOCKS library must be installed. Applications themselves, do not, however, require modifications.

IETF Timeline. The green domain in Figure 2.32 depicts a timeline of published IETF RFC and Internet Draft documents related to the SOCKS64 transition mechanism. There is only one RFC published regarding SOCKS64 [132].

Penetration. Apart from a few prototype implementations that are mentioned in SOCKS64's RFC document (NEC, Fujitsu), no other implementations appear to exist. Consequently, the deployment is likely very low or even non-existent today.

Operation. Figure 2.33 shows an application running on an end-host that initiates a connection to an external host using its DNS name. The client's SOCKS64 library intercepts the DNS request and initiates an authenticated TCP connection to the SOCKS server on port 1080. The SOCKS server returns a synthetic IPv6 address to the client and initiates an IPv4 connection with the remote host. From then on, it

works as a relay between the client and the remote host. Between the client and the server, the packets are sent over the "socksified" connection.

Miscellanea. SOCKS64 also provides some security as the client might first need to authenticate to the server. The greatest disadvantage of SOCKS64 is that it requires a SOCKS library by the client. Another significant disadvantage is also that it only supports client-initiated connections.

BIH

Overview. The motivation for the Bump-In-the-Host (BIH) [133] mechanism is to make it possible for IPv4-only applications that run on IPv6-enabled hosts to communicate to IPv6-only hosts. BIH is not intended as a generic IPv6 transition tool. The assumption is that many applications will never become IPv6-aware, e.g., custom-made enterprise networking applications. These applications perform DNS resolution, they adopt the client-server communication model and do not place IP addresses in the payloads. Initially, Bump-In-the-Stack (BIS) [134] and Bump-In-the-Application (BIA) [135] protocols were published by IETF to solve the same problem addressed by BIH. BIH merges the two and updates them into one Standards Track RFC document. Consequently, BIH has two implementation options: application-layer one (BIA) and network-layer one (BIS), the former of which is recommended by IETF. The network-layer approach translates IP headers of packets from IPv4 to IPv6 using the mechanism defined in SIIT [117]. The application-layer approach intercepts packets sooner, which means it does not execute any protocol translation, neither does it interfere with the DNS.

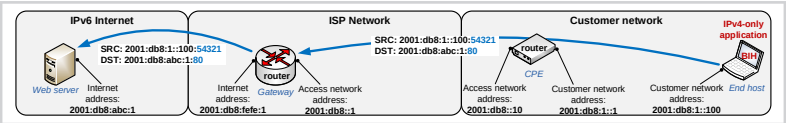
IETF Timeline. The red domain in Figure 2.32 shows a timeline of published IETF RFC and Internet Draft documents related to BIS, BIA and BIH transition mechanisms. For each of the mechanisms, one RFC document was published [133–135].

Penetration. As of yet, no implementations of BIH are publicly available. In the past, an implementation of BIA was available by i2soft Corporation, as mentioned in the BIA RFC document. However, this no longer seems to be available.

Operation. BIH is composed of several different components which must be implemented in the end-host [136, 137]:

Figure 2.34

In BIH transition mechanism, IPv4 requests are transformed into IPv6 requests in the end-host itself. No IPv4 packets are transferred on the wire.



- *Function Mapper* (BIA only): intercepts IPv4 socket calls and instead uses IPv6 to establish communication with the remote host.
- *Extension Name Resolver*: creates synthetic IPv4 addresses representing IPv6 destinations. However, if available, the destination's IPv4 address should be used if available. Also, it catches reverse DNS lookup queries done for synthetic IPv4 addresses.
- *Address Mapper*: manages local IPv4 address allocation (RFC1918) and manages mappings between locally generated or true IPv4 addresses and IPv6 addresses.
- *Protocol Translator* (BIS only): performs stateless IPv4-to-IPv6 translation (SIIT).

Figure 2.34 shows a scenario with an end-host with an IPv4-only application and BIH enabled. BIH tranlslates IPv4 requests of the application on-the-fly, so that synthetic IPv6 packets are sent to the network.

Miscellanea. Only specific and tailor-made solutions are envisioned for BIH, which will solve the specific problems of a limited number of environments. The main reason is that end-host must be upgraded with BIH.

2.3 Related Work

The area of IPv6 transition mechanisms is not a new research field. Soon after the definition of IPv6 protocol by IETF [12], researchers along with engineers, began examining various transition mechanisms in order to best resolve the problematic of transitioning the Internet from IPv4 to IPv6. The work in this area may be classified into four categories: new mechanism proposals, performance evaluations and comparisons, reviews and analyses. In the following section, we present the landscape of related work by discussing conference and journal papers that have been published for each of these categories.

2.3.1 New Mechanism Proposals

In order to propose a new transition mechanism that is to be considered by a wide spectrum of computer networking professionals, simply writing a journal or conference paper is usually insufficient. One must also compose an Internet Draft and submit it to the IETF. After that, it normally takes a few years and a lot of work to transform the initial Internet Draft into a Proposed Standard RFC. In this section, we review academic publications that have been written on the subject of transition mechanisms, some of which also introduce ideas that were published as papers.

In one of the first papers to address IPv6 transition mechanisms, Afifi et al. [138] present a general model for IPv6 transition, review the dual stack and translation (SIIT, NAT-PT and AIIH) models and propose a new IPv4-in-IPv6 tunneling mechanism – the Dynamic Tunneling Interface. The main idea presented is to leverage IPv4-in-IPv6 tunneling by using DNS service to obtain tunnel end-point addresses. However, the only Internet Draft [139] that was also published within IETF about DTI has expired and no further effort in this direction has subsequently been made by the authors.

AlJa'freh et al. [140] propose a new translation scheme that leverages a separate mapping between two assigned IPv4 and IPv6 public addresses with the actual IPv4 and IPv6 source and destination addresses for each session rather than using the actual IPv4 addresses and IPv4-mapped IPv6 addresses, as was proposed by Xing et al. [117] and Tsirtsis et al. [28]. However, this scheme introduces even greater complexity than the obsolete NAT-PT because it relies on a V4-V6 enabled gateway, coupled with three kinds of DNS servers: DNS4, DNS6 and DNS46. The proposal also fails to take into account the importance of being economical with IP addresses. Finally, the mechanism has not been published as an Internet Draft within IETF.

Another translation scheme, GATEVI, has been proposed by Xia et al. [141]. Compared with NAT-PT, this scheme does not require a stateful gateway but rather, enables stateless operation, which increases efficiency while decreasing complexity. It addresses the IPv4-to-IPv6 connection scenarios that are not considered by the replacement technique for NAT-PT, Stateful NAT64 [30]. Given that introducing IPv6 into IPv4-only networks is something that should be done today anyway (as opposed to "fixing" IPv4-only networks to enable initiating connections to IPv6-only networks) the IPv4-to-IPv6 scenarios have not attracted a lot of attention within IETF. GATEVI also does not support IPv4 address sharing, which is another desired feature in many modern

networks. Finally, GATEVI was never published as an Internet Draft within IETF.

Wu et al. [142] propose a method of locating the appropriate spot on the network to perform translation by leveraging tunneling, if necessary. This way, connection speed, operation complexity and scalability issues are avoided. Based on this method, PET (Prefixing, Encapsulation and Translation), a framework that integrates tunneling, translation and prefix-related control plan operations for IPv4-IPv6 interconnection, is proposed. This is intended to be a generic framework for IPv4 to IPv6 transition. The dynamic placement of the translation function is a novel concept. However, the authors do not demonstrate as to how this framework resolves the increasing need for IPv4 address sharing and their explanation lacks operational details, e.g., detailed packet flows for different communication models. Also, the authors do not seem to demonstrate why the placement of various transition methods is a larger problem than the inherent issues of these methods themselves (MTU issues, packet inspection issues, etc.). Finally, PET has never been proposed within IETF, which makes it even less operationally relevant.

Shi et al. [143] describe a new address and protocol translation transition mechanism based on NAT-PT [28]. Although the authors state that they have created a novel mechanism and extended the protocol translation algorithm SIIT [117], it remains unclear as to how exactly their proposal differs from NAT-PT. Also, their work does not seem to be published within IETF. Considering the fact that NAT-PT has been declared obsolete by IETF [29], their proposal, being based on NAT-PT, demonstrates the same issues as does NAT-PT.

Park et al. [144] propose a modified DSTM [55] mechanism, which allows for IPv4-only hosts to establish connections with DSTM hosts. They also provide a performance evaluation of an implementation in the *ns-2* network simulator, where they measure the transmission delay and response time of DNS queries. However, as DSTM has itself been abandoned by IETF, (probably as a result of its complexity and the requirement to change end-hosts' stacks) their proposal is not operationally relevant. Nor was this concept ever published within IETF as an Internet Draft.

Wang et al. [145] propose DTTS, a mechanism that employs the dual stack approach with the dynamic tunneling technique and which is similar to DSTM [55]. Accordingly, the end-hosts' network stacks must be modified to support DTTS, which is a very unrealistic requirement. The authors lay down the DTTS architecture and present deployment scenarios. However, their mechanism has not been proposed to the IETF

in the form of an Internet Draft.

2.3.2 Implementations

For the sake of completeness, we also discuss academic publications written on the subject of implementations of IPv6 transition mechanisms, even though these are not directly relevant to this thesis. The general problem with implementations of transition mechanisms as research tools is that they do not provide the necessary basis for fair comparison of the mechanisms, as it is impossible to separate measuring an implementation from measuring a mechanism. This is the main reason why we propose a theoretical performance analysis and comparison framework in Chapter 5.

Zhao et al. implement [146] a variant of Class 5 (Section 3.1) IPv4 address sharing mechanism. They provide their own address format and port allocation algorithm. Furthermore, they present technical details of CPE and gateway Linux implementations using a TUN driver for tunneling.

Hou et al. [122, 147] present an implementation of the IVI mechanism using OpenVPN open source tunneling software, without actually implementing address format and port allocation algorithms. The implementation is more proof of concept than an ISP-grade implementation for testing in real-world networks.

Tahir et al. [148] tend to describe a high-level implementation of the DSTM [55] in their 6iNet test-bed. However, they only provide a few implementation details as their paper predominantly discusses the 6iNet research network, other transition mechanism approaches (tunneling, dual stack, translation) and a description of the DSTM mechanism.

AlJa'afreh et al. [149], who authored the proposal paper for the BDMS mechanism [140], have implemented BDMS in OMNeT++ simulator. They also provide a performance evaluation, where they measure Round Trip Time (RTT), End-to-End Delay, Packet Dropping and Queuing Total Delay. However, as the evaluation does not take into account any other (similar) mechanism, it fails to offer any comparison of BDMS with competent solutions.

2.3.3 Performance Evaluations and Comparisons

Evaluating performance of IPv6 transition mechanisms is a complicated task because it is difficult to draw conclusions with respect to transition mechanisms by testing their (often very heterogenous) implementations.

Raicu et al. [150] measured the effects of two IPv6-in-IPv4 tunneling scenarios. In the first measurement, the tunnel was established between the routers, while in the second, the tunnel end-points were the end-hosts themselves. Raicu et al. compare the measurements of various metrics (throughput, latency, CPU utilization and TCP connection time) to the IPv6-only baseline. They report an insignificant effect of router-to-router tunneling and as much as a 66 % increase in CPU utilization at the end-hosts when end-to-end tunneling. This is a good example of how a different implementation of the same packet operation (encapsulation and decapsulation) may be implemented very differently. Effectively, this means that the authors of this paper did not really measure the characteristics of the mechanisms but rather, they assessed the quality of implementations.

AlJa'afreh et al. [151, 152] test and compare BDMS and DSTM transition mechanisms. They avoid much of the implementation-induced bias by using OMNeT++ simulator as the implementation platform. They seem to choose metrics carefully enough (one-way delay, round-trip delay, throughput) so that BDMS dominates over DSTM mechanism. It is nevertheless unclear as to what is the added value of having two delay-based metrics that essentially show the same result and why the authors have not considered qualitative metrics like application-layer compatibility, which would certainly be in favor of the DSTM mechanism. Such a comparison could also be performed using the performance analysis framework proposed in this thesis in a more well-defined and implementation-independent method.

Similarly, Hanumanthappa et al. [153] evaluate and compare a variant of SIIT dubbed BD-SIIT (Bi-Directional SIIT) and DSTM by implementing them in a *ns-2* simulator. The BD-SIIT mechanism described in this paper is generally very reminiscent of the BDMS vs. DSTM comparison [151] discussed above.

Narayan et al. [154] and Chang et al. [155] also authored papers on the performance evaluation of IPv6 transition mechanisms (6to4, configured tunnel and tunnel broker). They consider the tested implementations as legitimate representatives of the mechanisms, which is precisely what should be overcome. Because of this, we believe their results do not demonstrate the real relationships between these mechanisms.

2.3.4 Reviews

One of the main motivators for the systematic review of transition mechanism space is that thus far, many survey papers on the subject have been partial – having either

focused on one of the transition mechanism classes, e.g., translation, or having covered only a part of mechanism space by using actual mechanisms in the reviews instead of their abstractions (classes in this thesis).

In this manner, Govil et al. [156] review the following transition mechanisms: dual-stack, NAT-PT and NAPT-PT, SIIT, DSTM, general IPv4-over-IPv6 tunneling, configured tunneling, automatic tunneling, ISATAP, 6to4, 6over4, BIS, BIA, TRT, and SOCKS64. Jayanthi et al. [157] provide a very similar review of IPv6 deployment mechanisms. In addition to yet another review of IPv6 deployment mechanisms, Tatipamula et al. [158] also focus on deploying IPv6 over dedicated data links and MPLS backbones. Geer [159] briefly describes more recent mechanism proposals also covering IPv4 address sharing area (DS-Lite, IVI, NAT64).

To summarize, all of the authors above discuss various transition mechanisms along with some of their features and properties. Mostly, they are focusing on IPv6 deployment mechanisms, while IPv4 address sharing mechanisms are not discussed. Also, these analyses are not systematic given that they do not decouple properties from the mechanisms and attribute the features (both positive and negative) to the properties as opposed to the mechanisms.

Mackay et al. [160] also review dual-stack, 6to4, tunnel broker, IPv6 over ATM/MPLS, ISATAP, Teredo, SIIT, NAT-PT, BIS/BIA, TRT, and SOCKS64 mechanisms and then very briefly discuss various aspects like security, performance, functionality, node requirements, address requirements, application requirements, ease of use, and ease of management. Finally, they describe the use of different transition mechanisms in three different scenarios: Internet service providers, enterprise networks and unmanaged networks.

Che et al. [161] first provide a short introduction into IPv6 and a discussion about the status of the IPv6 deployment in April 2009. They also list and discuss the drivers and the inhibitors for the IPv6 transition and a comparison of IPv6 and NAT as the IPv4 address exhaustion solutions. Furthermore, they outline a comparison of different transition approaches (dual-stack, translation, tunneling, and IPv6 over WAN links) and different tunneling techniques (6to4, 6over4, Teredo, and ISATAP). In a performance evaluation, the authors measure average TCP segment delay and IP packet loss in four different scenarios using OPNET Modeler. Finally, they describe the migration challenges and solutions including interoperability, education, planning, and business issues.

In a review of recent NAT standardization efforts, Wing [162] discussed address sharing mechanisms and provided some insight into the consequences of using ISP-level address sharing. He described Stateful NAT64 and DS-Lite and highlighted their advantages and disadvantages. However, as the emphasis of his review is not on address sharing, he did not offer a structured classification, nor did he provide a systematic analysis of the tradeoffs involved.

Waddington et al. [163] provided one of the first attempts to identify the common properties of multiple mechanisms which yield a limited classification of transition mechanisms. However, actual mechanisms are still considered instead of their abstractions and the paper, being more than a decade old, lacks many of the new concepts and mechanisms that have since then been developed.

Wu et al. [164] and Cui et al. [165] describe a branch of transition mechanisms referred to as *softwires*, describing the *4over6* or IPv4-over-IPv6 architecture of transition mechanisms that have recently been used by many of the IPv4 address sharing mechanisms.

Huston published one of the first reviews of IPv4 address sharing mechanisms [166], in which he presented CGN (NAT444, DS-Lite) and A+P approaches. He described their operation as well as some of their most significant advantages and disadvantages. We extend this work by presenting a mechanism classification system, and by systematically analysing tradeoffs and including newer address sharing mechanism proposals.

2.3.5 *Analyses*

In this section, we review publications that have gone one step further from merely reviewing various transition mechanisms and which are also more closely related to the most important scientific contributions presented in this thesis.

Ripke et al. [167] study the impact of port-based address sharing on residential broadband access networks. Specifically, they are interested in the behaviour of the end-hosts regarding the use of transport-layer identifiers, i.e., mostly UDP and TCP ports. They provide an analysis of considerations for address sharing, including NAT traversal, state TIME-WAIT issues and port assignment strategies. By observing the number of used ports over time, they try answering the question as to how many ports customers should be assigned in A+P (Address Plus Port) schemes. Finally, they present an analysis of port consumption dependence on the TIME-WAIT period in TCP.

Cui et al. [168, 169] review tunneling-based transition mechanisms for ISP back-

bone and access networks. Most importantly, they analyse the following dimensions of tunneling mechanisms: state maintenance (per-user, per-session, stateless), IPv4 address multiplexing (no multiplexing, network address translation, port-range), IPv4 address allocation (address allocation to user side, address management in a CGN) and IPv4/IPv6 address coupling. However, they do not provide a methodology for their classification, nor do they cover the whole design space of transition mechanisms.

Bush et al. [35] present their vision of transition to IPv6, which also includes IPv4 address sharing mechanisms. They warn about the consequences of deploying inappropriate mechanisms, which would result in an Internet that is very different from the one we are familiar with today. Furthermore, they emphasize the importance of avoiding CGNs, which make core networks too complex to allow for the easy deployment of future services. Also, in their experience, it is incorrect to expect that deploying more IPv4 "life support" devices will assist in the transition. In their experience, it will only further delay it. They present a 2-D space of transition mechanisms, with the first dimension being the amount of stored state and the second being the type of transition (either v4-over-v6 or v6-over-v4). In contrast, we focus on IPv4 address sharing mechanisms, not on IPv6 transition mechanisms in general.

At IETF 80, Xie et al. [170] presented a comparison of address sharing mechanisms. Due to the use of vague terminology, it is unclear as to which mechanisms are considered. They do not offer justification for some of the claims that are made (e.g., how customer hosts using the NAT444 could be reachable from the Internet). Their comparison could benefit significantly from the classification system proposed in this thesis.

A typical example of research that could significantly benefit from a concise classification of IPv4 address sharing mechanisms was presented by Deng et al. [171]. They analyze DS-Lite, Stateful NAT64 and "AplusP" proposals, but such classification is overly simplified. For example, only for "AplusP" can we identify a whole range of various approaches, yet Deng et al. only test an implementation of one of the variants.

Conversely, Wu et al. [172] present a survey of mainstream translation and tunneling mechanisms, along with technical details, advantages and disadvantages. Being the most recent paper on the subject, it successfully captures all the latest transition techniques that have been proposed within IETF. Wu et al. also provide recommendations on appropriate mechanism selection for different scenarios. However, they still operate with actual proposed mechanisms, which we argue is suboptimal, given that those

proposals tend to occasionally change (for example, MAP-E is still an Internet Draft). Wu et al. also fail to provide a classification of mechanisms, which would abstract less operationally important details. This means they do not demonstrate the causal relationships between specific mechanism properties and design choices.

3

IPv4 Address Sharing Mechanisms

Some ISPs do not have enough IPv4 addresses to provide a dedicated IPv4 address to each customer. To support continued growth, individual IPv4 addresses will have to be shared between multiple customers, which we refer to as "ISP-level address sharing". However, the consequences of deployment of these mechanisms for the Internet users is not well understood.

Unfortunately, ISPs often do not have enough information about the potential consequences of their decisions. E.g.:

- Would the deployment of a double network address translation (NAT) mechanism prevent Xbox LIVE customers who share the same IP address from playing games online?
- Will cyber-criminals be untraceable, because content providers (today) only log time and IP address of the attacker, but not source ports?
- Does the deployment of a particular mechanism create provider lock-in, so that the customers have to use, say, the Internet TV service of their ISP as competitors' services fail to work?
- Will the End-to-End Principle, which is one of the core principles of the Internet [173], become even more endangered with ISP-level address sharing?
- Will all new protocols have to tunnel over HTTP as this may be the only remaining application-layer protocol that traverses address sharing devices?

We present a systematic approach to classifying and analyzing existing IPv4 address sharing mechanisms. To compare and to understand them, we abstract some of their details and explore the whole solution space. First, we define the classification dimensions and properties. Then, we infer nine classes categorizing existing mechanisms. Similar mechanisms are classified into the same class.

Our main research objective is to propose a classification for IPv4 address sharing mechanisms. We feel the need for such classification is significant: revealing gaps and conflicts, while new IETF Internet Drafts of address sharing proposals keep coming, many of them expiring after a year or two. Other networking research papers focus on these drafts arbitrarily, while they could focus on whole classes instead and thus gain more universal value. Additional classes might be defined in the future. Further, our results will inform the design of new address sharing mechanisms.

We consider networks where address sharing must be used, including broadband ISPs providing Internet access to large numbers of customers. However, this excludes mobile ISPs even though their number of subscribers have long surpassed the number of wireline subscriptions. We believe ISP-level address sharing will have stronger impact on wireline users than on mobile ones, as non-mobile end-hosts usually have higher requirements, e.g., peer-to-peer networking, than mobile end-hosts. Moreover, it is a common practice for mobile users to use WiFi networks when available, so their device becomes another end-host in our topology. We only consider unicast; multicast is out of scope, as it is not related to address sharing. We use the terms *port* and *flow* in the context of transport-layer protocols like TCP and UDP.

We make some assumptions about the networking topology. First, we do not consider address sharing mechanisms where end-host modification is required; this is a realistic requirement as it is infeasible to change deployed hosts, e.g., it was years after the IPv6 RFC was published until Windows XP had production-ready IPv6 support. Second, every customer is assumed to have a CPE, even though some mechanisms allow for connecting end-host directly to the access network. CPE, however, may be modified or replaced for the purpose of deploying some mechanism.

CPEs can share a public IPv4 address in two ways:

1. *Carrier-Grade NAT* (CGN): using a translator with a Network Address and Port Translation (NAPT) function located in the core of the ISP's network, which multiplexes multiple CPEs on a single IPv4 address.

2. *Address-Plus-Port* (A+P) [174]: by communicating in a port-restricted manner, where bits from the port field are used to extend the IPv4 address field, i.e., choosing a source port for outgoing packets from a subset of the whole 16-bit port range and receiving incoming packets destined to a port from the same subset. For this to work, the CPEs have to port-restrict outgoing packets and the gateways have to route incoming packets using the destination port.

We only consider A+P CPEs and not A+P end-hosts. An A+P CPE therefore must perform a (port restricted) NAPT function to support multiple end-hosts.

In the time of dial-up Internet access, IP addresses were shared over time using Dynamic Host Configuration Protocol (DHCP) [175] or other provisioning protocols. Today, when broadband always-on access is ubiquitous, sharing addresses over time is not considered an effective ISP-level address sharing method.

A basic building block of many IPv4 address sharing mechanisms, NAPT44 (NAPT from IPv4 to IPv4, also known as Traditional NAT [20]) has been deployed for a long time in home networks and enterprises, but is not (yet?) commonly deployed within ISP core networks [50].

NAPT44 uses transport-layer identifiers (usually TCP and UDP ports) to multiplex privately addressed [78] hosts to a public IPv4 address. Using NAPT44, source addresses (and possibly ports) are translated as the packets are transported from an end-host behind a NAPT to the Internet, and destination address (and possibly port) translation is performed in the reverse direction. When attempting to initiate a flow toward a machine behind the NAPT device, the packet is sent from the Internet host to a specific address and port of the NAPT device, which appears to be the final destination. The destination address (and possibly port) are translated so that the packet is forwarded to the appropriate host (usually using RFC 1918 addressing).

NAPT44 in customer networks is understood [162, 176] although it is well known that different translators behave differently [177], even though IETF has some efforts to standardize behavior [178, 179]. The Session Traversal Utilities for NAT (STUN) protocol was developed to enable discovery of the presence and behavior of translators [180]. However, in ISP-level address sharing, several unforeseen technical issues arise. For example, as the NAPT44 function must reside in the ISP's core network, to address all the customers' CPEs we have to use a sufficiently large block of private (or special purpose [181]) IPv4 addresses in the access network. If this block is a private address block [78], there will be issues with overlapping address space [182]. As an NAPT44 translator is stateful, the size of its mapping table increases with the number of customers [183].

In this chapter, we first propose a classification for IPv4 address sharing mechanisms. First, we discuss the difference between the inherent issues of any ISP-level address sharing and the issues related to properties of specific mechanisms and the methodology. We propose five classification dimensions, and we classify the existing proposals into the classes (Section 3.1). Next, we analyze the properties of the mechanisms along the proposed dimensions (Section 3.2). Finally, we discuss the deployment tradeoffs, that we identified (Section 3.4).

3.1 Classification

An examination of the design space instead of individual mechanisms allows us to determine the benefits and disadvantages of each mechanism and to see what research needs to be done to conceive new useful approaches. We are interested in features such as state storage resource required, IPv6 encouragement, and requirements on the access network.

3.1.1 Inherent ISP-Level Address Sharing Issues

It is important to understand the difference between the inherent issues of any ISP-level address sharing and the issues related to properties of specific mechanisms. In this thesis, we only discuss the latter. Ford et al. [184] have analyzed the potential issues of IPv4 address sharing. Here, we only summarize issues introduced by ISP-level address sharing that are common to all the mechanisms discussed in this thesis.

- *Variable port requirement dynamics:* The total number of customers able to share an IPv4 address will depend upon assumptions about each customer's average number of ports in use, and the average number of simultaneously active customers.
- *Connection to a well-known port number:* Inbound connections will not work in the general case.
- *Limited to TCP, UDP, and ICMP:* All address sharing mechanisms are limited to TCP, UDP, and ICMP, thereby preventing customers from fully utilizing other transport-layer protocols of the Internet (e.g., SCTP).
- *MTU Packet Too Big attack:* A malevolent user could send an ICMP "Packet Too Big" (Type 3, Code 4) message indicating a next-hop maximum transmission unit (MTU) of anything down to 68 octets. This value will be cached by the off-net server for all customers sharing the address of the malevolent user. This could lead to a denial of service.
- *Traceability:* As an IPv4 address is no longer a unique identifier, tracing particular customers is challenging.

- *Reverse DNS:* Many service providers populate forward and reverse DNS zones for the public IPv4 addresses that they allocate to their customers. Where public addresses are shared across multiple customers, such strings are no longer sufficient to identify individual customers.

- *6to4 incompatibility:* The 6to4 transition mechanism requires a publicly routable IPv4 address to function.

3.1.2 Classification Methodology

The methodology for determining the five dimensions is as follows.

- *Mechanism analysis*: Examine all existing mechanisms and extract their properties from the IETF RFC and Internet Draft documents.

- *Form candidate dimensions*: Group properties that describe the same aspects of mechanisms together, e.g., one mechanism might require state storage in the gateway, another may not. These are two properties of the same dimension (state storage).

- *Remove specifics*: Ignore those candidate dimensions for which at least one existing mechanism yields "Non Applicable", e.g., the address format and port-set allocation algorithms of stateless A+P mechanisms [185] is not applicable in other mechanisms, where there is no address format at all.

- *Assure unique clustering*: Where two candidate dimensions yield equal clusterings of existing mechanisms, choose one using operational relevance. If there are important issues with one or more properties of the left-out candidate dimension, we still discuss them.

- *Remove less relevant dimension candidates*: The final set of dimensions is refined by removing dimensions containing operationally unimportant properties. This is the most subjective step. After mechanism analysis, identify important issues that were explicitly stated as such in the documents or were given as a motivation for defining one or more mechanisms. As a final check, make sure that removal one of the candidate dimensions would not lead to such an important issue being ignored.

We have discussed the methodology, the resulting dimensions and properties with many operators, fellow mechanism designers and practitioners in general at various IETF meetings. We believe that the fact, that they found our work valuable, provides an important validation for the correctness and value of the classification as a whole.

Mechanism Analysis

We consider the following mechanisms to develop the classification:

- Gateway-Initiated Dual-Stack Lite Deployment [186],
- Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion [187],
- The Address plus Port (A+P) Approach to the IPv4 Address Shortage [174],
- Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers [30],
- DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers [118],
- The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition [122],
- Stateless Automatic IPv4 over IPv6 Tunneling with IPv4 Address Sharing [188],
- NAT444 [208],
- IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd) [190],
- Mapping of Address and Port with Encapsulation (MAP) [211],
- IPv4 Residual Deployment on IPv6 infrastructure – protocol specification [212],
- Stateless 4over6 in access network [213],
- A Lightweight AplusP Approach for Public IPv4 Address Sharing in IPv6 Environments [210],
- Lightweight 4over6: An Extension to the DS-Lite Architecture [216],

- NAT offload extension to Dual-Stack lite [217],
- Stateless DS-Lite [218],
- dIVI: Dual-Stateless IPv4/IPv6 Translation [219],
- Mapping of Address and Port using Translation (MAP-T) [220],
- dIVI-pd: Dual-Stateless IPv4/IPv6 Translation with Prefix Delegation [221],
- 4via6 Stateless Translation [222],
- 464XLAT: Combination of Stateful and Stateless Translation [223].

By analyzing the mechanisms above, we extract the following properties (in alphabetical order):

- flow state stored in the gateway for every session,
- flow state stored in the gateway for every subscriber IPv4 address or port allocation,
- for provisioning of additional IPv4 addresses and port ranges, additional signalling is required,
- for provisioning of additional IPv4 addresses and port ranges, additional signalling and CPE/subscriber network readdressing is required,
- hub-and-spoke topology between subscribers within the same ISP,
- IPv4 address and port (range) are embedded in the IPv6 address,
- IPv4 address and port (range) are not embedded in the IPv6 address,
- IPv4 address and UDP/TCP port allocation is dynamic,
- IPv4 address and UDP/TCP port allocation is not specified,
- IPv4 address and UDP/TCP port allocation is static,
- IPv6 address format follows RFC 6052 [119],

- IPv6 address format is arbitrary,
- IPv6 address format is of type “4rd”,
- mesh IPv4 topology between subscribers within the same ISP (direct CPE-CPE paths possible),
- NAPT function is mandatory somewhere in the mechanism architecture,
- NAPT function is not mandatory anywhere in the mechanism
- NAPT function located in CPE and Gateway,
- NAPT function located only in the Gateway,
- NAPT function located only in the CPE,
- NAPT function type is NAPT44,
- NAPT function type is NAPT64, architecture,
- no flow state stored in the gateway,
- no special packet treatment in the access network (routed packets),
- no special requirements for provisioning of additional IPv4 addresses and port ranges,
- no topology between the subscribers within the same ISP,
- point-to-multipoint tunneling in the access network using IPv4-in-IPv6 tunnels,
- point-to-point tunneling in the access network using IPv4-in-IPv6 tunnels,
- port-set calculation algorithm is not specified,
- port-set calculation algorithm is of type “sequential”,
- port-set calculation algorithm is of type “modulo”.

- provisioning of additional IPv4 addresses and port ranges can be performed automatically,
- provisioning of additional IPv4 addresses and port ranges can only be performed on request,
- stateless NAT46 translation and stateless NAT64 translation performed on packets when traversing the access network,
- there is no IPv6 address format,
- there is no port-set calculation algorithm needed,

Based on the list of properties above, we can understand that a rigid classification is needed in order to perform further investigation and analysis of IPv4 address sharing mechanisms. But most importantly, it becomes clear that a certain level of abstraction is required in order to be able to compare the mechanisms and to develop the performance evaluation framework, described in Chapter 5.

Form Candidate Dimensions

Next, by analyzing the dependencies between the properties above and by grouping the extracted properties together, we develop the first list of candidate dimensions for the classification (in alphabetical order):

- IPv4 address and port allocation policy,
- IPv4 topology,
- IPv6 address format,
- level of IPv6 requirement,
- location of the IP address sharing function,
- NAPT type,
- port-set calculation algorithm,
- requirement of IPv4 address and port (range) embedded in the IPv6 address,

- requirement of NAPT,
- requirements for provisioning of additional IPv4 addresses and port ranges,
- state storage in the gateway,
- traversal method through the access network.

After this step, there are 12 candidate dimensions. Some of those are strongly dependent on others, which means they are not very informative. With these candidates, we move on to the next step of the methodology.

Remove Specifics

Some of the 12 input dimensions in this step are not applicable to all the mechanisms so we have rule them out. For example, there are no port-set calculation algorithms and IPv6 address formats involved in the NAT444 mechanism. We end up with the following candidate dimensions:

- IPv4 address and port allocation policy,
- IPv4 topology,
- level of IPv6 requirement,
- location of the IP address sharing function,
- NAPT type,
- requirement of NAPT,
- state storage in the gateway,
- traversal method through the access network.

After this step, there are 8 candidate dimensions left. We proceed to the last two steps of the methodology to obtain the final set of dimensions of the classification.

Assure Unique Clustering

Some dimensions do not bring any added value into the classification because they are highly dependent on others. This means that removing one of two or more interdependent dimensions yields identical clustering. In this step, we identify the dimension "requirement of NAPT" and "NAPT placement" as such, as they divide the mechanisms into identical clusters like. Also, we rule out the dimension "NAPT type", given that it clashes with the "level of IPv6 requirement" dimension. It turns out that "NAPT64 type" of the translation function actually means "IPv6 required", so the dimension property issues are the same.

- IPv4 address and port allocation policy,
- IPv4 topology,
- level of IPv6 requirement,
- location of the IP address sharing function,
- state storage in the gateway,
- traversal method through the access network.

We proceed with the 6 remaining dimensions to the last step of the methodology.

Remove Less Relevant Dimension Candidates

In the last step, we remove those dimensions that are not operationally relevant. As noted before, this is the most subjective step in the methodology. We remove the dimension "IPv4 topology". Mesh topologies, compared to hub-and-spoke topologies, theoretically influence the resource usage on the Gateway, as CPE-CPE direct paths are possible. However, such setups are relatively difficult to establish and maintain, mostly because of the security issues. Also, intra-subscriber traffic is usually much lower in volume than subscriber-to-Internet traffic. This means that the benefit of such a setup is insignificant.

Finally, we have obtained the following final 5 dimensions of the classification, which are analyzed in detail in the following sections:

- IPv4 address and port allocation policy,

- level of IPv6 requirement,
- location of the IP address sharing function,
- state storage in the gateway,
- traversal method through the access network.

3.1.3 Classification Dimensions

The rest of the this chapter is aligned with the classification dimensions. However, as the classification is inferred from existing mechanisms, the coverage may not be complete. Nevertheless, in the following sections, we argue completeness of each individual dimension.

Dimension 1: Location of the IP Address Sharing Function

The IP address sharing function can be located either in the *CPE* (A+P mechanisms), in the *gateway*, or in the *CPE and gateway* (CGN mechanisms). In A+P case, the customer can choose between using a CPE with a port-restricted NAPT function to connect their hosts or connecting a single A+P-capable host directly to the access network. In the former case, the user is in control of the translation (e.g., port-forwarding). Where there is address sharing in the gateway (CGN), it becomes a critical function of the ISP, which in turn has to manage any gateway-located NAPT function.

This dimension is important as A+P mechanisms preserve the Internet's end-to-end principle to customer premises.

Given our assumed CPE-Gateway topology described earlier and our wish to support unmodified end-hosts, the address sharing function cannot be placed anywhere other than the CPE or the gateway.

Dimension 2: State Storage in the Gateway

State information in the gateway may need to be held *per flow*, *per allocation*, or it can be *stateless*. Note as stateful CPE devices have been widely deployed without major difficulties, this dimension only considers the gateway, which normally is supposed to hold state for a large number of customers.

From the multiple perspectives of performance, maintenance, scalability, cost, and complexity, one of the most desired properties of a mechanism is statelessness. The

process of packet traversal through the mechanism is as determined from the packet IP header [189].

Per-allocation (of port and/or address) stateful mechanisms require gateway devices store information mapping IPv4 addresses and port-sets to tunnel ID, IPv6 prefix, or CPE address.

Per-flow (UDP, TCP, and ICMP effectively) stateful mechanisms require one entry in the gateway state table per flow. As flows are short-lived, and each customer can establish many simultaneously; this state has a high churn rate.

This dimension is important because the volume of state to be stored influences the state synchronization, logging, processing, and storage requirements of the gateway. State storage will not be more fine-grained than per-flow. On the other hand, the granularity between per-flow and per-allocation can be arbitrary (and is equivalent when allocation includes only one port). The situation where a port-set is allocated among multiple customers within one allocation does not make sense as there is no way to know to which customer each packet should be forwarded.

Dimension 3: Traversal Method Through the Access Network

Here, we refer to means by which the payload of IPv4 packets is exchanged between the CPE and the gateway. We consider the method and extent to which packet header manipulations are required. We identify the following methods *routing*, *tunneling*, *double address family translation*, and *reversible header translation*.

Routing is the simplest traversal method. No packet header manipulation occurs, and therefore IPv4 and IPv6 packets can be carried from source to destination through native networks.

By tunneling we refer to any process of encapsulating, transporting, and decapsulating a packet—for example, wrapping IPv4 packets in an additional IP header, meaning original packets travel through a nonnative network intact.

Double address family translation leverages the Stateless IP/ICMP Translation Algorithm (from here on abbreviated as stateless NAT64) [117], which is a method of translating the IP header of a packet from IPv4 to IPv6, and vice versa. As the translation can be done algorithmically, it is useful as a means to transport the payload originally placed into an IPv4 header over an IPv6-only access network; then translating it back to IPv4 and forwarding it to the IPv4 Internet. In this case, we need to perform the address family translation twice—in the CPE (v4 to v6) and the gateway

(v6 to v4).

Reversible header translation can be seen as a special case of double address family translation, with most of the IPv4 header information preserved [190].

The tradeoffs when choosing a traversal method are significant and described in the Section 3.2 below. Completeness of this dimension is hard to argue as one can envision an improved tunneling or translation mechanism, which will introduce new issues for analysis. If a new method is later invented, this dimension must be extended.

Dimension 4: Level of IPv6 Requirement

Not all IPv4 address sharing mechanisms are IPv6 transition mechanisms. Some of them require IPv6 in one or more parts of the network, while others work fine without IPv6. The level of IPv6 requirement is directly related to the semantics of the translation function for address sharing. We can distinguish three cases for IPv6: *no IPv6 required*, *IPv6 partly required*, and *IPv6 required*.

The first case covers those mechanisms where IPv6 networking is independent of the address sharing mechanism and can optionally be provided using a traditional dual stack method. In most mechanisms, IPv6 is *partly required*, which means access network has to be IPv6-enabled for successful operation. Finally, some mechanisms require IPv6-enabled customer networks, which allows for IPv6-only ISP networks, where IPv4 is present solely at the Internet border.

When considering the widespread adoption of IPv6, it is important to evaluate to what extent a specific mechanism encourages, supports, utilizes, or requires IPv6 in the ISP.

There are three networks considered in the assumed topology: the IPv4 Internet, the access network, and the customer network. There are four possible combinations of IPv4 and IPv6 values for access and customer network. This dimension excludes the combination where IPv6 is required in the customer network and IPv6 is *not* required in the access network. Although such a mechanism could be envisioned in theory, it does not make sense as migrating customer networks to IPv6 is considered more challenging than migrating the access network as this is owned by the ISP.

Dimension 5: IPv4 Address and Port Allocation Policy

A mechanism either provides *static and dynamic* allocation or *static-only* allocation.

In *dynamic allocation*, a port and possibly also the shared IPv4 address are selected as required by the NAPT function. These are chosen on a per-flow basis as each new flow is established. The port number and address associations may be freed and reused as the flow times out. With *static allocation*, an IPv4 address and a port-set are reserved per allocation and are then (until possible reallocation) used by NAPT function for only one customer.

In CGN, static or dynamic allocation may be used. In A+P, only static allocation is possible as the address sharing function is located at the edge of the network. The CPE chooses ports for new flows from its preallocated set.

Note that, in this context, the terms static and dynamic describe the granularity and persistence of address and port allocations. They do not describe the state storage needed in the gateway. We use the terms *stateless* and *per-allocation* stateful for that (Dimension 2). Address and port allocation policy is important because it influences address sharing ratio, state storage in the gateway, and security. Both dynamic and static are the only viable options for allocation policy. The question whether or not a class of mechanism can provide *only* dynamic allocation is irrelevant, as the *static and dynamic* option denotes what policies a class of mechanism *can* support (as opposed to *must* support).

3.2 Detailed Property Analysis

To identify mechanism tradeoffs, we have to understand their properties. We discuss the issues of each property along the five dimensions of our classification. For each dimension and dimension property, we list and discuss all relevant operational issues.

3.2.1 Dimension 1: Location of the IP Address Sharing Function

If the address sharing function is located in the gateway, we call such mechanisms CGNs, otherwise we refer to them as A+P mechanisms. In this section, we discuss how this difference impacts support for end-to-end connectivity, gateway and CPE complexity, etc.

CPE and Gateway

Port forwarding through two levels of NAPT: End-to-end connectivity is difficult to achieve, as it is non-trivial for an end host to have ports forwarded to their CPE. Existing port mapping protocols, Universal Plug and Play (UPnP) [191] and NAT

Port Mapping Protocol (NAT-PMP) [192] do not support double NAPT. However, PCP [193] will support it according to the charter of IETF *pcp* working group. Unfortunately, in 2013 the IETF is still discussing PCP, and it is yet to be implemented by vendors. Also, it adds additional complexity into the address sharing mechanism. If port forwarding support is not provided, applications that rely on it (e.g., BitTorrent), will not function optimally [194].

Gateway

Limited control over the NAPT function: In CGN schemes, customers may not modify the NAPT44 function, e.g., adapt it to new protocols, since it is locked in the ISP's core. Installing and enabling new Application Layer Gateways (ALGs) for custom applications may invoke lawyers. Similarly, letting customers configure static port forwarding rules in the centralized NAPT44 function is impractical from an ISP's perspective and raises security considerations (denial of service, lack of authentication and confidentiality, off-path source spoofing, and other threats [193], section 18). Some CGN implementations may support dynamic request of port forwarding rules by using signaling protocols such as PCP [193], NAT-PMP [192], and UPnP [191, 195]. The latter are less adapted for CGN scenarios as the port reservation dialog may not be successful if most of the ports are already in use by other customers.

Higher gateway complexity: CGN gateways are more complex because they must store and synchronize a lot of flow state (see the "Stateful per flow" discussion in Section 3.2.2). It also concentrates failure points [196].

CPE

Only static IPv4 address and port-set allocation possible: A+P CPEs must be given an IPv4 address and port-set in advance, i.e., statically. If an end host does not have active flows, its ports are unused yet they cannot be used by another customer.

Higher CPE complexity: A CPE with NAPT is more complex. As today the NAPT function is ubiquitous, that in itself is not the main issue—the problem is assuring that A+P CPE is aware of its allocated IPv4 address and port-set. Especially in per-allocation stateful (Dimension 2) mechanisms, additional signaling is required. Also, A+P CPE's parameters must be synchronized with the gateway.

3.2.2 Dimension 2: State Storage in the Gateway

In this section, we discuss how the specific properties impact logging and high availability requirements, scalability, and address usage efficiency.

Per Flow

State synchronization: When gateways are clustered, either for high availability or load balancing, any state storage adds significantly to the complexity of the cluster [197]. All cluster nodes must synchronize state, which is hard when state is changing rapidly. The synchronization process must be very robust and fast. For example, if a customer establishes a TCP flow, its entry is stored in the gateways' state tables. This must be immediately synchronized with other nodes in order for them to match any subsequent packets from the customer to this specific flow. The problem of high resource usage (CPU, memory and intra-node communication) must also be addressed by carefully designing such clusters for traffic bursts.

Hairpinning: Hairpinning is when a packet is returned along the same path in the opposite direction somewhere in its way from source to destination. An IPv4 packet sourced by an end host has to be delivered to the gateway in the core network first, even it is destined to another customer of the same ISP. This is inefficient as all traffic has to be processed by the gateway. However, stateless mechanisms allow CPE-CPE direct paths. The source CPE does not send the packets to the gateway, as it can infer the destination IPv6 address from the packet's IPv4 address.

Logging requirements: In many jurisdictions, ISPs are required to identify customers based on an IP address and a timestamp. Traditionally, this was feasible because every customer was assigned a unique address either dynamically (e.g., via DHCP) or statically (fixed). Even in the former case, DHCP logging was possible, as only per-allocation logging was satisfactory. However, with ISP-level address sharing it becomes harder to identify customers based solely on an IP address and a timestamp. At any moment, many customers share the same IP address. When the gateway is stateful per flow, it is necessary to log all mappings of internal identifiers to public addresses. Moreover, if the authorities cannot provide ISP with the source port of the inspected connection, the ISP has to log destination IP addresses and destination port numbers, which introduces privacy concerns (in some countries, such logging is illegal). Per-

flow logging is resource intensive: It requires fast, reliable and large storage systems. See RFC6269 [184], section 12 for more on traceability.

Scalability: Scalability is critical for fast growing networks. Each new customer connected to the network causes hundreds of new flows being established. This requires larger state tables, more CPU power to match packets to the state table entries, and to synchronize clustered gateway nodes.

Per Allocation

State synchronization: Here, the entries in the gateway state table are changed when a customer is (de)allocated an IPv4 address or port-set. How fast the state changes depends on the ISP resource allocation policy and is related to the IP address sharing ratio. However, such state changes much less frequently than per-flow state.

Hairpinning: The issue is exactly the same as above.

Additional signaling: In A+P, the CPE needs to know its public IPv4 address and port-set for port-restriction. The per-allocation stateful A+P mechanisms do not encode IPv4 address and port-set information into the IPv6 prefix or address. Hence, additional signaling is needed to deliver this information from the gateway to the CPE.

Stateless

Dependency between IPv6 and IPv4 addressing: To derive the IPv4 address and the port-set from the IPv6 address or prefix assigned to the CPE, at least some bits of the IPv4 address and the port-set have to be encoded in them. This means that IPv4 and IPv6 addressing schemes are tightly coupled. If the CPE only has one IPv6 address or prefix assigned before deployment of a stateless mechanism, there are two deployment possibilities. First, complete IPv6 readdressing in the access network can be considered, which causes service unavailability and can be operationally demanding, especially if customers already rely on static IPv6 (prefix) assignments. Second, an additional IPv6 prefix for address sharing purposes can be assigned to each CPE, which could cause routing table inflation if route aggregation is not in place. Finally, any subsequent changes in IPv4 addressing and/or port-set allocation cause IPv6 readdressing as well.

Mapping rules: Mapping rules must be synchronized among all devices taking part in a stateless mechanism. These define how IPv4 prefixes reserved for IPv4 address sharing are mapped to IPv6 addresses and prefixes in the access network. If an ISP has many (smaller) IPv4 prefixes, the mapping rules can be impractical to administer.

Less efficient IPv4 address usage: As IPv4 address and port-set (re)allocations are non-trivial (because of the IPv4 and IPv6 addressing dependency shown above), it is more likely that ISPs will initially allocate IPv4 resources with less granularity (i.e., they will rather initially allocate 1024 ports to each customer, though they might not need them [167], rather than risk frequent reallocations as those could cause service degradation). Thus, in practice, stateless solutions could lead to lower IPv4 address sharing ratios than other A+P mechanisms.

Incompatible with Discontinuous IPv4 Address Blocks: Also, stateless tunneling is more difficult to use when an ISP has many smaller discontinuous IPv4 address blocks instead of a few large ones. For each IPv4 address range, separate IPv4-to-IPv6 mapping rules have to be administered in the domain (in CPEs and gateways).

3.2.3 Dimension 3: Traversal Method Through the Access Network

In this section, we explain how the traversal method of a mechanism influences possible MTU issues, packet inspection issues, security and performance issues, etc.

Routing

IPv4 routing does not encourage IPv6: Ideally, IPv4 address sharing mechanisms should encourage transition to IPv6 at least in some parts of the network. However, IPv4 routing does not encourage transition of IPv4-only networks to IPv6. Of course, dual-stack can be used in this case, but as its deployment is completely independent of such IPv4-only mechanisms, it is expected that a significant number of ISPs (short-visioned) will not consider it.

Tunneling

Maximum Transmission Unit (MTU) issues: Different sizes of IPv4 and IPv6 headers cause problems with handling the maximum packet size to any system connecting the two address families. There are four mechanisms for dealing with this issue: Path MTU Discovery (PMTUD) [198], fragmentation [187], transport-layer negotiation such as

the TCP Maximum Segment Size (MSS) option [199], and increasing MTU size on all the links in the access network at least by 40 B to accommodate both the IPv6 encapsulation header and the IPv4 datagram without fragmenting the IPv6 packet

Packet inspection issues: Any middleboxes in the access network that process IPv4 packets have to be able to unwrap tunneling to inspect one header deeper to discover the payload properly. Examples are Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) devices, which perform deep packet inspection or special environments, e.g., some 3rd Generation Partnership Project (3GPP) and PacketCable environments or transparent Web proxy caches. In these environments, significant additional support is needed in various devices [200].

Packet size overhead: Because of the additional header, tunneling causes bandwidth overhead compared to other traversal methods. With average payload of ≈ 550 B, tunneling causes around 4% overhead, while with average payload of 1400 B, it causes around 2% overhead [200].

Routing loop vulnerabilities: Tunneling makes routing loop attacks possible [74]. This vulnerability can be abused as a vehicle for traffic amplification to facilitate denial-of-service (DoS) attacks [201]. However, with address sharing mechanisms, filtering makes it relatively easy to mitigate such attacks.

Double Address Family Translation

Maximum Transmission Unit (MTU) issues: This issue is exactly the same as above.

Checksum recalculation: When the packets translated between IPv4 and IPv6, the transport-layer protocol checksums must be recalculated. This may impose a significant impact on overall performance, as whole packets have to be included in checksum recalculation. Even though stateless NAT64 avoids checksum recalculation in cases of checksum-neutral prefixes, this is not applicable to some mechanisms, where IPv6 addresses also encode port information [119].

Potentially limited transparency to IPv4 Do not Fragment (DF) bit: In general, stateless NAT64 is transparent to the IPv4 DF bit. However, if a stateless NAT64 implementation chooses to "provide a configuration function, that allows the translator not to

include the Fragment Header for the non-fragmented IPv6 packets", which is allowed by RFC 6145 [117], end-to-end DF bit transparency is broken.

Potentially limited transparency to IPv4 Type Of Service (TOS) field: In general, stateless NAT64 is transparent to the IPv4 TOS octet. However, as RFC 6145 [117] states, "an implementation of a translator SHOULD support an administratively configurable option to ignore the IPv4 TOS and always set the IPv6 traffic class (TC) to zero". In this case, IPv4 TOS transparency is broken.

Potentially unsupported fragmented zero-checksum UDP packets: RFC 6145 [117] states that fragmented IPv4 UDP packets that do not contain a UDP checksum are not in general translated by the stateless NAT64 translator. However, this need not be the case, as the translator can be configured to forward the packet without a UDP checksum, which will also work for zero-checksum UDP packets.

Limited transparency to ICMP: RFC 6145 [117] defines that some of ICMP [202] message types (13, 14, 15, 16, and others) are not translated by stateless NAT64, which means that end-to-end ICMP transparency is not preserved.

Loss of IPv4 header options: IP/ICMP protocol translation algorithms do not support translating IPv4 header options, which means they will be lost when a packet traverses v4-v6-v4 stateless translators. This should not have significant consequences, as IPv4 header options are very rarely used today. Even when used, approximately half of such packets are dropped somewhere on their path [203].

Reversible Header Translation

Maximum Transmission Unit (MTU) issues: This issue is exactly the same as above.

Loss of IPv4 header options: As with double address family translation, reversible header translation lacks support for translating IPv4 header options.

3.2.4 Dimension 4: Level of IPv6 Requirement

In this section, we discuss how the level of IPv6 requirement of mechanisms will impact the future Internet and the duration of IPv4/IPv6 coexistence.

No IPv6 Required

No IPv6 encouragement: None of these mechanisms will contribute to encouraging IPv6 transition because operators are not required by any means to even consider deploying IPv6 in any of their networks.

Administration of IPv4 infrastructure: We consider IPv4 protocol a legacy protocol, which means that eventually it will fade away and at that time administrating IPv4 infrastructure will not be necessary any more. Assuming IPv6 deployment is in place, IPv4 administration contributes extra significant network administration cost.

IPv6 Partly Required

IPv4 in customer networks: Since access networks normally represent a large part of an ISP's network, migrating them to IPv6 is a substantial move in the direction of IPv6 transition. However, if customer networks remain IPv4-only (or even dual-stack), this means IPv4 will be kept in use for a long time, which will prolong the transition to IPv6-only Internet. In this aspect, IPv4 address scarcity can be seen as a strong driver toward IPv6-only networks where feasible.

Administration of IPv4 infrastructure: The is exactly the same as for mechanisms where no IPv6 is required.

IPv6 Required

IPv4-only application incompatibility: We expect that at some point, the ISPs who find it difficult to administer IPv6 *and* IPv4 addressing in customer networks will consider deploying mechanisms which allow for IPv6-only customer networks. On IPv6-only end-hosts, IPv6 applications without support for IPv6 will not work. [204].

IP protocol-aware application incompatibility: Because connection endpoints use different address families, NAPT64 introduces incompatibilities with some application-layer protocols as shown in [204]. This is true for IP-protocol-aware application protocols—BitTorrent, FTP, and Session Initiation Protocol (SIP) [205] being widely used examples. For every such protocol, an ALG can be constructed, but each new ALG contributes more complexity to network operation.

Only IPv6-enabled hosts supported: Public IPv4 address sharing among dual-stack and IPv4-only end hosts is not supported by such mechanisms. This means that any non-IPv6 ready devices will not be able to connect to IPv4 services, which can be a serious limitation in heterogeneous environments. Legacy devices such as old faxes or printers with embedded networking are problematic examples.

Requires DNS64 service for operation: These mechanisms require DNS64 in order to be effective; this means another service to administer. It also means IPv4 traffic destined to IPv4 address literals are not supported. This means that if the end host tries to browse to `http://203.0.110.10`, requests will fail immediately, as no DNS request is made to cause synthesis of a usable IPv6 address.

3.2.5 Dimension 5: IPv4 Address and Port Allocation Policy

In this section, we discuss how this dimension impacts address sharing ratio, state storage in the gateway, and security.

Static and dynamic

Although mechanisms with this property support both allocation policies, we discuss the issue with dynamic allocation in this section and issues with static allocation in Section 3.2.5.

Stateful per flow: Dynamic allocation is stateful per flow, so we must record which resources are allocated to which flows. This introduces logging, scalability, state synchronization, and other issues (see Section 3.2.2).

Static-only

Low address sharing ratio: Since port-sets are allocated to customers instead of individual ports to flows, many ports remain unused. This is due to the need to allocate a large enough port-set to a customer so that they will never use all of the allocated ports (which would cause service degradation). Because the number of used ports by a customer can vary significantly, the worst case becomes the universal case. This means that, for any given address space, fewer customers can be offered service than with dynamic allocation.

Port randomization security issues: The TCP protocol is inherently vulnerable to spoofed off-path packet injection attacks [206]. To implement an attack on a TCP session established between two hosts, the adversary must guess the 4-tuple (source port, destination port, source address, destination address) of the TCP connection together with 32-bit sequence ID. The attack is feasible, and static port allocation makes the problem even worse—the 16-bit port space becomes smaller, which makes the 4-tuple easier to guess [200].

3.3 Mechanism Review and Classification

In this section, we identify and describe nine classes of IPv4 address sharing mechanisms (no specific order). Table 3.1 summarizes the properties for each class. Also, we provide a per-class outgoing packet flow diagram, which demonstrates the address sharing operation. Some of the classes contain multiple mechanisms, while others only have one member. This is because some classes contain competing mechanism proposals that are still being decided on in the IETF. There are 216 possible classes in the classification space. Many combinations do not make sense, e.g., having an address sharing function in the CPE and dynamic address and port allocation together. Analyzing the rest of the space in detail is out of scope of this thesis. It is important to note that it is not always straightforward to identify a nonsensical combination as such. Designing a class of mechanisms cannot be done by simply choosing one of the combinations with the desired features. Complete architecture with potential address mapping rules, address formats and packet manipulation functions must be considered and defined as well. Only when it is possible to successfully construct these elements into a working mechanism, it can be said that a combination of properties of the classification dimensions can be "populated" with a useful mechanism class.

The flow diagrams do not show the process of provisioning a CPE. The access network interface(s) of a CPE can be configured and provisioned using one of a variety of protocols, e.g., DHCP, DHCPv6, Port Control Protocol (PCP) [193], Technical Report 069 (TR-069) [207] or manually. For some scenarios, the CPE only requires an IPv4 or IPv6 prefix or both; for others, one or more port-sets or encapsulation parameters. How the CPE is provisioned with prefixes and port-sets is not important to our classification, as it does not affect tunneling, encapsulation, translation, etc. Each different form of provisioning offers a different set of features and a different level of complexity.

We now describe the basic operation of each class (Figures 3.1-3.8). An end-host sends IPv4 packets destined to the IPv4 Internet to the LAN default gateway that will be the CPE (in one of the classes, a preamble must be performed first to obtain a reachable IP address). Next, packets are forwarded by the CPE's external interface to the access network's default gateway where further processing takes place as necessary. From there, packets are forwarded to the IPv4 Internet. The numbers in the figures correspond to the consecutive steps required for sending a packet.

Figure 3.1

In Class 1 mechanisms, IPv4 traffic is processed by two successive NAPT44 functions, in the CPE and in the gateway.

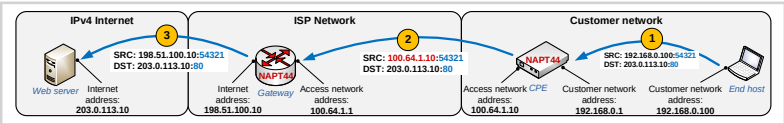


Figure 3.2

In Class 2 mechanisms, IPv4 traffic is tunneled in IPv6 packets and routed to the gateway, where the NAPT44 function is located.

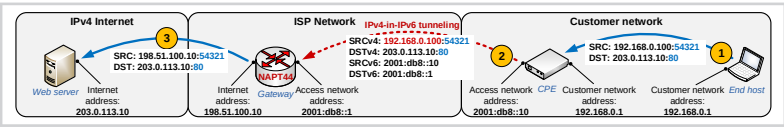


Figure 3.3

In Class 3 mechanisms, IPv4 traffic is tunneled to the intermediary gateway using one of the tunneling technologies (e.g., PPP or PPPoE) and then to the border gateway, where the NAPT44 function is located.

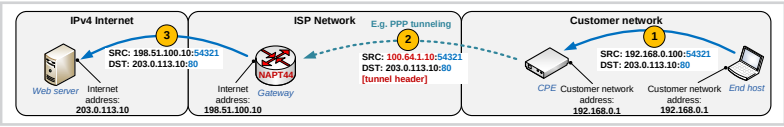


Table 3.1

IPv4 Address Sharing Mechanism Classes

Dimension	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*	*Class 6*	*Class 7*	*Class 8*	*Class 9*
Location of the IP address sharing function [3 possible values: CPE, gateway, CPE and gateway]	CPE and gateway	gateway	gateway	gateway	CPE	CPE	CPE	CPE	gateway
State storage in the gateway [3 possible values: stateless, per flow, per allocation]	per flow	per flow	per flow	per flow	stateless	per allocation	stateless	stateless	per flow
Traversal method through the access network [4 possible values: routing, tunneling, double address family translation, reversible header translation]	routing	tunneling	tunneling	routing	tunneling	tunneling	double address family transla-tion	reversible header transla-tion	double address family transla-tion
Level of IPv6 requirement [3 possible values: no IPv6 required, IPv6 partly required, IPv6 required]	no IPv6 required	IPv6 partly required	no IPv6 required	IPv6 required	IPv6 partly required	IPv6 partly required	IPv6 partly required	IPv6 partly required	IPv6 partly required
IPv4 address and port allocation policy [2 possible values: static and dynamic, static-only]	static and dynamic	static and dynamic	static and dynamic	static and dynamic	static-only	static-only	static-only	static-only	static and dynamic

3.3.1 Class 1

Given that NAPT44 functionality is already present in most CPE devices on the market, Class-1 mechanisms add an additional level of NAPT44 in the core of the ISP's network. ISPs have deployed such technology for a long time because it is simple and it builds solely on well-known NAPT44 translation. This is popular for aggressive IPv4 address sharing, but the end-to-end principle of the Internet is not preserved. To reduce addressing conflicts with RFC 1918 address space, IANA has allocated a special IPv4 address block to be used by ISPs for address sharing purposes [181].

NAT444 (sometimes called double NAT or CGN) is representative of this class (Figure 3.1). The IETF made some effort to standardize this [208] but the Draft expired. However, another Internet Draft [209] defines required behavior of CGNs in general.

3.3.2 Class 2

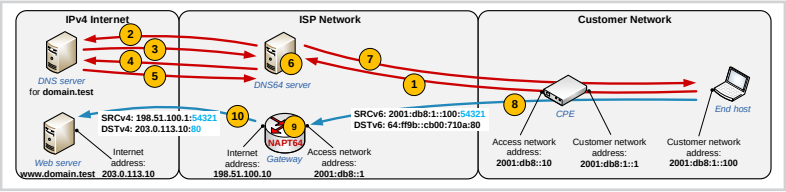


Figure 3.4

In Class 4 mechanisms, DNS64 server is used by IPv6-only hosts to provide synthetic IPv6 addresses which represent IPv4 hosts.

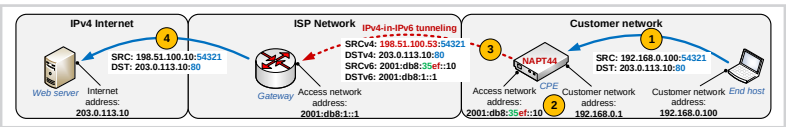


Figure 3.5

In Class 5 mechanisms, IPv4 traffic is first processed by NAPT44 in the CPE and then statelessly tunneled to the gateway, which routes it to the Internet.

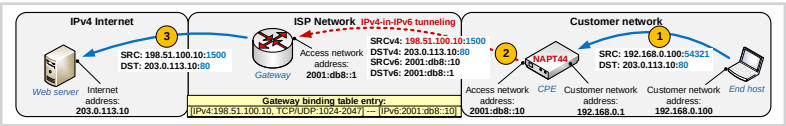


Figure 3.6

Class 6 mechanisms are very similar to Class 5 mechanisms, except that they do not encode IPv4 address and port-set information in IPv6 addresses but uses a binding table in the gateway instead.

The aim here is to remove double NAT by moving the NAPT44 function to the network core and away from the CPE. IPv4 traffic is tunneled between the CPE and the gateway over an IPv6 access network, which also allows elimination of addressing conflicts between customers.

DS-Lite [187] is representative of this class (Figure 3.2).

3.3.3 Class 3

This class is similar to Class 2 of CGN mechanisms—the main difference being allowance for other tunneling techniques rather than v4-over-v6, e.g., Point-to-Point Protocol (PPP) or Point-to-Point Over Ethernet (PPPoE). This means a Class 3 mechanism can be deployed without IPv6 at all.

Gateway Initiated DS-Lite [186] is representative of this class (Figure 3.3).

3.3.4 Class 4

This is a class of CGN mechanisms that use IPv6 as the fundamental protocol of the access network and carry IP packet contents in IPv6 packets before translating them for forwarding over the IPv4 Internet. In order to obtain the reachable IP address of the destination host, the IPv6-only end-host first queries its DNS resolver, usually the provider's DNS64 server. The DNS64 [118] server tries to fetch an AAAA resource record for the domain in question. If the domain is not IPv6-ready, this request fails, and the DNS64 server retries the query, this time by looking for an A record. Note A and AAAA record queries can be performed simultaneously to reduce delay. If the AAAA record exists, then the communication continues over IPv6 as usual. If no AAAA record is found, but an A record exists, the corresponding IPv4 address will be sent to the DNS64 server, which in turn algorithmically generates a synthetic IPv6 address using a common NAT64 prefix, which is routed via the NAT64 gateway. Such mechanisms do not allow direct IPv4 addressing of the end-hosts, but use Network Address and Port Translation from IPv6 to IPv4 (NAPT64) in the gateway to achieve IPv4 address sharing. NAPT64 translates IPv6 packets to IPv4 packets, and vice versa. This is significantly more complex than NAPT44. It causes additional issues compared to NAPT44 due to the address family translation [204].

Stateful NAT64 [30] is representative of this class (Figure 3.4).

3.3.5 Class 5

These mechanisms employ A+P at the CPE and leverage *stateless tunneling* (dimensions 2 and 3) for transferring IPv4 traffic across IPv6-only networks. All proposals in this class require no per-flow and per-allocation state in ISP's gateway. Thus, all information, required for routing packets on ISP's gateway, is derived algorithmically from fixed preconfigured domain-wide settings and information encoded in IPv6 addresses [210]. Also, as the NAPT function is located in the CPE, gateways can be lightweight. As an example, figure 3.5 shows the addressing format and port-set encoding of 4rd. However, different encodings are also possible.

The following mechanisms are representatives of this class (Figure 3.5): I-D.ietf-softwire-map [211], I-D.murakami-softwire-4rd [212], I-D.sun-softwire-stateless-4over6 [213], I-D.matsuhira-sa46t-as [188], AplusP Lite [210].

3.3.6 Class 6

Class-6 mechanisms again employ A+P, but use *stateful tunneling* as a traversal method (dimensions 2 and 3), which refers to per-allocation state required in the gateway to perform IPv4-in-IPv6 tunneling between the CPEs and the gateway. We are not referring to per-flow state, required for maintaining a NAT table in the gateway, as this is one of the A+P approaches. Additional signaling is needed to notify CPEs of their respective IPv4 addresses and port-sets: DHCP [214], PCP [215] and TR-069 variants are example protocols that serve this purpose.

The following mechanisms are representatives of this class (Figure 3.6): I-D.cui-softwire-b4-translated-ds-lite [216], I-D.zhou-softwire-b4-nat [217], I-D.draft-penno-softwire-sdnat [218].

3.3.7 Class 7

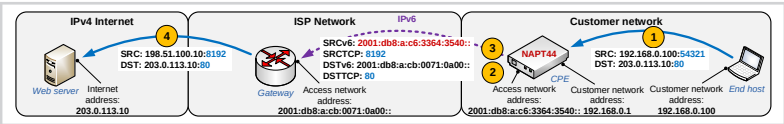


Figure 3.7

In Class 7 mechanisms, IPv4 traffic is translated to IPv6 in the CPE and then back to IPv4 in the ISP's gateway.

This class of A+P mechanisms is similar to Class 6 with tunneling replaced by double address family translation [147]. This eliminates issues of tunneling. First, the packets

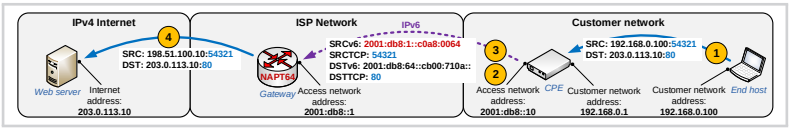


Figure 3.8

In Class 9 mechanisms, IPv4 traffic is statelessly translated to IPv6 in the CPE and then statefully back to IPv4 in the ISP's gateway.

are translated from IPv4 to IPv6 and then back from IPv6 to IPv4. Both translations are performed algorithmically and are completely stateless. In Figure 3.7, the addressing format and port-set encoding of Double IVI (dIVI) [219] is shown, but different formats are possible.

The following mechanisms are representatives of this class (Figure 3.7): I-D.ietf-softwire-map-t [220], I-D.xli-behave-divi-pd [221], I-D.murakami-softwire-4v6-translation [222], IVI [122, 147].

3.3.8 Class 8

This class of A+P mechanisms is similar to Class 7 with the exception of traversal method used. Reversible header translation is defined by 4rd mechanism. It uses an IPv6 fragmentation header to store some information from IPv4 header, making it reversible and almost lossless (only IPv4 options are lost, which is acceptable since they are not often used today in the Internet [203]). As this traversal method removes several limitations of tunneling and double address family translation (discussed in Section 3.2), this mechanism is considered as a class of its own.

4rd is representative of this class [190] (Figure 3.7).

3.3.9 Class 9

This class of CGN mechanisms is similar to Class 2 with the exception of traversal method and translation function used. However, it was developed to provide limited (outbound, client-server) IPv4 access to IPv4-only applications on directly connected IPv6-only provisioned end-hosts (no CPE involved).

464XLAT [223] is representative of this class (Figure 3.8).

3.4 Tradeoff Analysis

Having determined an appropriate set of dimensions for classifying address sharing mechanisms and performed a detailed property analysis of each dimension, it now

Table 3.2

Tradeoffs: classes of mechanisms trading the desired features (unordered)

Address-Plus-Port	*Carrier-Grade-NAT*
End-to-end connectivity (control over NAPT function)	Simple CPEs, easy provisioning and management
Scalability	Technology mature & available
Stateful	*Stateless*
Flexible addressing, no IPv6-IPv4 addressing dependency	Easy Load-Balancing
Efficient IPv4 address usage	Easy High-Availability
Scattered address-space supported	CPE-to-CPE direct paths
Tunneling	*Double Translation*
Keeps IPv4 packets intact	No routing loop vulnerability
No checksum recalc. overhead	No packet inspection issues
Mature and widespread method	No tunneling packet size overhead
IPv6 Required	*IPv6 Not Required*
Transition encouraged	Easy deployment
Less administration	Legacy application compatibility
Static Allocation	*Dynamic Allocation*
Manageable state	High address sharing ratio
Efficient logging	More secure

remains to select the most significant of these to determine the tradeoffs. Table 3.2 shows a summary.

3.4.1 Carrier-Grade-NAT Versus Address-Plus-Port

Infrastructure simplicity and ease of deployment together with technology maturity and availability are important features for ISPs as they easily translate to reduced costs. Also, waiting for A+P mechanisms to become widely delivered by vendors can mean losing customers in the meantime. However, ISP customers require end-to-end protocols and are not concerned with infrastructure issues. If users are not able to traverse CGNs to use very popular applications (gaming, VoIP, peer-to-peer, streaming), or if

these applications show significant performance degradation, the ISP market will start to segment by the quality of NAT traversal support (through ALGs). The scalability of A+P solutions is a further cost-reducing benefit to the ISP.

3.4.2 Stateful Versus Stateless

The benefits of stateful gateways mostly relate to stateful A+P solutions rather to CGN solutions. ISPs located in regions where Internet penetration is still gaining momentum often have many scattered IPv4 address ranges, which makes them good candidates for stateful solutions. Also, as these ISPs value IPv4 addresses, being able to effortlessly allocate different port-sets to customers in more nimble way is also welcome. In networks where IPv6 is already deployed in the access and the core networks, complete IPv6 readdressing increases cost of deploying a (stateless) IPv4 address sharing mechanism. In this case, independence of IPv6 and IPv4 addressing schemes is beneficial, although a separate IPv6 addressing scheme for address sharing purposes can be used, which introduces additional administrative complexity and cost. However, stateless solutions are attractive in several scenarios. Large ISPs with significant intercustomer traffic are motivated to search for stateless solutions that allow for direct intercustomer communication. Also, avoiding state eliminates many of the difficulties brought by state synchronization requirements, including those involved in supporting high-availability and load-balancing.

3.4.3 Tunneling Versus Double Translation

The third dimension has four different traversal methods. However, as routing and reversible header translation are related to specific mechanisms rather than mechanisms classes, the real decision is whether an ISP should choose a mechanism with IPv4-in-IPv6 tunneling or double stateless NAT64 translation. The former is a mature and proven method of carrying IPv4 packets over IPv6-only networks. The caveats are known, and workarounds are available. Tunneling protects the inner packet from being semantically distorted. However, double translation avoids the caveats of tunneling (e.g., those related to MTU) and also requires less processing in the path from the CPE to the gateway. This point is valid especially in those networks where packet inspection is performed.

3.4.4 IPv6 Required Versus IPv6 Not Required

In our classification IPv6 Required means required in the access and customer networks. Such schemes highly encourage IPv6 transition as only the ISP border remains configured with IPv4 address(es). IPv6-only networks require one Internet protocol less to administer. However, those mechanisms that do not require any IPv6 deployment (not even in the access network) are usually easier to deploy quickly and do not cause incompatibilities with legacy IPv4-only software.

3.4.5 Static Allocation Versus Dynamic Allocation

By choosing a mechanism with dynamic address and port allocation, the ISP can use a very small number of IPv4 addresses to support many customers as the sharing ratio can be an order of magnitude higher than a static allocation mechanism. However, even in the static case, we can easily multiplex 64 customers on one IPv4 address with each customer allocated 1024 ports. Compared to the current situation where one customer is allocated one public IPv4 address, the compression of static allocation is a major benefit. Together with the reduced flow state storage of static allocation comes more efficient logging, which is especially important as the ISPs frequently offer faster plans to customers, and being able to log flows in the dynamic allocation case is about four orders of magnitude more storage-intensive than static allocation, where only port-set allocations need be logged. It is important to note that it is dynamic allocation that causes logging problems, not CGN mechanisms alone. Dynamic allocation schemes are less prone to spoofed off-path injection attacks on TCP sessions.

3.5 Discussion

Throughout this chapter, we considered ISP networks which are now short of available IPv4 addresses and are forced to deploy one or more IPv4 address sharing mechanisms.

The CGN versus A+P dilemma is not the same as "NAPT-in-the-CPE" versus "NAPT-in-the-core" dilemma, which we believe is the common misconception. As our classification finally separates the dimensions and analyzes them individually, this enables us to see clearer that logging complexity, for example, is not dependent on the location of the NAPT function, but rather on the IPv4 address and port allocation policy; a very important difference. We induce the following insights from the work in previous sections:

1. Address translation and port-restriction can be regarded as two separate functions, which can be performed at different places independently. This opens a space for new mechanisms that have not been envisioned before.

2. Address family translation in the context of traversal method is completely unrelated to classical NAPT. In our experience, there is a lot of misunderstanding of various roles translation (in general) can play in the context of address sharing mechanisms. This misunderstanding is mainly based on the fact that there is no fundamental framework available to the community in which to operate and view the various proposals. The chaos in the IETF is an obvious result of this confusion. A solid framework which we suggested in this chapter will help in more structured progress on this topic in the future.

3. The only actual address sharing mechanism that really pushes forward the transition to IPv6 is Stateful NAT64 (Class 4). All other (classes of) mechanisms are more tolerant to IPv4. More research is needed in this direction if our goal is to encourage IPv6 transition.

We realize the IETF is still actively working on defining details of and standardizing various IPv4 address sharing mechanisms. Although industry is pressing for a stop to the research and development of new mechanisms and for standardization and deployment of current proposals, we show in the rest of the thesis, that there are still gaps to fill in this area. For example, using our classification, we envision a mechanism with the following properties: address sharing function located in the CPE (A+P), stateless gateway, routing as the access network traversal method, IPv6 required in the access and the customer network, and static address and port allocation. Such a mechanism would highly encourage IPv6 transition and would have all the benefits of A+P and stateless mechanisms. The classification presented in this chapter provides grounds for work in the next chapters because it provides the necessary abstraction of the mechanisms in the form of various classes.

4

AP64: A Scalable IPv4 Address Sharing Mechanism for IPv6 Networks

Thus far, we have discussed existing IPv6 transition technologies. We have reviewed IPv6 deployment mechanisms in Chapter 2 and systematized IPv4 address sharing mechanisms in Chapter 3.

At the end of the previous chapter, we presented a mechanism, that is designed to fill a gap in the IPv4 address sharing mechanism design space. In this chapter, we present the AP64 mechanism, which satisfies all of the most desired requirements. First, we discuss the motivation for developing the AP64 mechanism. Next, we provide the definition of the mechanism. Finally, we describe the architecture and provide all the technical details, including the description of the operation of individual mechanism functions.

4.1 Motivation

One of the purposes of researching IPv4 address sharing mechanisms is to discover those empty parts of the design space that should group together as many desired properties as possible.

We present a novel mechanism that would support IPv6-only customer networks, allow for a stateless gateway and avoid all of the hurdles of tunneling, double translation, and reversible header translation traversal methods. A mechanism possessing all these properties has not been proposed thus far. The main three high-level benefits of such a mechanism are the following:

- *Highly IPv6-encouraging*: IPv6-only customer networks are supported. This means that no IPv4 addressing is provisioned on customer network hosts' interfaces. However, these networks have access to IPv4 Internet in such a way that multiple customer networks share one IPv4 address. We envision a future period of IPv6 transition when there will be few (or no) reasons to continue with provisioning dual stack to customer networks, as the administration of IPv4 and IPv6 is more complex and time consuming than is the administration of IPv6 only. After all major Internet services are IPv6-ready, it will suffice to have limited access to IPv4 Internet, which is a side-effect of IPv6-to-IPv4 translation mechanisms using DNS64.

- *Scalable*: AP64 is a stateless A+P mechanism, which means that no per-flow or per-allocation state needs to be maintained in the gateway. The IPv6-to-IPv4

translation is performed algorithmically. This greatly simplifies high availability and load balancing functionalities and allows for anycast routing to the gateways. Also, the access network is more scalable because adding new CPEs is easy and the load is only dependent on the number of packets flowing through the gateway and not on the number of established sessions.

- *Routed*: Tunneling, double translation and reversible header translation traversal methods introduce their own additional issues: MTU problems, fragmentation issues, packet inspection issues etc. AP64 only uses IPv6 routing in the access network, which means that no special treatment is required for the traversal method.

4.2 Definition

As AP64 does not fall into any of the 9 classes of the classification, we introduce a new class – Class 10, which opens a new "box" in the design space. Furthermore, we introduce the terms and present the AP64 architecture. We provide a complete description of the mechanism, which is ready to be implemented for real-world deployment. Included in the description are the mapping rules, address and port mapping algorithms and AP64 device functions.

4.2.1 A New Class: Class 10

Considering the properties and dimensions of the classification introduced in Chapter 3, we identify a new possible combination of properties:

- location of the IP address sharing: *CPE*,
- state storage in the gateway: *stateless*,
- traversal method through the access network: *routing*,
- level of IPv6 requirement: *IPv6 required*,
- IPv4 address and port allocation policy: *static-only*.

Table 4.1

IPv4 Address Sharing Mechanism Classes

Dimension	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*	*Class 6*	*Class 7*	*Class 8*	*Class 9*	*Class 10*
Location of the IP address sharing function [CPE, gateway, CPE and gateway]	CPE and gateway	gateway	gateway	gateway	CPE	CPE	CPE	CPE	gateway	*CPE*
State storage in the gateway [stateless, per flow, per allocation]	per flow	per flow	per flow	per flow	stateless	per allocation	stateless	stateless	per flow	*stateless*
Traversal method through the access network [routing, tunneling, double address family translation, reversible header translation]	routing	tunneling	tunneling	routing	tunneling	tunneling	double address family transla-tion	reversible header transla-tion	double address family transla-tion	*routing*
Level of IPv6 requirement [no IPv6 required, IPv6 partly required, IPv6 required]	no IPv6 required	IPv6 partly required	no IPv6 required	IPv6 required	IPv6 partly required	IPv6 partly required	IPv6 partly required	IPv6 partly required	IPv6 partly required	*IPv6 required*
IPv4 address and port allocation policy [static and dynamic, static-only]	static and dynamic	static and dynamic	static and dynamic	static and dynamic	static-only	static-only	static-only	static-only	static and dynamic	*static-only*

We refer to this combination of properties as *Class 10*. A feature comparison mechanism classes including the new Class 10 is outlined in Table 4.1.

In order to better understand the whole design space of the mechanism classification, we give a heatmap visualization of all 216 possible combinations of dimension properties in Figure 4.1. We have attributed a value of 1, 2 or 4 to each of the properties for every dimension. The value was attributed based on the results of the detailed property analysis in Chapter 3. The list of the properties with their corresponding values is the following:

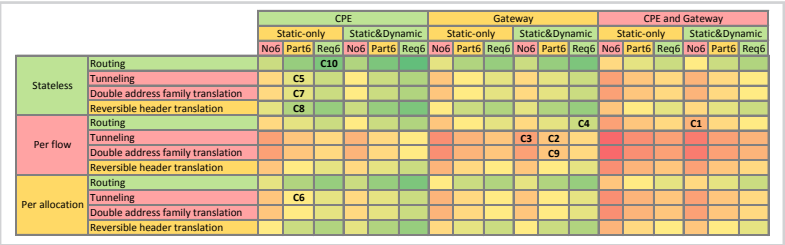


Figure 4.1

Heatmap visualization of all 216 possible combinations of dimension properties. Lowest sums are colored green, while highest sums are colored red. Labels C1 to C10 designate mechanism classes from class 1 to class 10.

- *Dimension 1: Location of the IP address sharing function*
  - CPE: 1
  - gateway: 2
  - CPE and gateway: 4
- *Dimension 2: State storage in the gateway*
  - stateless: 1
  - per allocation: 2
  - per flow: 4
- *Dimension 3: Traversal method through the access network*
  - routing: 1
  - reversible header translation: 2
  - tunneling: 4

    - double address family translation: 4

- *Dimension 4: Level of IPv6 requirement*
    - IPv6 required: 1
    - IPv6 partly required: 2
    - no IPv6 required: 4
- *Dimension 5: IPv4 address and port allocation policy*
    - static and dynamic: 1
    - static-only: 2

For every point in the classification space, we have summed the values of all 5 properties. These sums were used to generate the heatmap. Lowest sums are colored green, while highest sums are colored red. Labels C1 to C10 designate mechanism classes from 1 to 10.

Being an A+P mechanism class, Class 10 does not allow for dynamic port allocation policy. This negatively influences the maximum IPv4 address sharing ratio, which is orders of magnitude lower than in dynamic allocation schemes. However, when multiplexing 64 customers on one IPv4 address, each customer is able to use 1024 transport-layer ports, which is still considered sufficient for consumer customer networks [167].

The AP64 mechanism, as defined in the following sections, is a Class-10 mechanism.

4.2.2 Terminology

We begin by defining AP64 mechanism terms which are used in the remainder of this chapter.

- *AP64 domain:* a set of interconnected AP64 CPEs and AP64 gateways. An ISP can deploy one or more AP64 domains.
- *AP64 rule:* a set consisting of a Rule IPv6 Prefix, a Rule IPv4 Prefix and a length of Extended Address (EA) bits field.
- *AP64 CPE:* a CPE that implements the AP64 mechanism.

- *AP64 gateway:* a gateway that implements the AP64 mechanism.
- *Port-set:* a unique set of transport-layer ports, allocated to an AP64 CPE.
- *Port-set ID (PSID):* a set of bits, uniquely identifying a port-set allocated to an AP64 CPE.
- *Customer IPv6 prefix:* an IPv6 prefix, assigned to an AP64 CPE by a provisioning method that is unrelated to AP64 (DHCPv6-PD [224], SLAAC [93] or manually).
- *AP64 IPv6 address:* an IPv6 address that is used to reach AP64 CPE within the access network and for the IPv6 source address translation function (NAPT66).
- *Rule IPv6 prefix:* an IPv6 prefix assigned by an ISP for a mapping rule.
- *Rule IPv4 prefix:* a global IPv4 prefix assigned by an ISP for a mapping rule.
- *Embedded bits:* identify a (part of) shared IPv4 address and a port-set ID.

4.2.3 Architecture

The AP64 mechanism uses the well known SIIT [117] IP address family translation in the gateway to algorithmically translate IPv6 packets into IPv4 packets and vice versa. However, it uses a completely new component in the CPE. We refer to this component as *Port-Restricted NAPT66 (PR-NAPT66)*. This is a function that statefully translates source and destination IPv6 addresses and transport-layer identifiers (TCP and UDP ports, ICMP identifiers, fragment identifiers). Original identifiers must be translated by PR-NAPT66 so that they fit into the port-set that is allocated to the AP64 CPE.

Nearly all the IPv4 address sharing Internet Drafts that are published within IETF Softwire working group describe their own address format and address/port mapping algorithms. After an analysis document [185] that details the advantages and disadvantages of each algorithm was published, the IETF started a Mapping of Address and Port (MAP) project group. The goal of this group was to come up with a port/address mapping framework which could then be applied to various mechanisms. As a result, a Standards Track Internet Draft is now being discussed [211], which defines the mapping framework as well as the IETF-preferred IPv4-over-IPv6 address sharing mechanism, using tunneling as the access network traversal mechanism. We lean on MAP framework for to define address and port mapping algorithms of AP64.

4.2.4 Mapping Rules

To allow for stateless gateway operation, IPv4-embedded IPv6 addresses [119] must be used. This means that IPv4 address and port-set information is embedded into IPv6 addresses. The mappings between these are performed by address mapping algorithms according to the *mapping rules*. Mapping rules direct the forwarding of packets by CPEs and gateways. Every AP64 node is provisioned with one or more mapping rules. The same set of rules must be applied within an AP64 domain.

When there are multiple IPv4 blocks to be used for IPv4 address sharing, (additional) forwarding mapping rules must be used. Traffic directed outside the AP64 domain (to the IPv4 Internet) is forwarded to the gateway.

There are two types of mapping rules:

- *Basic Mapping Rule (BMR):* every AP64 device must have such a rule. In combination with Customer IPv6 prefix, the BMR is used to derive the AP64 IPv6 address for a device, as well as the shared IPv4 address and the port-set assigned to the CPE. It is also used for the mapping functions operation (forwarding).
- *Forwarding Mapping Rule (FMR):* optional additional mapping rules used for mapping functions operation (forwarding). The Basic Mapping Rule is also a Forwarding Mapping Rule. Given a destination IPv4 address and port within the AP64 domain, the matching FMR may be used to derive the AP64 IPv6 address of the interface, through which that particular IPv4 destination address and port combination can be reached.

Mapping rules of both types consist of the following three parameters: *Rule IPv6 prefix* (including the prefix length), *Rule IPv4 prefix* (including the prefix length) and *Rule EA-bits length* (in bits).

AP64 CPE or gateway finds its BMR by performing a longest match between the Customer IPv6 prefix and the Rule IPv6 prefix, while considering the entire set of mapping rules. BMR is in turn used for shared IPv4 address derivation. After BMR is selected, the AP64 IPv6 address is derived by every CPE and the gateway formed from the BMR's Rule IPv6 prefix. This address is assigned to the access network interface of the AP64 device and is used to terminate all AP64 traffic that is sent or received by the device.

4.2.5 Port Mapping Algorithm

While describing an IPv4 address sharing mechanism, we need to multiplex multiple customers on a single IPv4 address using application-layer identifiers for differentiating between them. To do this in the A+P manner [174], we must allocate non-overlapping port-sets to the CPEs.

Port is a 16-bit field used in TCP and UDP transport-layer protocols that allows multiple applications to share a single network connection. As not only TCP and UDP ports, but also ICMP and fragment identifiers, are influenced by the port-restricted translation function (PR-NAPT66), the notion of port must be generalized to include ICMP and fragment identifiers as well.

As a result, *PortMapping* algorithm, also known as Generalized Modulus Algorithm [211], requires input parameters k, r, m, and s and yields a port-set p:

- k: series of bits, uniquely identifying a port-set for an IPv4 address (PSID).
- r: number of CPEs sharing an IPv4 address (IPv4 address sharing ratio), must be a power of 2. It holds that $r = \max k + 1$.
- m: number of contiguous ports in a port range within a port-set, must be a power of 2.
- s: number of ports from 0 to $s-1$ to exclude, must be a power of 2. The default is 1024, which includes all System Ports [225].
- P: port-set allocated to the CPE with PSID k.

Pseudo-code for *PortMapping* is shown in Algorithm 1. Given a port p, number of contiguous ports m, and address sharing ratio r, PSID k can be derived using the formula $k = \lfloor p/m \rfloor \mod r$, where $m = 16 - \log_2 r - \log_2 s$ (*InversedPortMapping* algorithm).

Figure 4.2 graphically demonstrates how a particular 16-bit port (member of a port-set) is constructed. In the middle, there is the PSID (k). On the left-hand side, there is an port-range index (j) and on the right-hand side (i) is the port identifier within a port-range.

Algorithm 1

PortMapping algorithm

```
1: P ← empty set
2: for i ← 0, m − 1 do                          ▷ Range of contiguous ports
3:     for j ← (s/m)/r, (65536/m)/r) − 1 do     ▷ All the contiguous port ranges
4:         p ← (r ∗ m ∗ j + m ∗ k + i)
5:         Add p into port-set P
6:     end for
7: end for
```

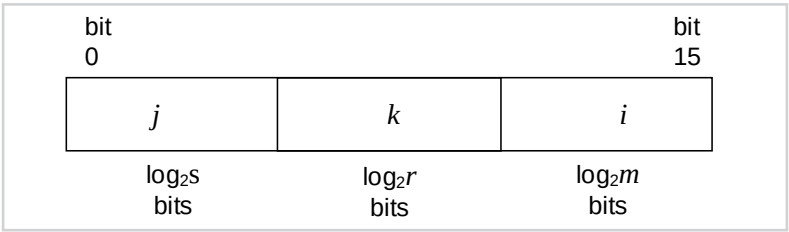


Figure 4.2

Bit representation of a port in a port-set.

4.2.6 *Address Mapping Algorithms*

Address mapping algorithms are used by CPE and gateway devices to map IP addresses and transport-layer identifiers into different IP addresses and vice versa. IPv4 address mapping into AP64 IPv6 addresses is necessary to allow for completely stateless gateway operation. We define the following three AP64 address mapping algorithms together with their corresponding input and output parameters.

- *AP64AddressPortSet:* this algorithm is used by the CPE to provision itself an AP64 IPv6 address and a port-set, both of which are required by PR-NAPT66 and NAPT66 functions.
  - Input:
    - *Customer IPv6 prefix:* `CPrefix`
    - *BMR rule IPv6 prefix:* `BMR6Prefix`
    - *BMR rule IPv4 prefix:* `BMR4Prefix`

        - *BMR EA-bits length:* `BMREALength`
    - Output:
        - *AP64 IPv6 address:* `AP64Addr`
        - *PSID:* `PSID`

- *IPv4PortToAP64:* this algorithm is used by the gateway's SIIT function to translate incoming IPv4 packets into AP64 packets.
    - Input:
        - *IPv4 address:* `IPv4Addr`
        - *Transport-layer identifier:* `Port`
        - *FMR rule IPv6 prefixes:* `FMR6Prefixes`
        - *FMR rule IPv4 prefixes:* `FMR4Prefixes`
        - *FMR EA-bits lengths:* `FMREALenghts`
    - Output:
        - *AP64 IPv6 address:* `AP64Addr`

- *AP64ToIPv4:* this algorithm is used by the gateway's SIIT function to translate outgoing AP64 packets into IPv4 packets.
    - Input:
        - *AP64 IPv6 address:* `AP64Addr`
        - *FMR rule IPv6 prefixes:* `FMR6Prefixes`
        - *FMR rule IPv4 prefixes:* `FMR4Prefixes`
        - *FMR EA-bits lengths:* `FMREALengths`
    - Output:
        - *IPv4 address:* `IPv4Addr`

Pseudo-code for *AP64AddressPortSet*, *IPv4PortToAP64*, and *AP64ToIPv4* is shown in Algorithms 2, 3, and 4 respectively.

Additional address mapping algorithms are needed for AP64 to function:

- *IPv4Embedded64* and *IPv4Embedded46*: according to "IPv4-Embedded" address format, defined in RFC6145 [117], when IPv6 addresses are mapped into IPv4 addresses and vice versa.
- *AP64Masquerade*: trivial address mapping that simply replaces the real source IPv6 address with the AP64 IPv6 address of the CPE.
- *MatchState*: trivial address mapping that replaces the destination address of the returning packets to their original address according to NAPT bindings.

Algorithm 2

AP64AddressPortSet address mapping algorithm. Input: CPrefix, BMR6Prefix, BMR4Prefix, BMREALength. Output: AP64Addr, PSID. Temporary variables: EAOffset, EABits, IPv4Addr, PSIDPadded, IID, AP64Prefix.

1: *EAOffset* ← length of *BMR6Prefix*.
2: *EABits* ← first *BMREALength* bits of *CPrefix* at offset *EAOffset*.
3: *IPv4Addr* ← concatenate *BMR4Prefix* and as many of the most significant bits of *EABits* as are required to obtain 32 bits.
4: *PSID* ← the rest of the bits in *EABits*.
5: *PSIDPadded* ← left-zero-pad *PSID* to 16 bits.
6: *IID* ← concatenate 16 zero bits, *IPv4Addr* and *PSIDPadded*.
7: *AP64Prefix* ← right-zero-pad *CPrefix* to 64 bits, if needed.
8: *AP64Addr* ← concatenate *AP64Prefix* and *IID*.

4.2.7 Packet Translation Functions

Both AP64 CPE and gateway apply several packet translation functions to incoming and outgoing packets. Ports and IPv6 addresses are translated into PR-NAPT66 and NAPT66, and IPv4/IPv6 packet headers are replaced by SIIT.

SIIT

Stateless IP/ICMP Translation (SIIT) [117] has already been discussed in Chapter 2. In the context of AP64, SIIT translation function is used by the gateway to algorithmically translate IPv4 headers into IPv6 headers and vice versa, using different address mapping algorithms.

Algorithm 3

IPv4PortToAP64 address mapping algorithm. Input: IPv4Addr, Port, FMR6Prefixes, FMR4Prefixes, FMREALengths. Output: AP64Addr. Temporary variables: FMR6Prefix, FMR4Prefix, FMREALength, FMR4PrefixLength, PSIDLength, PSID, PSIDPadded, IID, IPv4Suffix, AP64Prefix.

1: {*FMR6Prefix*,*FMR4Prefix*,*FMREALength*} ← among all FMR rules, allocated by *FMR6Prefixes*, *FMR4Prefixes* and *FMREALengths*, select the one in which *IPv4Addr* has the longest match with *FMR4Prefix*.
2: *FMR4PrefixLength* ← length of FMR4Prefix.
3: *PSIDLength* ← *FMREALength*−(32−*FMR4PrefixLength*).
4: *PSID* ← result of *InversedPortMapping* algorithm, where p = *Port* and $r = 2^{PSIDLength}$.
5: *PSIDPadded* ← left-zero-pad *PSID* to 16 bits.
6: *IID* ← concatenate 16 zero bits, *IPv4Addr* and *PSIDPadded*.
7: *IPv4Suffix* ← the rest of the bits in *IPv4Addr* which are not contained within *FMR4Prefix*.
8: *AP64Prefix* ← concatenate *FMR6Prefix*, *IPv4Suffix*, *PSID* and as many zero bits as required to obtain a 64-bit prefix.
9: *AP64Addr* ← concatenate *AP64Prefix* and *IID*.

Algorithm 4

AP64ToIPv4 address mapping algorithm. Input: AP64Addr, FMR6Prefixes, FMR4Prefixes, FMREALengths. Output: IPv4Addr. Temporary variables: FMR6Prefix, FMR4Prefix, FMREALength, EAOffset, EABits.

1: {*FMR6Prefix*,*FMR4Prefix*,*FMREALength*} ← among all FMR rules given by *FMR6Prefixes*, *FMR4Prefixes* and *FMREALengths*, select the one where the *AP64Addr* has the longest match with *FMR6Prefix*.
2: *EAOffset* ← length of *FMR6Prefix*.
3: *EABits* ← first *FMREALength* bits of *AP64Addr* at offset *EAOffset*.
4: *IPv4Addr* ← concatenate *BMR4Prefix* and as many of the most significant bits of *EABits* as are required to obtain 32 bits.

PR-NAPT66

Network Address and Port Translation from IPv6 to IPv6 is a general stateful IPv6 NAPT function that can translate source and/or destination IPv6 addresses and transport-layer identifiers. It is tightly related to NAPT44 [20] and has been rejected by IETF as harmful and unnecessary for general use for the stated reason that it causes end-to-end connectivity issues in the IPv6 world, just as NAPT44 does in IPv4 world. However, NAPT66 has nevertheless been implemented and the implementation is publicly available [226].

Port-Restricted Network Address and Port Translation from IPv6 to IPv6 (PR-NAPT66) is a variant of NAPT66 with port-restriction functionality. When a packet is received by the CPE from an internal end-host, the IPv6 address is translated according to address mapping algorithms and the source port of the packet is rewritten so that it fits the port-set provisioned to the CPE. This allows for IPv4 address sharing among multiple customers.

It is imperative to note that in AP64, NAPT66 address translation is only used in the IPv4 address sharing context. This does not affect native IPv6 connectivity in any way.

4.2.8 Operation

Figure 4.3 shows an AP64 CPE manipulating outgoing and returning packets from an end-host to the IPv4 Internet and back. The manipulation process includes the following steps:

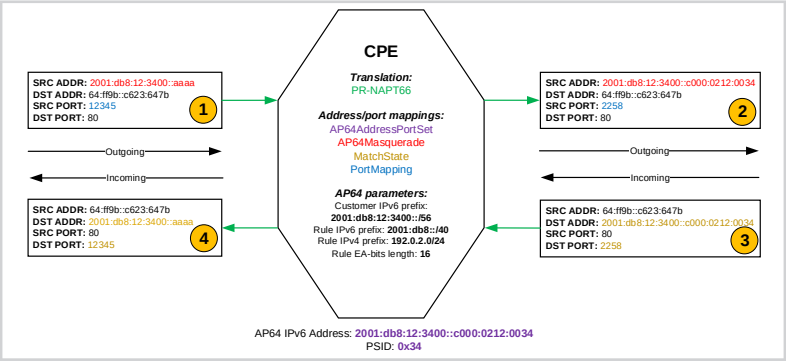


Figure 4.3

AP64 CPE manipulating outgoing and returning packets sourced by an end-host directed to IPv4 Internet.

- *Step 0:* When AP64 CPE is initialized, the *AP64AddressPortSet* address mapping algorithm is used to derive the AP64 IPv6 address and PSID for the CPE.
- *Step 1:* The initial packet is received by the CPE from an IPv6-only end-host. The packet is destined to the IPv4 Internet and IPv4-embedded IPv6 address is used as the destination IP address. This address is synthesized from a corresponding IPv4 address by the DNS64 server that is being used by the end-host.
- *Step 2:* The source IP address of the packet is translated so that the new source IP address is the AP64 IPv6 address of the CPE. *AP64Masquerade* mapping algorithm is used. This is required because the gateway has no way of algorithmically deriving the original IPv6 address of the end-host when the returning IPv4 packet is translated into the corresponding IPv6 packet. It can, however, algorithmically derive the AP64 IPv6 address of the CPE. Port-restriction is also enforced in this step, so that the source port is translated according to the *PortMapping* algorithm if it does not already fit into the provisioned port-set of the CPE.
- *Step 3:* The returning packet is received by the CPE. Source and destination addresses and ports are reversed compared to the packet in Step 2.
- *Step 4:* The destination IP address and the destination port (if necessary) are translated by the CPE according to the existing NAPT binding table entries, denoted as the *MatchState* address mapping algorithm.

Figure 4.4 shows an AP64 CPE manipulating outgoing and returning packets from an end-host to a CPE in the same AP64 domain (inter-customer IPv4 traffic). The manipulation process includes the following steps:

- *Step 0:* When the AP64 CPE is initialized, the *AP64AddressPortSet* address mapping algorithm is used to derive the AP64 IPv6 address and the PSID for the CPE.
- *Step 1:* Initial packet is received by the CPE from an IPv6-only end-host. The packet is destined to another CPE in the same AP64 domain and an IPv4-embedded IPv6 address is used as the destination IP address. This address is synthesized from a corresponding shared IPv4 address of the destination CPE by the DNS64 server that is used by the end-host.

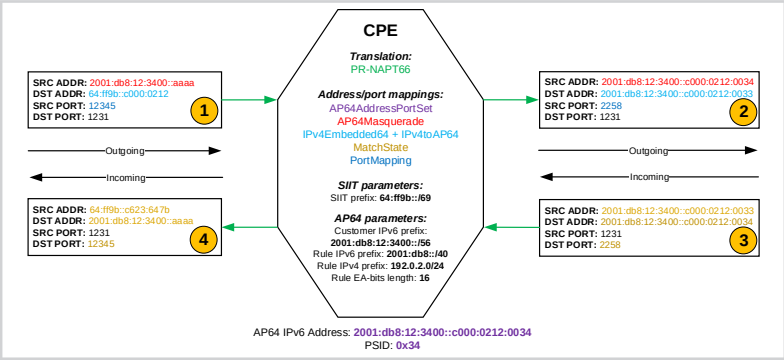


Figure 4.4

AP64 CPE manipulating outgoing and returning packets sourced by an end-host directed to another CPE.

- *Step 2:* The source IP address of the packet is translated so that the new source IP address is the AP64 IPv6 address of the CPE. *AP64Masquerade* mapping algorithm is used. This is required because the gateway has no way to algorithmically derive the original IPv6 address of the end-host when the returning IPv4 packet is translated to the corresponding IPv6 packet. It can, however, algorithmically derive the AP64 IPv6 address of the CPE. Port-restriction is also enforced in this step so that the source port is translated according to the *PortMapping* algorithm if it does not already fit into the provisioned port-set of the CPE. The destination address is also translated. By chaining address mapping algorithms *IPv4Embedded64* and *IPv4toAP64*, the original destination IP address is first mapped into an IPv4 address, which is in turn mapped into the AP64 IPv6 address of the destination CPE.

- *Step 3:* The returning packet is received by the CPE. Source and destination addresses and ports are reversed compared with the packet in Step 2.

- *Step 4:* The source and destination IP addresses and the destination port (if necessary) are translated by the CPE according to existing NAPT binding table entries, denoted as the *MatchState* address mapping algorithm.

Figure 4.5 shows an AP64 gateway manipulating outgoing and returning packets from an IPv6-only end-host to the IPv4 Internet. The manipulation process includes the following steps:

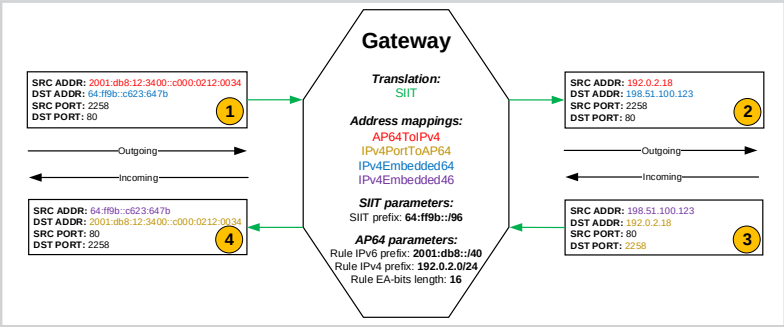


Figure 4.5

AP64 gateway manipulating outgoing and returning packets sourced by an end-host directed to IPv4 Internet.

- *Step 1:* The initial packet is received by the gateway from an AP64 CPE (sourced by an IPv6-only end-host). The packet is destined to the IPv4 Internet and an IPv4-embedded IPv6 address is used as the destination IP address. This address is synthesized from a corresponding IPv4 address by the DNS64 server used by the end-host. The source IP address belongs to the AP64 CPE which has sent the packet to the gateway.
- *Step 2:* The IPv6 header is stripped off the packet and a new IPv4 header is algorithmically generated by the SIIT translation function according to the *AP64ToIPv4* and *IPv4Embedded64* address mapping algorithms.
- *Step 3:* The returning packet is received by the gateway. Source and destination addresses and ports are reversed compared with the packet in Step 2.
- *Step 4:* IPv4 header is stripped off the packet and a new IPv6 header is algorithmically generated by the SIIT translation function according to the *IPv4PortToAP64* and *IPv4Embedded46* address mapping algorithms.

AP64 CPEs and gateways operating in the way specified allow for IPv4 address sharing among multiple customers. Each customer only consumes a portion of a shared IPv4 address. Application-layer identifier (port) multiplexing is used.

4.2.9 Discussion

In this chapter we have described a novel mechanism class for IPv4 address sharing that exhibits a set of properties that have been proved to be beneficial for internet service

provider network operation. In order to validate its value, a fair comparison to other mechanism classes is desired. As we want to compare all mechanism classes, we treat AP64 and its new class of mechanisms equally in this aspect. In the next chapter we propose a performance evaluation framework for IPv4 address sharing mechanisms, that we use for comparing all the mechanism classes including Class 10.

5

Theoretical Performance Analysis and Comparison Framework

In this chapter, we propose a performance analysis and comparison framework for IPv4 address sharing mechanisms. This is essentially a set of tools that enable researchers to evaluate and compare the performance of various transition mechanisms in a fair way, i.e., with minimal hardware dependency, implementation and other experimental bias. The basic idea of the framework is to decompose the mechanisms that are to be compared and evaluated into basic building blocks – basic packet processing operations that are experimentally evaluated individually only once. We provide experimental performance evaluations for all of the basic operations that are used in the mechanism classes and argue for their universality. By summarizing the evaluations of the building blocks, theoretical evaluations for any transition mechanism, that leverages these operations may be provided.

5.1 Motivation

Evaluating the performance of IPv6 transition mechanisms is a daunting task. In failing two major experiments that were designed to measure the performance of various transition mechanism implementations. the following lessons were learned:

- Running implementations of mechanisms on the same or equivalent hardware is essential, but insufficient. By using a virtual environment, a fair distribution of physical resources could be assumed.
- Comparing the measured performances of user-space and kernel-space mechanisms provides no useful results. This became a major issue because many of the implementations that were available were just user-space prototypes.
- Comparing mechanism implementations, that run on different operating system platforms is also impossible. This was a major issue in the course of this research because one of the mechanisms was implemented in NetBSD, while the others were running on Linux.
- Evaluating a mechanism "as a whole" is less accurate than evaluating it "per partes", i.e., evaluating its components individually and then adding them up to arrive at the final results.

In order to evaluate the performance of IPv4 address sharing mechanisms regardless of these limitations, we propose a theoretical performance evaluation framework

for IPv4 address sharing mechanisms. We further evaluate the implementations of the basic blocks of all mechanisms. These evaluations can in turn be used to calculate the theoretical performance evaluations of actual mechanisms. In this way, the performance of mechanisms may be evaluated without using the implementations of "whole" mechanisms.

5.2 Methodology

The main goal of this scientific research is to provide a concise and well-defined methodology that can effectively be used to theoretically evaluate the performance of IPv4 address sharing mechanisms. The methodology should allow a researcher to selet a new mechanism, identify its leveraging of basic packet operations and calculate a performance index for the mechanism that is comparable to performance indices of other IPv4 address sharing mechanisms. This will assist in designing more efficient mechanisms in future. This will also extend our tradeoff analysis from Chapter 3, so that practitioners will be able to grasp a sense of quantitative impact of the mechanism they are considering deploying.

In evaluating the performance of mechanisms, the following metrics are considered:

- *CPE Time*: How much processing time does a CPE require to process packet, that is either initializing a flow, or is a part of an existing flow?
- *CPE Space*: How much storage space does a CPE require to store flow state information?
- *Gateway Time*: How much processing time does a gateway require to process a packet that is either initializing a flow, or is a part of an existing flow?
- *Gateway Space*: How much storage space does a gateway require to store flow state information?

The required processing time of both CPE and gateway influences the delay and potentially, also the throughput of the mechanism as a whole. The required storage space of both CPE and gateway influences scalability and hardware requirements.

However, the starting point for evaluating IPv4 address sharing mechanisms that can be induced from the detailed property analysis already offers some insight into time and space complexities of the mechanism classes. Let n be the number of concurrent

flows that are established through the IPv4 address sharing mechanism being studied and let m be the number of subscribers that an ISP is providing Internet access to over a single gateway. These two parameters influence the behavior of a mechanism the most. It turns out that time complexity is equal to $\mathcal{O}(1)$ in both CPE and gateway network elements for all mechanism classes. This is because the processing time for each packet is not dependent on the number of concurrent flows, nor on the number of subscribers the ISP has. It is important to note, that the assumption is, that the gateway is not congested. This is reasonable to assume, as ISPs have to plan for such infrastructure, that easily survives the packet bursts during normal operation. However, this is not the case with space complexities, where the following can be observed:

- CGN mechanisms have $\mathcal{O}(n)$ space complexity of the gateway, as there are as many flow state entries required as there are concurrent flows.
- A+P mechanisms and Class 10 mechanisms have $\mathcal{O}(n)$ space complexity of the CPE, as there are as many flow state entries required as there are concurrent flows.
- Class 6 mechanisms have $\mathcal{O}(m)$ space complexity of the gateway, as there are as many flow state entries required as there are customers behind the gateway for whom the ISP is providing Internet access.
- Class 4 and Class 10 mechanisms require an additional operation – referred to as *preamble* – which involves resolving a synthetic IPv6 address from a DNS64 server for IPv4-only Internet end-hosts. During this operation, a number of DNS requests and replies is exchanged between the source end-host, DNS64 server and possibly other DNS servers in the global DNS infrastructure. While it is impossible to accurately predict how much time the preamble takes in a general case an estimation is provided in the following sections that regard the preamble as one of the basic packet operations.

Table 5.1 shows a summary of high-level time and space complexities for all 10 mechanism classes of the extended classification. Even though the time complexities are equal, there are differences in how much processing time different mechanisms require for processing packets. The goal of this thesis is to measure these differences and also to consider them when evaluating the performance of mechanism classes.

Table 5.1

Time and Space Complexities for Mechanism Classes

Class	*CPE Time*	*CPE Space*	*Gateway Time*	*Gateway Space*	*Extra*
Class 1	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	
Class 2	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	
Class 3	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	
Class 4	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	Preamble
Class 5	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
Class 6	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(m)$	
Class 7	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
Class 8	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	
Class 9	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	
Class 10	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Preamble

Note: n is the number of concurrent flows, that are established through the mechanism and m is the number of customers one gateway is serving.

In order to achieve this goal, we first dissect all IPv4 address sharing mechanism classes that are being studied and identify the basic operations on packets that are individually measured and evaluated. Next, we obtain and configure kernel-space implementations of these basic operations, construct a testbed for each one and experimentally evaluate their time requirements for packet processing. Space requirements are not experimentally evaluated, as the space complexity evaluations given above provide a sufficient distinction between classes. Also, $\mathcal{O}(1)$ space complexity means that no state storage is required in the CPE or gateway, so that there are no differences between mechanism classes in this respect. We define a performance index for each basic operation based on the performance evaluations of processing times.

Finally, using the evaluations of basic operations, we theoretically evaluate the performance of all mechanism classes by calculating a cumulative performance index for each mechanism class, based on the performance indices of basic operations and space

complexities shown in Table 5.1.

5.3 Performance Evaluation of Basic Operations

Basic operation is defined as an operation performed by a CPE or a gateway on a packet that either translates or tunnels a packet. As all 10 classes of mechanisms of the classification are studied, the following basic operations may be identified:

- *NAPT44.* Per-flow stateful translation of IP addresses and transport-layer identifiers (ports in UDP and TCP, sequence identifiers in ICMP) in IPv4 packets [20].
- *NAPT64.* Per-flow stateful translation of IP header from IPv4 to IPv6 and vice versa [30], along with IP addresses and transport-layer identifiers (ports in UDP and TCP, sequence identifiers in ICMP).
- *NAPT66.* Per-flow stateful translation of IP addresses and transport-layer identifiers (ports in UDP and TCP, sequence identifiers in ICMP) in IPv4 packets (Chapter 4).
- *NAT64.* Stateless translation of IP addresses in IPv6 packets [117].
- *TUNNEL.* Tunneling of IPv4 packets in IPv6 packets [66].

In the following subsections, the processing time performance of all basic operations is evaluated. A general description of the testbeds and other experimental methods is first provided. Next, two baseline scenarios are evaluated. For these, we select simple IPv4 and IPv6 forwarding of packets through a device without translating or tunneling the packets. Finally, we describe the testbed and discuss the results of each operation.

5.3.1 Experimental Methods

In order to evaluate the processing time performance of basic operations, we construct a two-node testbed and develop a custom tool to perform the processing time measurements.

Time measurement tool: `ptimediff`

In order to measure the time, that the Processor Node spends processing the packet, we develop a purposive program in C programming language, called `ptimediff`. It leverages `libpcap` library to capture a packet traversing incoming and outgoing network interfaces, extract their timestamps and print the time difference to standard output. The source code for `ptimediff` is listed in Appendix A.

As opposed to measuring the one-way delays of packets that are being sent from one host and received by another, measuring processing time within one machine introduces time precision problems. In the course of this study, we found microsecond precision not to be granular enough, as the smallest processing times measured were only a half dozen microseconds. This is why nanosecond precision was implemented into `ptimediff`. In fact, the `libpcap` library was patched to always return to nanoseconds when reading the variable that would usually provide microseconds, as Orosz et al. [227] have shown.

Testbed

The testbed for evaluating the performance of all basic operations consists of the following three devices:

- *Processor Node.* A computer that runs the implementations of basic operations and performs the measurements using the `ptimediff` measurement tool. It has a Pentium 4, 3.0 GHz CPU and 1.5 GB of RAM. It uses two network interfaces, one for incoming packets and another for outgoing packets. The incoming interface is connected to the Generator Node, while the outgoing interface is connected to the layer-3 switch. Debian 7.1 operating system with Linux kernel version 2.6.32 is installed.

- *Generator Node.* A computer that only serves as packet generator and generates packets, that the Processor Node can process. It is a virtual machine, running within VMware Workstation on a personal notebook computer. Generator Node has one network interface which is connected to the incoming interface of the Processor Node. Debian 6.0 operating system with Linux kernel version 2.6.32 is installed.

- *Switch.* A layer-3 switch (Mikrotik RB433AH) that has an IP address(es) configured to be used as destination IP addresses in packets that are generated by the Generator Node. RouterOS operating system with version 4.9 is installed.

On the Generator Node, standard GNU/Linux utility `ping6` is used to generate 10 ICMPv6 type *Echo Request* packets per second for 10 seconds (sending 100 packets in total to the Processor Node for every tested basic operation). As the processing time for the first (and only the first) packet was always a few times higher than the processing times of subsequent packets, it was discarded. The interest of this study is *minimal* processing time, i.e., how much time the Processor Node *minimally requires* in order to process a packet.

5.3.2 IPv4 Baseline

In the IPv4 baseline scenario, we simply measure the time required for the Processor Node to route and forward an IPv4 packet from the incoming to the outgoing interface. No basic packet operation is involved in this scenario.

Testbed

The Generator Node sends IPv4 packets that are destined to the Switch through the Processor Node. Figure 5.1 shows the testbed configuration.

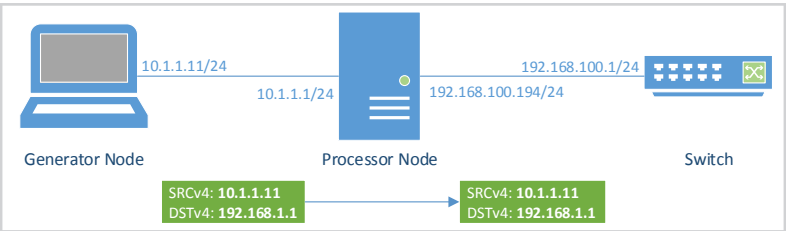


Figure 5.1

Testbed for IPv4 baseline scenario. Processing time of basic IPv4 routing and forwarding is measured.

Results

On average, the Processor Node requires $7.82 \pm 0.58 \mu s$ to route and forward an IPv4 packet from the incoming to the outgoing interface.

5.3.3 IPv6 Baseline

In the IPv6 baseline scenario, we measure the time required for the Processor Node to route and forward an IPv6 packet from the incoming to the outgoing interface. No basic packet operation is involved in this scenario.

Testbed

The Generator Node sends IPv6 packets destined to the Switch through the Processor Node. Figure 5.2 shows the testbed configuration.

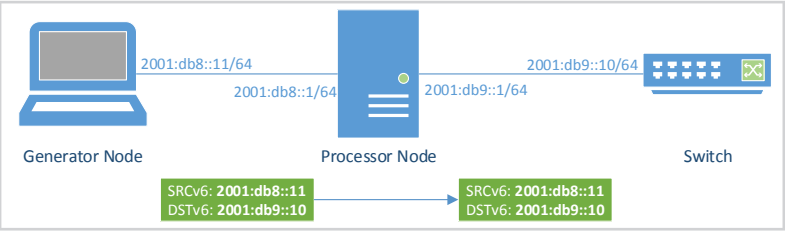


Figure 5.2

Testbed for IPv6 baseline scenario. Processing time of basic IPv6 routing and forwarding is measured.

Results

On average, the Processor Node requires $10.25 \pm 0.80 \mu s$ to route and forward an IPv6 packet from the incoming to the outgoing interface.

5.3.4 Basic Operation: NAPT44

In the NAPT44 basic operation scenario, we measure the time required for the Processor Node to translate the IPv4 address and application-layer identifier (*Identifier* field in ICMP header in our case) of an IPv4 packet.

Testbed

The Generator Node sends IPv4 packets, destined to the Switch through the Processor Node. The Processor Node performs the NAPT44 translation of the packets. For NAPT44, Netfilter [228] is used. Netfilter has been part of the Linux kernel from version 2.4 onwards. It is an open source and very mature (initiated in 1998) and widespread (present in all Linux-based routers) implementation. Netfilter was configured in the Processor Node using the following iptables command:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Figure 5.3 shows the testbed configuration.

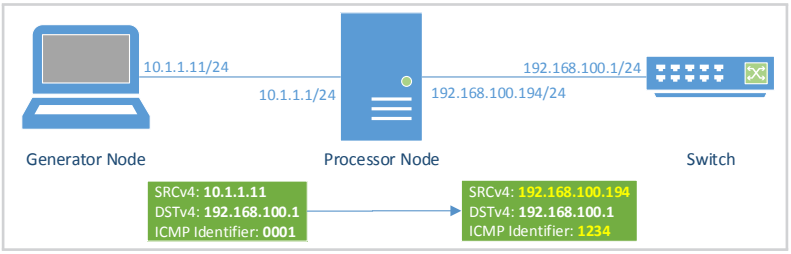


Figure 5.3

Testbed for NAPT44 scenario. Processing time of NAPT44 basic operation is measured.

Results

On average, the Processor Node requires $8.3 \pm 0.46 \mu s$ to perform the NAPT44 translation of an IPv4 packet.

5.3.5 *Basic Operation: NAPT64*

In the NAPT64 basic operation scenario, we measure the time required for the Processor Node to translate the IPv6 header, IPv6 address and application-layer identifier (*Identifier* field in ICMP header in our case) of an IPv6 packet into a IPv4 header, an IPv4 address and another application-layer identifier.

Testbed

The Generator Node sends IPv6 packets destined to the IPv4 address of the Switch through the Processor Node. The Processor Node performs the NAPT64 translation of the packets. For NAPT64, we use Ecdysis [31] (release 2011-11-17), an open source implementation of Stateful NAT64 [30] for Linux and OpenBSD. The code for Ecdysis was written by some of the coauthors of the Proposed Standard RFC document. We use the well-known prefix `64:ff9b::/96` for IPv6-to-IPv4 translation and the IPv4 address of the outgoing interface as the shared IPv4 address. Figure 5.4 shows the testbed configuration.

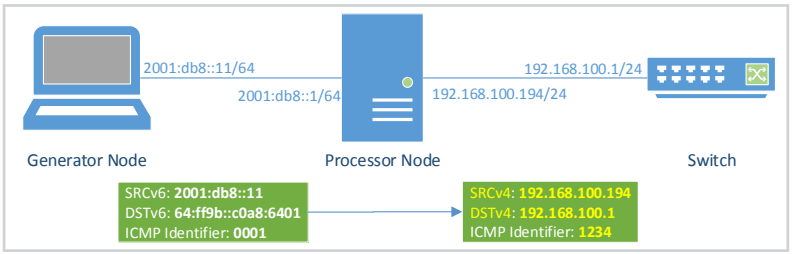


Figure 5.4

Testbed for NAPT64 scenario. The processing time of the NAPT64 basic operation is measured.

Results

On average, the Processor Node requires $19 \pm 1.69\mu s$ to perform the NAPT64 translation of an IPv6 packet into an IPv4 packet.

5.3.6 Basic Operation: NAPT66

In the NAPT66 basic operation scenario, we measure the time required for the Processor Node to translate IPv6 address and application-layer identifier (*Identifier* field in ICMP header in our case) of an IPv6 packet.

Testbed

The Generator Node sends IPv6 packets destined to the IPv6 address of the Switch through the Processor Node. The Processor Node performs the NAPT66 translation of the packets. For NAPT66, we use Terry Moës' implementation, that was documented in his master thesis [229]. The implementation consists of two kernel modules for Linux 2.6.32 kernel, `nf_nat66_core` and `nf_nat66_standalone`, a patch to Linux header files and additional `iptables` modules for `SNAT66` and `DNAT66` targets. The reason an implementation that is not mature and widely adopted has been included in this series of tests is to demonstrate that the concept of translating IPv6 addresses – which is generally considered to be fundamentally wrong and unnecessary by the IETF community – nonetheless has its advantages. As shown in Chapter 4, NAPT66 can be used constructively. As this implementation hooks directly into Netfilter, we believe it is not significantly inefficient because of code immaturity. NAPT66 implementation has been configured running the following `iptables` command:

```
# ip6tables -t nat66 -A POSTROUTING -o eth0 -j SNAT66 --to-range '2001:db9::1'
```

Figure 5.5 shows the testbed configuration.

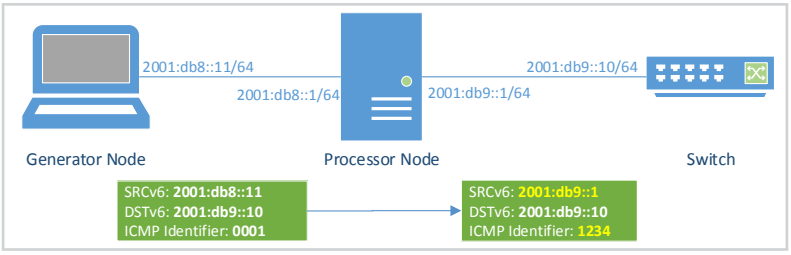


Figure 5.5

Testbed for NAPT66 scenario. Processing time of NAPT66 basic operation is measured.

Results

On average, the Processor Node requires $11.05 \pm 0.48 \mu s$ to perform the NAPT66 translation of an IPv6 packet.

5.3.7 *Basic Operation: NAT64*

In the NAT64 basic operation scenario, we measure the time required for the Processor Node to translate IPv6 header and IPv6 address of an IPv6 packet into an IPv4 header and IPv4 address.

Testbed

The Generator Node sends IPv6 packets destined to the IPv4 address of the Switch through the Processor Node. The Processor Node performs the NAT64 translation of the packets. For NAT64, we use a patched version of Ecdysis [31] (release 2011-11-17), an open source implementation of Stateful NAT64 [30] for Linux and OpenBSD. This was patched so that it does not keep any flow state and does not translate application-layer identifiers. The source code of the patch is listed in Appendix B. We use the well-known prefix `64:ff9b::/96` for IPv6-to-IPv4 translation and the IPv4 address of the outgoing interface as the shared IPv4 address. Figure 5.6 shows the testbed configuration.

Results

On average, the Processor Node requires $16.16 \pm 2.14 \mu s$ to perform the NAT64 translation of an IPv6 packet into an IPv4 packet.

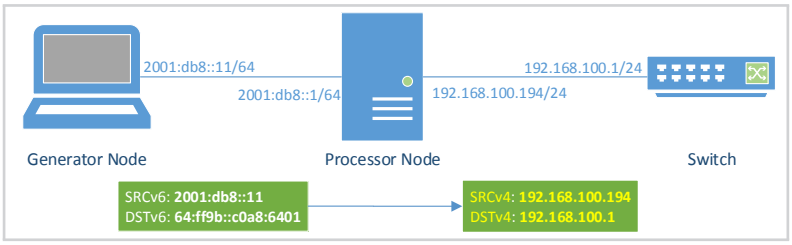


Figure 5.6

Testbed for NAT64 scenario. The processing time of the NAT64 basic operation is measured.

5.3.8 Basic Operation: TUNNEL

In the TUNNEL basic operation scenario, we measure the time required for the Processor Node to encapsulate an IPv4 packet in an IPv6 packet.

Testbed

The Generator Node sends IPv4 packets, destined to an imaginary IPv4 Internet address through the Processor Node. The Processor Node performs the TUNNEL operation, i.e., encapsulation of the packets. For TUNNEL, we use IPv4-in-IPv6 tunneling [66] implementation embedded into Linux kernel and managed by `iproute2` utility. This is also a very widespread and mature implementation of IP in IP tunneling. The tunnel on the Processor Node is configured using the following `iproute2` commands:

```
# ip tunnel add 4over6 mode ipip6 remote 2001:db8::10 local 2001:db8::1
```

Figure 5.7 shows the testbed configuration.

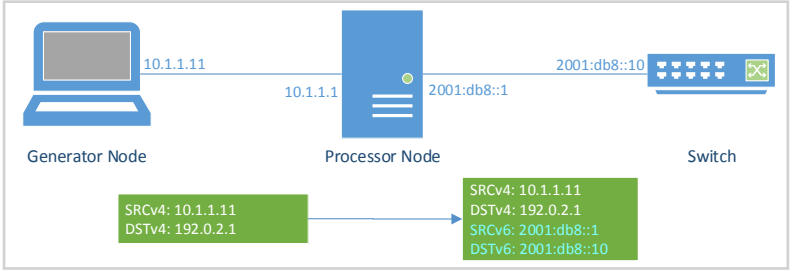


Figure 5.7

Testbed for TUNNEL scenario. Processing time of TUNNEL basic operation is measured.

Results

On average, the Processor Node requires $13.33 \pm 1.17 \mu s$ to perform the encapsulation of an IPv4 packet into an IPv6 packet.

5.3.9 Performance Comparison of Basic Operations

To understand the differences in processing times between basic operations, we provide a cumulative performance graph of processing times in Figure 5.8 and a graph of average processing times for all basic operations in Figure 5.9.

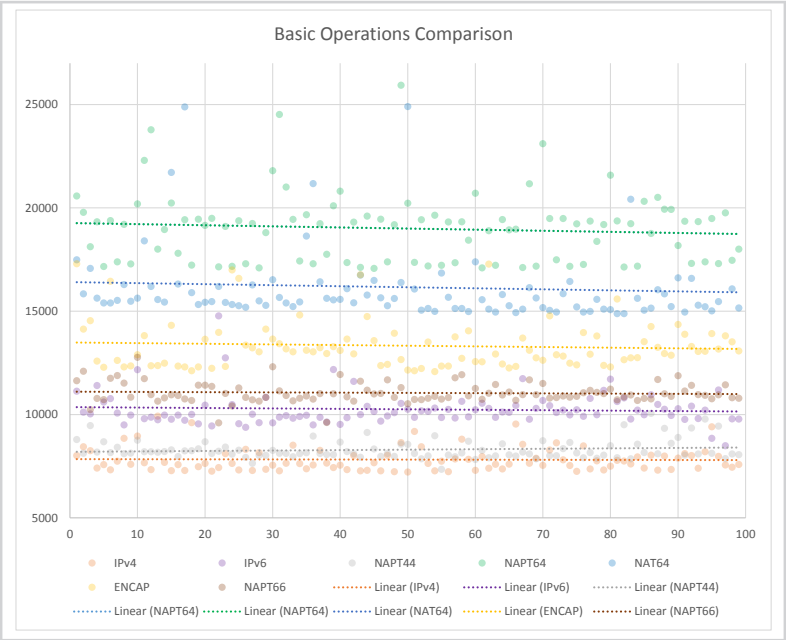


Figure 5.8

Processing times on Processor Node for all scenarios.

The results indicate, that both stateful and stateless IPv6-to-IPv4 translation operations are the most expensive, which is not surprising given that header translation must be performed. It is nevertheless interesting that tunneling also comes with significant cost as compared to stateful NAPT44 and NAPT66 translations. IPv6 processing itself takes more time and IPv4-only performance is in any case superior.

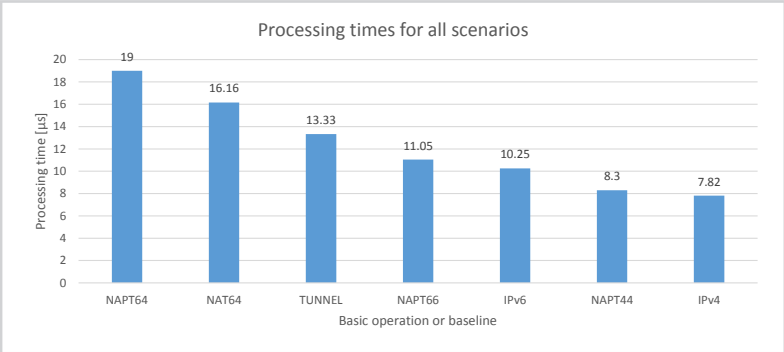


Figure 5.9

Average processing times on Processor Node for all scenarios.

To prepare grounds for theoretical performance evaluation of mechanism classes, we first derive performance indices for each basic operation, based on the results from the measurements. Performance index P of an operation is computed as follows:

- If IPv4 packets enter the incoming interface and IPv4 packets leave the outgoing interface, the performance index is calculated as $P = \frac{p}{p_{IPv4}}$, where p_{IPv4} is the processing time of IPv4 baseline scenario and p is the processing time of this operation.
- If IPv6 packets enter the incoming interface and IPv4 packets leave the outgoing interface or vice versa, the performance index is calculated as $P = \frac{p}{\frac{p_{IPv4}+p_{IPv6}}{2}}$, where p_{IPv4} is the processing time of the IPv4 baseline scenario, p_{IPv6} is the processing time of the IPv6 baseline scenario and p is the processing time of this operation.
- If IPv6 packets enter the incoming interface and IPv6 packets leave the outgoing interface, the performance index is calculated as $P = \frac{p}{p_{IPv6}}$, where p_{IPv6} is the processing time of the IPv6 baseline scenario and p is the processing time of this operation.

The rationale for this methodology is the following: The objective is to calculate the performance index as a difference to the "no-operation" (baseline) scenario. However, as IPv6 routing and forwarding is itself more expensive than IPv4 routing and forwarding, this difference in the index computation must also be considered. For operations

where a packet is translated from one address family to another, the average of both IPv4 and IPv6 baseline scenarios is taken into account.

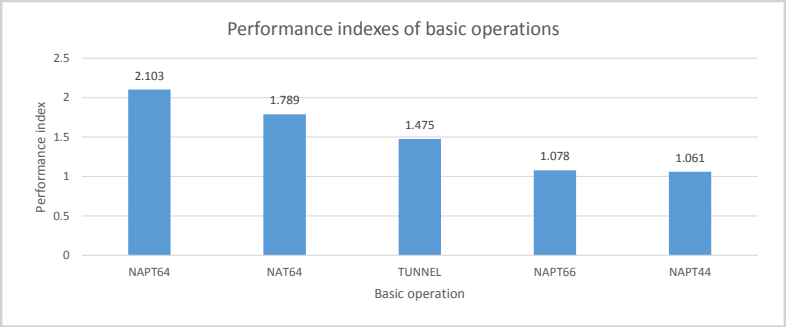


Figure 5.10

Performance indices for all basic operations.

Using the methodology above, the performance indices for basic operations shown in Figure 5.10 are calculated. These indices are used in subsequent sections to calculate theoretical performance evaluations of the mechanism classes.

5.4 *Theoretical Performance Evaluation of Mechanism Classes*

The goal of theoretical performance evaluation of mechanism classes is to be able to compare the classes regarding their performance.

5.4.1 *Methodology*

Performance index of a mechanism class is based on the evaluations of processing times and space complexities, which can be evaluated from mechanism properties. In practice, space requirements are much more important and critical than time processing requirements. This is because storing flow state in devices requires significant amount of device memory. This becomes obvious if we recall that CPE Time and Gateway Time complexity of all mechanism classes is $\mathcal{O}(1)$. Consequently, space complexities influence performance to a greater extent.

The performance index of a mechanism class is computed from CPE Time, CPE Space, Gateway Time and Gateway Space metrics using the following methodology (the definitions of n and m remain the same as those defined in Section 5.2):

- *CPE Time* of a mechanism class is the sum of performance indices of all basic operations that are performed by the CPE.
- *CPE Space* of a mechanism class is:
  - *0*, if CPE Space complexity is $\mathcal{O}(1)$,
  - *100*, if CPE Space complexity is $\mathcal{O}(n)$. This is because for CPE, space requirements are more important in order of magnitude than are time requirements.
- *Gateway Time* of a mechanism class is the sum of performance indices of all basic operations, that are performed by the gateway, multiplied by *1,000*. This is because time and space requirements of the gateway are at least an order of magnitude more important than are time and space requirements of the CPE.
- *Gateway Space* of a mechanism class is:
  - *0*, if Gateway Space complexity is $\mathcal{O}(1)$,
  - *1,000*, if Gateway Space complexity is $\mathcal{O}(m)$. The number of subscribers does not influence performance of per-allocation stateful mechanism classes as critically as does the number of concurrent flows influences per-flow stateful mechanism classes.
  - *10,000*, if Gateway Space complexity is $\mathcal{O}(n)$. The number of concurrent flows influences per-flow stateful mechanism classes most significantly.
- *Performance Index* of a mechanism class is the sum of CPE Time, CPE Space, Gateway Time and Gateway Space mechanism metrics above. If the sum is a decimal number, it is rounded to the nearest integer.

5.4.2 Results

Using the methodology above, the performance indices for all mechanism classes are calculated. These are listed in Table 5.2.

Two visual comparisons of mechanism classes' performance are provided in Figure 5.11 and Figure 5.12. Figure 5.11 one visually demonstrates the differences among stateless A+P mechanism classes while Figure 5.12 differentiates CGN mechanism classes.

Table 5.2

Performance indices and their components for all IPv4 address sharing mechanisms.

Class	*CPE Basic Operations*	*Gateway Basic Operations*	*CPE Time*	*Gateway Space*	*Gateway Time*	*Gateway Space*	*Perf. Index*
Class 5	NAPT44, TUNNEL	TUNNEL	2.536	100	1475	0	*1578*
Class 7	NAPT44, NAT64	NAT64	2.850	100	1789	0	*1892*
Class 8	NAPT44, NAT64	NAT64	2.850	100	1789	0	*1892*
Class 10	NAPT66	NAT64	1.078	100	1789	0	*1890*
Class 6	NAPT44, TUNNEL	TUNNEL	2.536	100	1475	1000	*2578*
Class 2	TUNNEL	NAPT44	1.475	0	1061	10000	*11062*
Class 3	TUNNEL	NAPT44	1.475	0	1061	10000	*11062*
Class 1	NAPT44	NAPT44	1.061	100	1061	10000	*11162*
Class 4	IPv6 baseline	NAPT64	1	0	2103	10000	*12104*
Class 9	NAT64	NAPT64	1.789	0	2103	10000	*12105*

Finally, we also present a comparison of all mechanisms in Figure 5.13. It is imperative to note, that A+P and CGN mechanisms are not directly comparable due to the weighting system that is used in deriving the Performance Index.

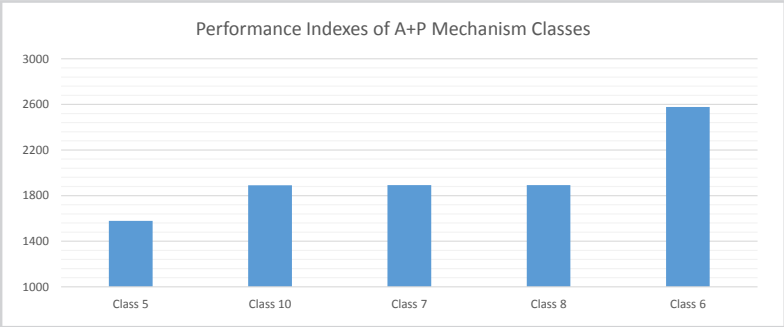


Figure 5.11

Performance comparison of A+P IPv4 address sharing mechanism classes.

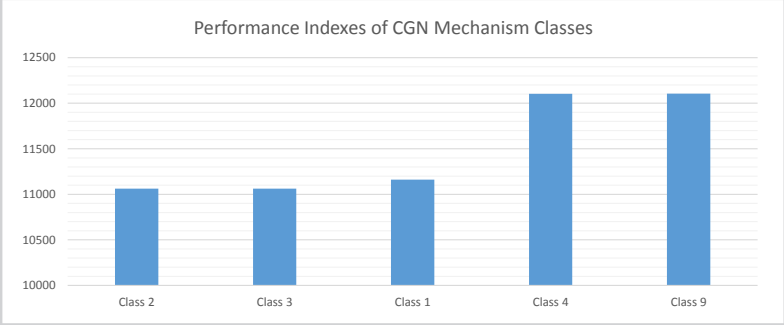


Figure 5.12

Performance comparison of CGN IPv4 address sharing mechanism classes.

5.4.3 Discussion

It is evident that the performance analysis framework segmented the 10 mechanism classes into 3 segments: CGN mechanisms (Classes 1, 2, 4, 5, and 9), stateless A+P mechanisms (Classes 5, 7, 8 and 10) and one per-flow stateful A+P mechanism (Class 6). This is not surprising as we have been aware of conceptual differences that lead to such performance differentiation in advance. However, significant differences within the three segments may also be observed. CGN mechanisms leveraging the NAPT64 translation are significantly more expensive than others. On the other hand, this research has demonstrated that A+P mechanisms which use tunneling as an access network traversal method (Class 5) are more efficient than those which leverage the double translation traversal method (Classes 7 and 8).

The proposed AP64 mechanism that falls into the Class 10 mechanisms performs

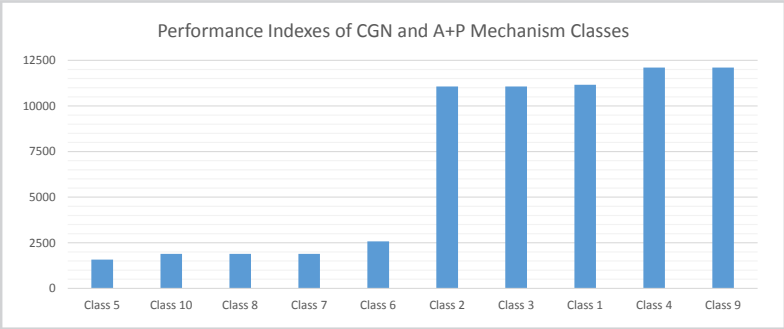


Figure 5.13

Performance comparison of all CGN and A+P IPv4 address sharing mechanism classes.

similarly to Class 7 and 8 mechanisms. Therefore, it is proven that this method is significantly more efficient than competing methods, i.e., Class 4 mechanisms.

6

Conclusion

This thesis has tackled the problem of IPv6 transition. First, we overviewed historical aspects of IPv6 transition and analysed the current state of the IPv6 transition. We established that by April 2013, the transition to IPv6 has hardly begun. Only 1.13% of Google users access Google's services over IPv6 while only 0.28% and 0.50% of all traffic in the two major Internet exchange points is IPv6. However, 15.76% autonomous systems are IPv6 ready while 5.27%, 10.80% and 24% of web servers in the Alexa Top 1M, 1000 and 500 lists, respectively, are available over IPv6. Next, we identified two major groups of IPv6 transition mechanisms: IPv6 deployment mechanisms and IPv4 address sharing mechanisms. We systematically review IPv6 deployment mechanisms by providing an overview, IETF timeline, penetration details, description of operation and various other notes for each mechanism. We proceed to a further analysis of IPv4 address sharing mechanisms because these are now significantly more relevant for IPv6 transition than are IPv6 deployment mechanisms.

We presented a novel classification of IPv4 address sharing mechanisms, which was then used to analyse their various properties. The goal of this research was to assess the tradeoffs involved when choosing a specific mechanism in a comprehensive but consistent way. First, we defined an IPv4 address sharing mechanism space of five dimensions. Next, we systematically reviewed all mechanisms that have been proposed to date, classifying them into nine distinct classes using the presented taxonomy. Moreover, we analysed the issues related to the properties of mechanisms along the dimensions of the proposed classification. Finally, we summarized the property analysis into a qualitative tradeoff analysis, focusing on the trading benefits of specific properties along the dimensions of the classification. As a result, it is demonstrated that the CGN versus A+P dilemma is not equal to the "NAPT-in-the-CPE" versus "NAPT-in-the-core" dilemma, which is believed to be a common misconception. Another important insight is that address translation and port-restriction may be regarded as two distinct functions that can be performed at different locations independently. It is also important to recognize that address family translation in the context of the traversal method is unrelated to classical NAPT translation. Finally, it is demonstrated that the only (existing) mechanism that is really IPv6-encouraging is Stateful NAT64 (Class 4). This is because Stateful NAT64 requires that edge networks are IPv6-only, which is what we are eventually aiming to achieve on the Internet. All other mechanisms require the provisioning of IPv4 to some extent into the edge networks, which means that substantial effort will be required in order to eventually eliminate these

deployments.

Using the five dimensions of the classification as a design space for IPv4 address sharing mechanisms, we identified a new set of properties that would comprise a new useful mechanism – AP64. We defined this mechanism, its architecture and functions, propose a new class in the classification (Class 10) and explain all the technical details that are needed for implementing it. We also argue its uniqueness and usefulness for the anticipated forthcoming years of IPv6 transition. Requiring IPv6-only edge networks, makes AP64 highly IPv6-encouraging, as Class 4 mechanisms are. Being an A+P mechanism (as opposed to Class 4 being CGN), AP64 is significantly more scalable because no per-flow or per-allocation state needs to be maintained in the gateway. Finally, using normal routing as the traversal method, AP64 removes all concerns about MTU and fragmentation issues, packet inspection issues etc. No special treatment of packets is required.

Finally, we presented a framework for evaluating the performance of IPv4 address sharing mechanisms. After failing two major performance evaluation experiments, it was learned that it is very difficult to assess and compare the performance of mechanisms "as a whole", simply because mechanisms are complex and their implementations are difficult to obtain, or are often implemented on different platforms and in different ways. Because of this, we decomposed the mechanism classes into five basic packet operations (NAPT44, NAPT64, NAPT66, NAT64 and TUNNEL) and evaluated their performance instead. The scientific approach was to measure the processing time of each basic operation by using "atomic", kernel-space and implementations that are as mature as possible. We established that IPv6-to-IPv4 translation is the most expensive, as NAPT64 and NAT64 operations required 19 and 16.16μs for processing a single packet, while TUNNEL required 13.33μs and NAPT44 and NAPT66 required 8.3μs and 11.05μs. Comparing these results to two baseline scenarios, we calculated the performance indices for all five basic operations. Using these indexes and space complexity evaluations based on mechanism class properties from the property analysis, allows for the to theoretical evaluation and comparison of the performance of all mechanism classes of the presented classification. We first confirmed that A+P mechanisms are much more efficient than are CGN mechanisms, which also proves that Class 10 provides a much better performance capacity than do competing mechanisms, such as those defined in Class 4. We also demonstrated that CGN mechanisms leveraging the NAPT64 translation are significantly more expensive than are other CGN

mechanisms. Finally, we have shown that A+P mechanisms, which use tunneling as an access network traversal method (Class 5), are more efficient than those, which use double translation.

During the course of this work, multiple challenges have been faced. First, one could argue that the design space which was constructed using the five proposed dimensions remains incomplete. We argue its completeness by going through the dimensions one by one and provide justification as to why there are no other possible values in the axis of any dimension, that were not considered during the course of this study. We believe this is the best that can be done at the moment, given that it is very difficult to predict future achievements and inventions. However, we believe that even if such an achievement invalidates the proposed design space, it would not be too difficult to "fix" the dimension in question by adding a new value and consequently, by extending the design space accordingly.

We also realize that although a theoretical framework for performance evaluation is being proposed in this study, experiments on implementations of basic operations have still been performed. Implementations are never optimal and the best that can be done is to assure that they are most comparable, i.e., that they all run in kernel-space, on the same operating system platform and that they are as mature as possible.

Finally, the usefulness of the AP64 mechanism that has been proposed in this thesis can be disputed by stating that it does not support IPv4-only applications, such as Skype. This is correct and is also true for any other mechanism that requires an IPv6-only edge network. Without having IPv4 addressing configured, IPv4-only applications like Skype simply fail. This is why it is suggested that AP64 could be used in a few years from now, when all major applications would be at least IPv6-compatible, if not IPv6-enabled. Also, as there are use cases for Stateful NAT64 (Class 4) today, there are certainly also use cases for AP64.

The following possible future work may be envisioned on the basis of the research presented in this thesis. First, we believe the remaining "empty boxes" of the proposed design space should be investigated, as they can lead to new useful mechanisms. The space for new mechanisms we have opened with our classification, has not been envisioned before. This was also the path that led to the discovery of AP64. However, proposing a mechanism in a publication is not enough to receive critical feedback from the standardization community, i.e., the IETF. Publishing AP64 in the form of an Internet Draft first and then presenting and defending it at future IETF meetings

could be successful, however, this is beyond the capacity of academic research. One way to address the "Skype problem" would be to consider a platform-dependent agent that is similar to what Class 9 mechanisms allow for. However, this would invalidate the initial assumption which states that end-hosts must not be modified in any way. However, such solutions may nevertheless be viable provided there is sufficient incentive from the ISP side. Another area for future research could be improving (upon) the performance analysis framework by finding more implementations of basic operations and using the best performing one for the framework.

Source Code: ptimediff

A

```
#include <stdio.h> #include <pcap.h> #include <pthread.h> #include <stdlib.h>
#include <math.h> #include <unistd.h>

/* Configuration */

int n = 100; int promisc = 0; int debug = 0; char *filter;

/* Common variables */

long now = 0; #define NUMTHREADS 2

/* Init */

pthread_t thread[NUMTHREADS]; pthread_mutex_t mutex;

void got_packet(u_char *args, const struct pcap_pkthdr *header,
    const u_char *packet);

void *listen(void *arg) {
    char *dev, errbuf[PCAP_ERRBUF_SIZE];
    pcap_t *session;
    struct bpf_program fp;
    bpf_u_int32 mask;
    bpf_u_int32 net;

    dev = (char *)arg;

    if (debug) printf("[%d]: Init\n", (int)pthread_self());

    pthread_mutex_lock(&mutex);

    if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
        fprintf(stderr, "Can't get netmask for device %s\n", dev);
        net = 0;
        mask = 0;
    }

    if (debug) printf("[%d]: Netmask\n", (unsigned int)pthread_self());
```

```
    session = pcap_open_live(dev, BUFSIZ, promisc, 0, errbuf);

    if (session == NULL) {
        fprintf(stderr, "Couldn't open device %s: %s\n", dev, errbuf);
        pthread_exit((void *) 2);
    }

    if (debug) printf("[%d]: Session\n", (unsigned int)pthread_self());

    if (pcap_compile(session, &fp, filter, 0, net) == -1) {
        fprintf(stderr, "Couldn't parse filter %s: %s\n", filter,
            pcap_geterr(session));
        pthread_exit((void *) 2);
    }

    if (debug) printf("[%d]: Compile filter\n",
        (unsigned int)pthread_self());

    if (pcap_setfilter(session, &fp) == -1) {
        fprintf(stderr, "Couldn't install filter %s: %s\n", filter,
            pcap_geterr(session));
        pthread_exit((void *) 2);
    }

    if (debug) printf("[%d]: Set filter\n",
        (unsigned int)pthread_self());

    pthread_mutex_unlock(&mutex);

    printf("[%d]: Ready.\n", (unsigned int)pthread_self());

    pcap_loop(session, n, got_packet, NULL);
    if (debug) printf("[%d]: Finish looping\n",
        (unsigned int)pthread_self());
    pcap_close(session);
    if (debug) printf("[%d]: Finish session\n",
        (unsigned int)pthread_self());

    pthread_exit((void *) 0);
}
```

```
void got_packet(u_char *args, const struct pcap_pkthdr *header,
    const u_char *packet) {
    long then = (long)header->ts.tv_usec + (long)header->ts.tv_sec *
        (long)1000000000;

    if (now == 0) {
        now = then;
    } else {
        printf("%ld\n", labs(then - now));
        now = 0;
    }
    if (DEBUG) printf("[%d]: ... packet processed\n",
        (unsigned int)pthread_self());
}

int main(int argc, char *argv[]) {
    void *status;

    char *dev1 = "eth1";
    char *dev2 = "eth2";

    pthread_mutex_init(&mutex, NULL);

    pthread_create(&thread[0], NULL, listen, (void *)dev1);
    pthread_create(&thread[1], NULL, listen, (void *)dev2);

    pthread_join(thread[0], &status);
    pthread_join(thread[1], &status);

    pthread_mutex_destroy(&mutex);
    pthread_exit(NULL);
}
```

Source Code: patch for stateless Ecdysis

B

```
diff -rupN ecdysis/nf_nat64_main.c ap64/nf_nat64_main.c
--- ecdysis/nf_nat64_main.c     2013-07-19 06:27:38.000000000 +0200
+++ ap64/nf_nat64_main.c        2013-07-19 06:33:20.000000000 +0200
@@ -87,8 +87,7 @@ struct net_device *nat64_dev;
  * \return     The payload length of the translated packet.
  */
 static int
-nat64_input_ipv6_recur(int recur, struct ipv6hdr *ip6, int len,
-               struct nat64_session **s)
+nat64_input_ipv6_recur(int recur, struct ipv6hdr *ip6, int len)
 {
        struct ipv6_opt_hdr     *ip6e;
        struct udphdr           *uh;
@@ -124,8 +123,7 @@ next_header:
        case IPPROTO_UDP:
                uh = (struct udphdr *)((char *)ip6 + hlen);

-               if ((len -= sizeof(*uh)) < 0 ||
-                               !(*s = nat64_ipv6_udp_session(ip6, \
                                    uh, recur)))
+               if ((len -= sizeof(*uh)) < 0)
                        return 0;

                return len + sizeof(*uh);
@@ -133,8 +131,7 @@ next_header:
        case IPPROTO_TCP:
                th = (struct tcphdr *)((char *)ip6 + hlen);

-               if ((len -= sizeof(*th)) < 0 ||
-                               !(*s = nat64_ipv6_tcp_session(ip6, \
                                    th, recur)))
+               if ((len -= sizeof(*th)) < 0)
                        return 0;

                return len + sizeof(*th);
@@ -146,12 +143,10 @@ next_header:
                        return 0;

                if (icmp6->icmp6_type & ICMPV6_INFOMSG_MASK) {
```

```
-                       if (!(*s = nat64_ipv6_icmp_session(ip6, icmp6, \
                            recur)))
-                               return 0;
                        return len + 8;
                } else {
                        int size = nat64_input_ipv6_recur(recur + 1,
-                                       (struct ipv6hdr *)(icmp6 + 1), \
                                            len, s);
+                                       (struct ipv6hdr *)(icmp6 + 1), \
                                            len);
                        return size ? size + sizeof(struct iphdr) + 8 : 0;
                }

@@ -249,17 +244,8 @@ adjust_checksum_ipv4_to_ipv6(uint16_t *s
                        (uint16_t *)(&ip6->saddr + 2), udp);
 }

-
-static void
-checksum_change(uint16_t *sum, uint16_t *x, uint16_t new, int udp)
-{
-        checksum_adjust(sum, *x, new, udp);
-        *x = new;
-}
-
 static struct iphdr *
-nat64_xlate_ipv6_to_ipv4(struct ipv6hdr *ip6, struct iphdr *ip4, \
    int plen,
-               struct nat64_session *s, int recur)
+nat64_xlate_ipv6_to_ipv4(struct ipv6hdr *ip6, struct iphdr *ip4, \
    int plen, int recur)
 {
        struct ipv6_opt_hdr *ip6e;
        struct udphdr *uh;
@@ -285,26 +271,22 @@ nat64_xlate_ipv6_to_ipv4(struct ipv6hdr
        }

        if (recur % 2 == 0) {
-               ip4->saddr = s->s_binding->b_saddr4.s_addr;
+               in4_pton(nat64_ipv4_addr, -1, (u8*)&(ip4->saddr), '\x0', \
```

```
                    NULL);
                *(struct in_addr*)&ip4->daddr = nat64_extract(&ip6->daddr);
        } else {
                *(struct in_addr*)&ip4->saddr = nat64_extract(&ip6->saddr);
-               ip4->daddr = s->s_binding->b_saddr4.s_addr;
+               in4_pton(nat64_ipv4_addr, -1, (u8*)&(ip4->daddr), '\x0', \
                    NULL);
        }

        switch (ip4->protocol) {
        case IPPROTO_UDP:
                uh = ip_data(ip4);
                memcpy(uh, ip6e, plen);
-               checksum_change(&uh->check, recur % 2 ? &uh->dest :
-                               &uh->source, s->s_binding->b_sport4, 1);
                adjust_checksum_ipv6_to_ipv4(&uh->check, ip6, ip4, 1);
                break;
        case IPPROTO_TCP:
                th = ip_data(ip4);
                memcpy(th, ip6e, plen);
-               checksum_change(&th->check, recur % 2 ? &th->dest :
-                               &th->source, s->s_binding->b_sport4, 0);
                adjust_checksum_ipv6_to_ipv4(&th->check, ip6, ip4, 0);
                break;
        case IPPROTO_ICMPV6:
@@ -366,9 +348,9 @@ nat64_xlate_ipv6_to_ipv4(struct ipv6hdr
                        nat64_xlate_ipv6_to_ipv4(
                                (struct ipv6hdr *)((char *)ip6e + 8),
                                (struct iphdr *)(ih + 1),
-                               plen - ((char *)ip6e + 8 - (char *)ip6), \
-                                   s,
+                               plen - ((char *)ip6e + 8 - (char *)ip6),
                                        recur + 1);
-
+
                }
                ih->checksum = 0;
                ih->checksum = ip_compute_csum(ih, plen);
@@ -435,7 +417,6 @@ nat64_input_ipv6(struct sk_buff *skb, st
        struct iphdr *ip4;
```

```
        struct ipv6hdr *ip6;
        struct in6_addr *daddr;
-       struct nat64_session *s;
        struct sk_buff *nskb;

        int len = skb->len;
@@ -451,12 +432,8 @@ nat64_input_ipv6(struct sk_buff *skb, st
        if (memcmp(daddr, nat64_config_prefix(), \
            nat64_config_prefix_len()/8)) {
                return -1;
        }
-
-       /* Check for expired sessions */
-       nat64_expire();

-       /* Find the corresponding session. Create one if none exist. */
-       if (!(plen = nat64_input_ipv6_recur(0, ip6, len, &s)))
+       if (!(plen = nat64_input_ipv6_recur(0, ip6, len)))
                return -1;

        /* Allocate a new sk_buff */
@@ -470,12 +447,11 @@ nat64_input_ipv6(struct sk_buff *skb, st
        }

        /* Translate the packet. */
-       if (!nat64_xlate_ipv6_to_ipv4(ip6, ip4, plen, s, 0)) {
+       if (!nat64_xlate_ipv6_to_ipv4(ip6, ip4, plen, 0)) {
                kfree_skb(nskb);
                return -1;
        }

-
        nat64_output_ipv4(nskb);

        /* Free the incoming packet */
@@ -484,8 +460,7 @@ nat64_input_ipv6(struct sk_buff *skb, st
 }

 static int
-nat64_input_ipv4_recur(int recur, struct iphdr *ip4, int len,
```

```
-               struct nat64_session **s)
+nat64_input_ipv4_recur(int recur, struct iphdr *ip4, int len)
 {
        struct udphdr           *uh;
        struct tcphdr           *th;
@@ -499,8 +474,7 @@ nat64_input_ipv4_recur(int recur, struct
        case IPPROTO_UDP:
                uh = (struct udphdr *)((char *)ip4 + hlen);

-               if ((len -= sizeof(*uh)) < 0 ||
-                               !(*s = nat64_ipv4_udp_session(ip4, uh, \
-                                   recur)))
+               if ((len -= sizeof(*uh)) < 0)
                        return 0;

                return len + sizeof(*uh);
@@ -508,8 +482,7 @@ nat64_input_ipv4_recur(int recur, struct
        case IPPROTO_TCP:
                th = (struct tcphdr *)((char *)ip4 + hlen);

-               if((len -= sizeof(*th)) < 0 ||
-                               !(*s = nat64_ipv4_tcp_session(ip4, th, \
-                                   recur)))
+               if((len -= sizeof(*th)) < 0)
                        return 0;

                return len + sizeof(*th);
@@ -521,13 +494,11 @@ nat64_input_ipv4_recur(int recur, struct
                        return 0;

                if (ICMP_INFOTYPE(icmp->type)) {
-                       if (!(*s = nat64_ipv4_icmp_session(ip4, icmp, \
-                           recur)))
-                               return 0;
                        return len + sizeof(struct icmp6hdr);
                } else {
                        len = nat64_input_ipv4_recur(recur + 1,
                                        (struct iphdr*)(icmp + 1),
-                                       len - sizeof(*ip4), s);
+                                       len - sizeof(*ip4));
```

```
                        return len ? len + sizeof(struct ipv6hdr) +
                                sizeof(struct icmp6hdr) : 0;
                }
@@ -544,7 +515,7 @@ nat64_input_ipv4_recur(int recur, struct

 static struct ipv6hdr *
 nat64_xlate_ipv4_to_ipv6(struct iphdr *ip4, struct ipv6hdr *ip6, \
    int plen,
-               struct nat64_session *s, int recur)
+               int recur)
 {
        struct udphdr           *uh;
        struct tcphdr           *th;
@@ -555,16 +526,16 @@ nat64_xlate_ipv4_to_ipv6(struct iphdr *i
        ip6->flow_lbl[0] = 0;
        ip6->flow_lbl[1] = 0;
        ip6->flow_lbl[2] = 0;
-
+
        ip6->payload_len = htons(plen);
        ip6->nexthdr  = ip4->protocol;
        ip6->hop_limit = ip4->ttl;

        if (recur % 2 == 0) {
                nat64_embed(*(struct in_addr *)&ip4->saddr, &ip6->saddr);
-               ip6->daddr = s->s_binding->b_saddr6;
+               in6_pton("2001:db8:12:3400::c000:0212:0034", -1, \
                (u8*)&(ip6->daddr), '\x0', NULL);
        } else {
-               ip6->saddr = s->s_binding->b_saddr6;
+               in6_pton("2001:db8:12:3400::c000:0212:0034", -1, \
                (u8*)&(ip6->saddr), '\x0', NULL);
                nat64_embed(*(struct in_addr *)&ip4->daddr, &ip6->daddr);
        }

@@ -572,11 +543,9 @@ nat64_xlate_ipv4_to_ipv6(struct iphdr *i
        case IPPROTO_UDP:
                uh = (struct udphdr *)(ip6 + 1);
                memcpy(uh, ip_data(ip4), plen);
-               checksum_change(&uh->check, recur % 2 ? &uh->source :
```

```
-                               &uh->dest, s->s_binding->b_sport6, 1);
                if(uh->check) {
                        adjust_checksum_ipv4_to_ipv6(&uh->check, ip4, \
                            ip6, 1);
-               } else {
+               } else {
                        uh->check = csum_ipv6_magic(
                                        &ip6->saddr, &ip6->daddr,
                                        plen, IPPROTO_UDP,
@@ -586,8 +555,6 @@ nat64_xlate_ipv4_to_ipv6(struct iphdr *i
        case IPPROTO_TCP:
                th = (struct tcphdr *)(ip6 + 1);
                memcpy(th, ip_data(ip4), plen);
-               checksum_change(&th->check, recur % 2 ? &th->source :
-                               &th->dest, s->s_binding->b_sport6, 0);
                adjust_checksum_ipv4_to_ipv6(&th->check, ip4, ip6, 0);
                break;
        case IPPROTO_ICMP:
@@ -659,7 +626,7 @@ nat64_xlate_ipv4_to_ipv6(struct iphdr *i
                        }
                        nat64_xlate_ipv4_to_ipv6(ip_data(ip4) + 8,
                                        (struct ipv6hdr *)(icmp6 + 1),
-                                       plen - sizeof(*icmp6) - \
+                                       plen - sizeof(*icmp6) - \
                                            sizeof(*ip6), s,
+                                       plen - sizeof(*icmp6) - \
                                            sizeof(*ip6),
                                        recur + 1);
                }
                icmp6->icmp6_cksum = 0;
@@ -682,7 +649,6 @@ nat64_input_ipv4(unsigned int hooknum,
                const struct net_device *out,
                int (*okfn)(struct sk_buff *))
 {
-       struct nat64_session *s;
        struct iphdr *ip4 = ip_hdr(skb);
        struct ipv6hdr *ip6;
        struct sk_buff *nskb;
@@ -694,13 +660,9 @@ nat64_input_ipv4(unsigned int hooknum,

        /* XXX What about fragments ? */
```

```
-       /* Check for expired sessions */
-       nat64_expire();
-
-       /* Find the corresponding session. Create one if none exist. */
-       if(!(plen = nat64_input_ipv4_recur(0, ip_hdr(skb), len, &s)))
+       if(!(plen = nat64_input_ipv4_recur(0, ip_hdr(skb), len)))
                return NF_ACCEPT;
-
+
        /* Allocate a new sk_buff */
        nskb = nat64_alloc_skb(sizeof(struct ipv6hdr), plen);

@@ -712,8 +674,7 @@ nat64_input_ipv4(unsigned int hooknum,

        ip6 = ipv6_hdr(nskb);

-       /* Translate the packet. */
-       if (!nat64_xlate_ipv4_to_ipv6(ip4, ip6, plen, s, 0)) {
+       if (!nat64_xlate_ipv4_to_ipv6(ip4, ip6, plen, 0)) {
                kfree_skb(nskb);
                return NF_DROP;
        }
@@ -939,5 +900,3 @@ static void __exit nat64_fini(void)

 module_init(nat64_init);
 module_exit(nat64_fini);
-
-
```

Razširjeni povzetek

C

C.1 Uvod

V poznih osemdesetih letih, samo nekaj let po začetku množične uporabe internetnega protokola, je postalo jasno, da bo potrebno začeti iskati načine za ohranjanje internetnih naslovov. Število povezanih naprav je preseglo 100.000 in leta 1992, kmalu po uvedbi brezrazrednega usmerjanja med domenami [5, 6] je IETF formaliziral proces iskanja novih rešitev z organizacijo začasne skupine IPng, ki naj bi se ukvarjala z internetnim protokolom naslednje generacije. Rezultat skupine je bil dokument [8], ki je analiziral vse takrat pomembne predlagane pristope. Nekateri od teh so bili predhodniki novega internetnega protokola IPv6, objavljenega v seriji dokumentov RFC, začenši s dokumentom RFC 1883 [12] leta 1996.

Protokola IPv4 in IPv6 sta si tako različna, da naprave, ki podpirajo samo IPv4, in naprave, ki podpirajo samo IPv6, ne morejo komunicirati med seboj. To pomeni, da bo potreben prehod interneta s protokola IPv4 na IPv6. Ustvarjalci protokola IPv6 so si predstavljali, da bodo vse naprave sčasoma pridobile tudi naslov IPv6, tako da bo protokol IPv4 počasi zapadel v pozabo. Žal se to (še) ni zgodilo, saj se je izkazalo, da je izjemno težko zamenjati nekaj, kar deluje (dovolj) dobro. Nenazadnje je naslovov IPv4 v resnici zmanjkalo, kar pomeni, da je zadnjih nekaj let motivacija za izvedbo prehoda bistveno večja kot 15 let nazaj.

Motivacija te disertacije je naslednja. Od rojstva protokola IPv6 je bilo predlaganih že mnogo prehodnih mehanizmov. Danes se kaže potreba po sistematičnem in popolnem pregledu teh tehnologij, ki je prvi korak k identificiranju prostora možnih mehanizmov. Nadalje, klasifikacija uveljavljenih mehanizmov in novih predlogov mehanizmov bi olajšala nadaljnje raziskovalno delo na področju prehodnih mehanizmov. Poleg tega potrebujemo orodja za praktike, s katerimi bodo lažje sprejemali ozaveščene odločitve o uvedbi tega ali onega mehanizma v lastna omrežja. Nenazadnje, identificiranje prostora mehanizmov bo omogočilo raziskovalcem razvoj novih potencialno uporabnih mehanizmov.

C.1.1 Pregled disertacije

Začnemo s pregledom prehodnih mehanizmov IPv6, kjer najprej predstavimo zgodovinski okvir prehoda interneta na protokol IPv6. To nam omogoča, da bolje razumemo motivacijo za razvoj posameznih mehanizmov. Nadaljujemo s pregledom mehanizmov za uvedbo IPv6 glede na klasično klasifikacijo (tuneliranje, prevajanje, dvojni sklad).

V naslednjem poglavju se osredinimo na mehanizme za deljenje naslovov IPv4, ki ostanejo osrednja tema disertacije. Predstavimo klasifikacijo mehanizmov za deljenje naslovov IPv4, za potrebe katere določimo pet dimenzij in možnih lastnosti mehanizmov znotraj dimenzij. Ta klasifikacija je nato podlaga za preostanek disertacije. Obstoječe prehodne mehanizme za deljenje naslovov IPv4 klasificiramo v devet razredov glede na njihove lastnosti ter opišemo njihovo delovanje. Nadaljujemo z podrobno analizo lastnosti vzdolž vsake dimenzije, ki so podlaga za analizo kompromisov med posameznimi razredi mehanizmov.

Ker pet dimenzij klasifikacije določa prostor mehanizmov, smo identificirali tudi kombinacijo lastnosti, ki sestavljajo nov uporaben mehanizem – AP64. Podamo definicijo mehanizma, arhitekture in njegovih funkcij ter vse potrebne tehnične podrobnosti za njegovo implementacijo. Utemeljimo tudi njegovo unikatnost in uporabnost pri prehodu interneta s protokola IPv4 na IPv6.

Nazadnje predlagamo ogrodje za vrednotenje zmogljivosti prehodnih mehanizmov za deljenje naslovov IPv4. Na podlagi eksperimentalnih izkušenj smo namreč ugotovili, da je vrednotenje zmogljivosti prehodnih mehanizmov kompleksen problem. Težko je namreč pridobiti kvalitetne implementacije, ki tečejo na ravni jedra operacijskega sistema na enotni platformi. Zato smo razbili klasificirane mehanizme na osnovne operacije nad paketi in eksperimentalno izmerili njihovo zmogljivost. Skupaj z ocenami prostorske kompleksnosti mehanizmov, ki izhajajo iz njihovih lastnosti, definiramo metodologijo za oceno zmogljivosti razredov mehanizmov.

C.1.2 Prispevki k znanosti

Ta disertacija vsebuje naslednje izvirne prispevke k znanosti:

- sistematičen pregled področja prehodnih mehanizmov IPv6, ki ga sestavljajo mehanizmi za uvedbo IPv6 in mehanizmi za deljenje naslovov IPv4,
- klasifikacija mehanizmov za deljenje naslovov IPv4 in ogrodje za teoretično analizo in primerjavo zmogljivosti vseh razredov predlagane klasifikacije,
- analiza lastnosti razredov mehanizmov za deljenje naslovov IPv4 in analiza kompromisov med posameznimi razredi,
- AP64: nadgradljiv mehanizem za deljenje naslovov IPv4 vrste naslov-plus-vrata za avtohtona omrežja IPv6.

C.2 Pregled prehodnih mehanizmov IPv6

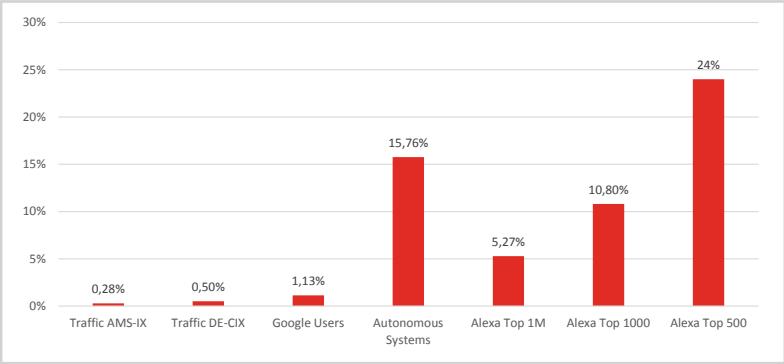
Za izvedbo prehoda interneta s protokola IPv4 na IPv6 so potrebna orodja in priporočila za njihovo uporabo v posameznih scenarijih. Ta orodja imenujemo prehodni mehanizmi – to so arhitekture, metode in tehnologije, ki omogočajo prehod interneta s protokola IPv4 na IPv6. Omogočajo namreč napravam iz ene vrste omrežja, da komunicirajo z napravami iz drugačnega omrežja.

Internet Engineering Task Force (IETF) je organizacija, ki razvija in promovira internetne standarde, posebno standardne povezane s protokoli TCP/IP. Večina prehodnih mehanizmov je tako oblikovanih in standardiziranih v okviru te organizacije. Tudi če je nek mehanizem najprej objavljen kot znanstveni prispevek, navadno avtor prepiše specifikacijo v dokument Internet Draft, ki kandidira za objavo v obliki dokumenta RFC.

Identificiramo lahko dve prekrivajoči se fazi prehoda interneta z IPv4 na IPv6. V prvi fazi je bil cilj predvsem povezati naprave in omrežja, ki so že podpirala IPv6 med seboj preko bistveno bolj razširjenih omrežij IPv4, kar je navadno vključevalo različne tunelirne mehanizme. Cilj je bil pripeljati povezljivost IPv6 čim dlje v robna omrežja interneta. Razlog za to potrebo je bil predvsem v tem, da velika večina ponudnikov dostopa do interneta ni imela ustrezne infrastrukture, ki bi podpirala protokol IPv6. Te mehanizme prve faze prehoda imenujemo *mehanizmi za uvedbo IPv6*. Pri drugi fazi pa ne gre več za povezovanje IPv6-otokov med seboj, ampak zagotavljanje dolgoročnega sobivanja protokolov IPv4 in IPv6. Dokler ne bo velika večina internetnih naprav združljivih z IPv6, bo nemogoče "izključiti" protokol IPv4. To pa je v navzkrižju s problemom izčrpavanja naslovnega prostora IPv4, kar je razlog, da je druga faza osredinjena predvsem na *mehanizme za deljenje naslovov IPv4*. Nekateri izmed mehanizmov sodijo v obe skupini. V tej disertaciji se osredinjamo na slednje mehanizme. Glavni razlog je v tem, da se je IETF v zadnjih letih ukvarjal predvsem s to vrsto mehanizmov, prav tako je vanjo usmerjenega tudi največ raziskovalnega in razvojnega dela.

C.2.1 Uvedba protokola IPv6 skozi čas

Hiter pregled nekaterih kazalnikov za prehod interneta s protokola IPv4 na IPv6 omogoča, da ugotovimo, da se prehod ni še niti dobro začel. Slika C.1 kaže nekatere kazalnike, ki na različnih ravneh podajajo sliko o trenutnem stanju prehoda.



Slika C.1

Združeni kazalniki prehoda interneta s protokola IPv4 na IPv6.

Ugotovimo lahko, da ponudniki vsebin (spletni strežniki v našem primeru) zaenkrat najbolje izvajajo prehod svojih storitev na protokol IPv6. Na to kaže dejstvo, da je 24 %, 10,8 % in 5,27 % strežnikov, ki so vsebovani na seznamih Alexa Top 500, Alexa Top 1000 in Alexa Top 1M, dosegljivih preko protokola IPv6. Ravno nasprotno velja za končne uporabnike, kar je razvidno iz števila Googleovih uporabnikov, ki dostopajo do njihovih storitev preko IPv6 ter iz količine prometa IPv6 na največjih vozliščih (Amsterdam in Frankfurt).

Trdimo, da so najpomembnejši kazalniki za ugotavljanje stanja prehoda razmerja med IPv6 in IPv4 prometom ter odstotkom naprav, ki dostopa do popularnih internetnih storitev preko protokola IPv6. Tudi če je stopnja uvedbe IPv6 na ravni avtonomnih sistemov 100 % in če so vse internetne storitve IPv6-omogočene, brez končnih uporabnikov, ki bi do teh dostopali preko protokola IPv6, ne bo mogoče opaziti pomembnega deleža prometa IPv6 na vozliščih. Dejansko to pomeni, da bo večina prometa še vedno uporabljala zgolj protokol IPv4.

C.2.2 Pregled mehanizmov za uvedbo IPv6

Skupni imenovalec teh mehanizmov [52] je namen uvedbe protokola IPv6 v omrežja, ki sicer uporabljajo zgolj IPv4. Deset in več let nazaj so bili ti mehanizmi zelo pomembni, saj je le peščica geografsko zelo razpršenih naprav v internetu podpirala protokol IPv6. Povsod na svetu je bilo je zelo težko ali celo nemogoče pridobiti avtohtono povezljivost IPv6. Mehanizmi za uvedbo IPv6 so omogočali povezovanje "IPv6-otokov v IPv4 morju". Tudi danes so še vedno ponekod v uporabi, čeprav bodo praktiki po-

gosteje zagotovili avtohtono povezljivost IPv6 po tem, ko je odločitev za uvedbo IPv6 enkrat sprejeta [50].

Zavoljo celovitosti obravnavamo vse mehanizme za uvedbo IPv6, ki so bili kakor koli standardizirani (statusi Proposed Standard, Draft Standard in Best Current Practice) ali drugače objavljeni kot dokumenti RFC (statusi Experimental, Informational in Historic). Izjema so tisti, ki so hkrati tudi mehanizmi za deljenje naslovov IPv4. Ti bodo obravnavni v naslednjih poglavjih. Mehanizme nato razdelimo na tri skupine:

- *Dvojni sklad:* naprava pridobi naslova IPv4 in IPv6 in lahko vzpostavlja in sprejema povezave s katero koli drugo napravo.
- *Tuneliranje:* Paketi IPv4 so tunelirani znotraj paketov IPv6, ali pa so IPv6 paketi tunelirani znotraj drugih vrst protokolov omrežne ali aplikacijske plasti.
- *Prevajanje:* Paketi IPv4 so prevedeni v pakete IPv6 in obratno.

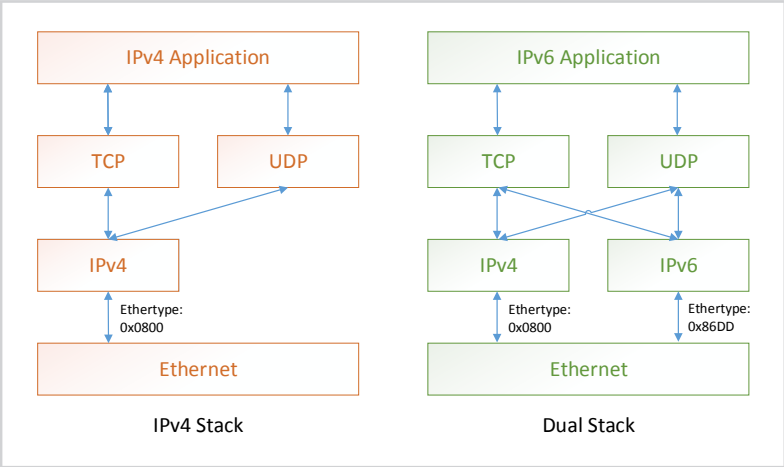
For each mechanism we provide a short description of the main concept.

Dvojni sklad

Ideja mehanizma dvojnega sklada [14, 53, 54], ki ga je potrebno ločiti od mehanizma DSTM [55], je enostavna: vsaka naprava v omrežju dvojnega sklada mora imeti podporo za protokola IPv4 in IPv6, nastavljena mora imeti oba naslova in vse omrežne aplikacije morajo podpirati tako IPv4 kot tudi IPv6. Take naprave lahko vzpostavljajo in sprejemajo povezave do/iz IPv4- in IPv6-naprav brez težav. V tem pomenu je koncept dvojnega sklada prisoten tudi v številnih drugih tunelirnih mehanizmih. Uvedba avtohtone povezljivosti IPv4 in IPv6 je priporočen pristop za prehod na IPv6. Vendar v nekaterih primerih v določene dele omrežne infrastrukture ni mogoče uvesti protokola IPv6, zato so potrebni drugi mehanizmi. Slika 2.14 predstavlja razliko med napravo, ki podpira samo IPv4 in napravo, ki podpira IPv4/IPv6 dvojni sklad.

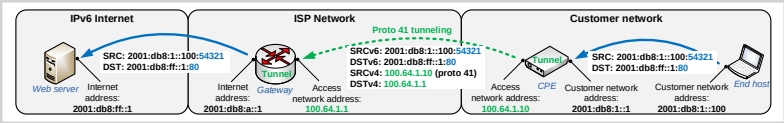
Tuneliranje

Nastavljeni tuneli. Kot že ime pove, morajo biti tuneli ročno nastavljeni prede lahko napravi izmenjata pakete. Nastavljeni paketi omogočajo enostaven način uporabe obstoječe infrastrukture IPv4 za prenos prometa IPv6 do IPv6-otokov [14, 53, 54]. Vsaka izmed končnih točk tunela mora nastaviti štiri parametre: izvorne in ciljne naslove IPv4 in IPv6 obeh končnih točk.



Slika C.2

Enojni sklad v primerjavi z dvojnim skladom v Ethernet omrežju.



Slika C.3

Pri mehanizmu z nastavljenimi tuneli je promet IPv6 navadno tuneliran v paketih IPv4 z uporabo enkapsulacije protokola 41.

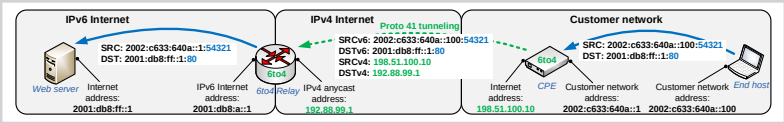
Slika C.3 prikazuje korake pri prenosu paketa z IPv6-naprave do IPv6-spletnega strežnika v internetu preko nastavljenega tunela. Uporabnikova končna točka je nastavljena na njegovi CPE (domači usmerjevalnik), medtem ko se ponudnikova točka nahaja v lastnem jedrnem omrežju.

6to4. 6to4 [75] je mehanizem za samodejno tuneliranje. Vsaka IPv4-naprava z javnim naslovom IPv4 lahko igra vlogo končne točke tunela 6to4. Posebna predpona IPv6 2002::/16 je rezervirana za mehanizem. Vsaka 6to4 končna točka samodejno ustvari predpono IPv6 velikosti /48, tako da združi svoj naslov IPv4 s predpono 2002::/16. 6to4 uporablja protokol 41 za enkapsulacijo paketov IPv6 v pakete IPv4. 6to4 uporablja mehanizem *anycast* [77] za posrednike 6to4, ki nudijo povezljivost med omrežjem 6to4 in IPv6-internetom.

Slika C.4 prikazuje končno napravo za CPE-jem z javnim naslovom IPv4 in podporo za 6to4, ki pošilja paket IPv6 v IPv6-internet preko posrednika 6to4.

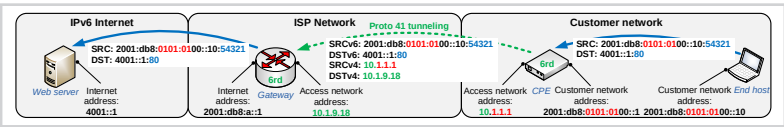
Slika C.4

Pri mehanizmu 6to4 je promet IPv6 navadno tuneliran v paketih IPv4 z uporabo enkapsulacije protokola 41 do najbližjega posrednika 6to4.



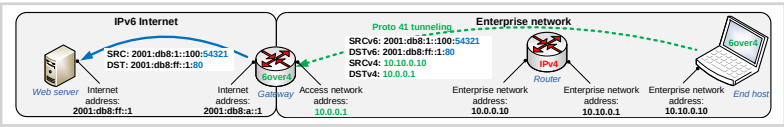
Slika C.5

Pri mehanizmu 6rd je promet IPv6 tuneliran v paketih IPv4 z uporabo enkapsulacije protokola 41 do posrednika 6rd internetnega ponudnika.



Slika C.6

Pri mehanizmu 6over4 je promet IPv6 tuneliran v paketih IPv4 z uporabo enkapsulacije protokola 41 preko infrastrukture IPv4 z uporabo metode *multicast*.



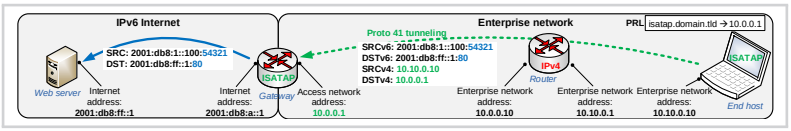
6rd Pomanjkanje nadzora nad potmi med odjemalci 6to4 in posredniki 6to4 se je izkazalo za problematično, zato 6rd [87, 88] predlaga premik posrednika v omrežje internetnega ponudnika in oglaševanje ponudnikovega lastne predpone IPv6 namesto rezervirane predpone za 6to4. Slika C.4 prikazuje končno napravo za CPE-jem z naslovnim prostorom RFC1918 ali javnim naslovom IPv4, ki je nastavljen z uporabo 6rd.

6over4 Namen mehanizma 6over4 [91] je vzpostavitev omrežja IPv6 kot prekrivnega (angl. overlay) omrežja nad obstoječo omrežno infrastrukturo IPv4.

Slika C.6 prikazuje omrežje podjetja povezano v IPv6-internet preko usmerjevalnika, ki podpira 6over4..

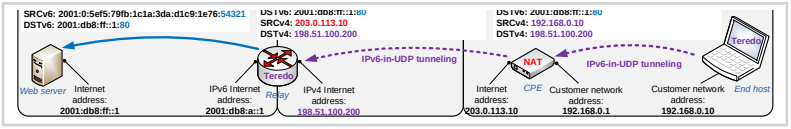
ISATAP ISATAP [94, 95] je konceptualno enak mehanizmu 6over4, le da za delovanje ne podpira metode *multicast*. Uporablja namreč komunikacijski model NBMA in zahteva določeno prednastavitev.

Slika C.7 prikazuje končno napravo s podporo mehanizmu ISATAP in nastavljenim seznamom PRL, pridobljenim preko enega od nastavitvenih mehanizmov (npr. DNS).



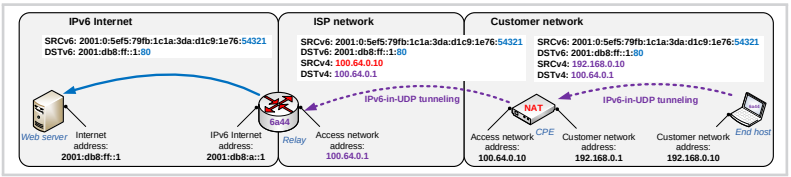
Slika C.7

Pri mehanizmu ISATAP je promet IPv6 tuneliran v paketih IPv4 z uporabo enkapsulacije protokola 41 preko infrastrukture IPv4 z uporabo prednastavljenih seznamov PRL.



Slika C.8

Pri mehanizmu Teredo je promet IPv6 tuneliran v paketih UDP do posrednika Teredo.



Slika C.9

Pri mehanizmu 6a44 je promet IPv6 tuneliran v paketih UDP do posrednika 6a44 v omrežju internetnega ponudnika.

Teredo Teredo [97] je mehanizem za samodejno tuneliranje, ki deluje preko naprav NAPT. Paketi IPv6 niso enkapsulirani z uporabo protokola 41, ampak v pakete UDP, ki omogočajo, da se več odjemalcev Teredo nahaja za isto napravo NAPT.

Slika C.8 prikazuje končno napravo s podporo mehanizmu Teredo, ki je v internet povezana preko naprave NAPT. Končna naprava pošilja paket IPv6 v IPv6-internet preko posrednika Teredo.

6a44 Kot je 6rd "lokalna" različica mehanizma 6to4, je 6a44 [107] "lokalna" različica mehanizma Teredo. Cilj mehanizma 6a44 je omogočiti internetnim ponudnikom, da ponudijo povezljivost IPv6 za svoje stranke tudi v primeru, da uporabljajo naprave CPE z NAPT, ki ne podpirajo IPv6.

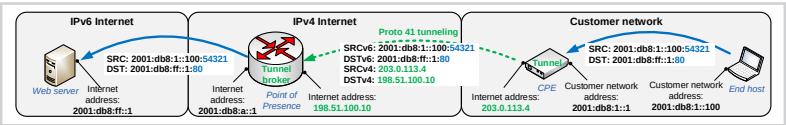
Slika C.9 prikazuje končno napravo s podporo za 6a44 za napravo NAPT, ki pošilja paket IPv6 v IPv6-internet preko posrednika 6a44 v omrežju internetnega ponudnika.

Posrednik tunelov Storitve posredovanja tunelov [108] ponuja vrsta organizacij po svetu že dolgo časa. Ideja je ponuditi storitev nastavitve tunela IPv6-v-IPv4 za končne naprave in usmerjevalnike, ki se ne morejo povezati neposredno v IPv6-internet.

Slika C.10 prikazuje scenarij, kjer je uporabnikova naprava CPE nastavljena s tune-

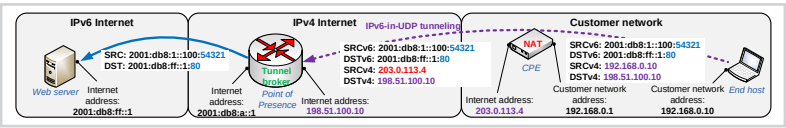
Slika C.10

Mehanizem s posredovanjem tunelov, kjer je promet IPv6 tuneliran med napravo CPE in ponudnikovo točko PoP z uporabo enkapsulacije protokola 41.



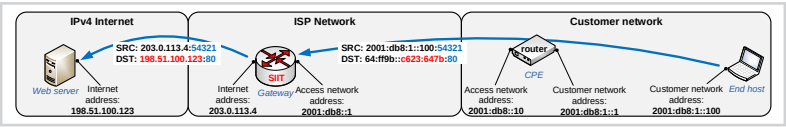
Slika C.11

Mehanizem s posredovanjem tunelov, kjer je promet IPv6 tuneliran od končne naprave do ponudnikove točke PoP z uporabo enkapsulacije UDP.



Slika C.12

Pri mehanizmih SIIT je promet IPv6 preveden v promet IPv4 z uporabo prevajalnika SIIT in ena-na-ena preslikav vrste IPv4-do-IPv6.



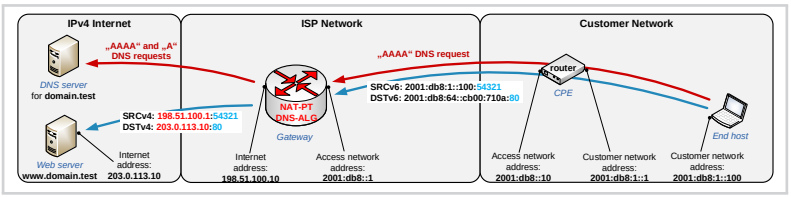
lom do posrednika tunelov z uporabo protokola 41 za enkapsulacijo paketov IPv6. Slika C.11 prikazuje podoben scenarij, vendar v tem primeru končna točka (računalnik uporabnika) neposredno vzpostavi tunel do posrednika tunelov, navadno z uporabo enkapsulacije UDP (protokola TSP in AYIYA).

Prevajanje

SIIT *Stateless IP/ICMP Translation* (SIIT) [116, 117] je algoritem za prevajanje zaglavij paketov IPv4 v zaglavja paketov IPv6 in obratno. Gre za prevajalnik na omrežni ravni. IPv4-naprava in IPv6-naprava lahko izmenjujeta pakete, če je med njima prevajalnik SIIT.

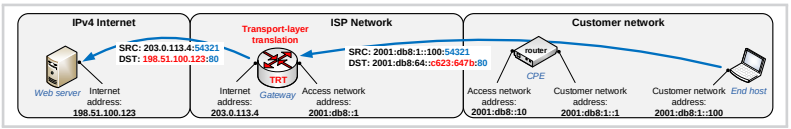
Slika C.12 prikazuje scenarij, kjer je promet IPv6, ki prihaja od končne naprave, algoritmično prevajan v promet IPv4 s pomočjo ustrezne zamenjave zaglavij IP.

NAT-PT Ne glede na to, da je NAT-PT [28] lahko uporabljen tudi kot mehanizem za deljenje naslovov IPv4, ga obravnavamo kot mehanizem za uvedbo IPv6, saj je bil kot tako zasnovan v poznih devetdesetih letih in je bil leta 2007 odpoklican s strani IETF [29]. Odtlej je bil zamenjan z drugimi mehanizmi, na primer s Stateful NAT64.



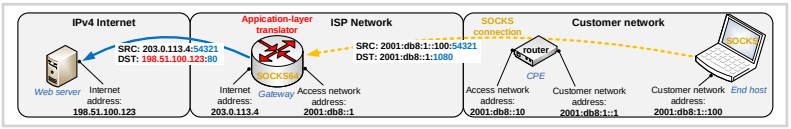
Slika C.13

Pri mehanizmu NAT-PT je promet IPv6 prevajan v promet IPv4 z uporabo prevajalnika NAT-PT in dinamičnih preslikav IPv4-v-IPv6.



Slika C.14

Pri mehanizmu TRT je promet IPv6 prevajan v promet IPv4 z uporabo prevajalnika TRT, ki izvaja prevajanje na prenosni ravni.



Slika C.15

Pri mehanizmu SOCKS64 je IPv6 promet prevajan v proet IPv4 s pomočjo strežnika SOCKS64, ki uporablja prevajanje na aplikacijski ravni.

Slika C.13 prikazuje scenarij, kjer je promet IPv6, ki prihaja od končne naprave, prevajan v promet IPv4, pri čemer se ohranja stanje, z ustrezno zamenjavo zaglavij IP.

TRT *Transport Relay Translator* (TRT) [126] je prevajalnik prenosne plasti, ki ohranja stanje. Ne prevaja in posreduje paketov IP, ampak zaključi prenosno povezavo (UDP, TCP) kar sam. Omogoča IPv6-napravam da si izmenjujejo pakete TCP in UDP z IPv4-napravami.

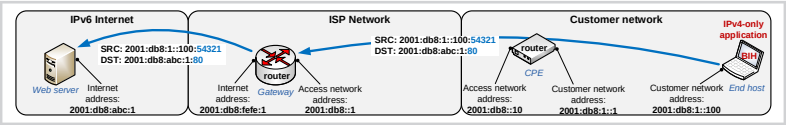
Slika C.14 predstavlja tipičen scenarij mehanizma TRT.

SOCKS64 Ideja mehanizma SOCKS64 [132] je podobna mehanizmu TRT, vendar v tem primeru govorimo o prevajanju na aplikacijski ravni. SOCKS je uveljavljen protokol v svetu IPv4 že od leta 1992. SOCKS64 omogoča IPv6-napravam in IPv4-napravam da komunicirajo.

Slika C.15 prikazuje aplikacijo, ki teče na končni napravi, ko vzpostavlja povezavo z zunanjo napravo s pomočjo imena DNS.

Slika C.16

Pri mehanizmu BIH so zahtevki IPv4 pretvorjeni v zahtevke IPv6 v sami končni napravi. V omrežje ni poslan noben paket IPv4.



BIH Motivacija za mehanizem *Bump-In-the-Host* (BIH) [133] je omogočiti IPv4-aplikacijam, ki tečejo na napravah s podporo protokolu IPv6, da komunicirajo z drugimi IPv6-napravami.

Slika C.16 prikazuje scenarij s končno napravo z IPv4-aplikacijo in omogočenim mehanizmom BIH. Ta prevede zahtevke IPv4 aplikacije neposredno, tako da so v omrežje poslani sintetični paketi IPv6.

C.3 Mehanizmi za deljenje naslovov IPv4

Predstavljamo sistematičen pristop h klasifikaciji in analizi obstoječih mehanizmov za deljenje naslovov IPv4. Da bi jih lahko primerjali in razumeli, abstrahiramo nekatere podrobnosti in raziščemo celoten prostor možnih mehanizmov. Najprej določimo dimenzije klasifikacije in njihove lastnosti. Nato izpeljemo devet razredov, ki klasificirajo obstoječe mehanizme za deljenje naslovov IPv4. Podobni razredi so klasificirani v iste razrede.

C.3.1 Metodologija za klasifikacijo

Metodologija, s pomočjo katere smo določili pet dimenzij klasifikacije, je naslednja. Najprej smo preučili obstoječe mehanizme in izluščili njihove lastnosti. Nato smo združili lastnosti, ki opisujejo iste vidike mehanizmov. Nadalje smo ignorirali tiste kandidatne dimenzije, ki vsaj za enega od mehanizmov niso bile relevantne. Kjer sta dve dimenziji vrnili enako razporeditev mehanizmov, smo izbrali operativno relevantnejšo. Končno množico dimenzij smo dobili, ko smo odsstranili še dimenzije, ki opisujejo operativno manj pomembne lastnosti. To je bil najbolj subjektiven korak v metodologiji.

C.3.2 Dimenzije klasifikacije

Dimenzija 1: lokacija funkcije deljenja naslovov IP. Funkcija za deljenje naslovov IP je lahko v mehanizmu locirana v napravi CPE (mehanizmi A+P), v prehodu (angl.

gateway) ali v obeh (mehanizmi CGN).

Dimenzija 2: hranjenje stanja v prehodu Informacije o stanju lahko prehod hrani za vsako sejo (angl. per flow), za vsako dodelitev naslova oz. vrat (angl. per allocation) ali pa je preprosto ne hrani (angl. stateless).

Dimenzija 3: metoda za prehajanje skozi dostopovno omrežje Ta dimenzija določa način, na katerega je vsebina paketov IPv4 izmenjana med napravo CPE in prehodom. Identificiramo naslednje metode: usmerjanje, tuneliranje, dvojno prevajanje in obratno prevajanje zaglavij.

Dimenzija 4: stopnja zahtevanja IPv6 Niso vsi mehanizmi za deljenje naslovov IPv4 tudi prehodni mehanizmi na protokol IPv6. Nekateri potrebujejo za svoje delovanje prisotnost protokola IPv6 v nekaterih delih omrežja, medtem ko drugi ne potrebujejo protokola IPv6. Ločimo med tremi primeri: IPv6 ni zahtevan, IPv6 je delno zahtevan in IPv6 je obvezen.

Dimenzija 5: politika dodeljevanja naslovov IPv4 in vrat Mehanizem lahko nudi statično in dinamično alokacijo naslovov in vrat, ali pa izključno statično alokacijo.

C.3.3 Pregled mehanizmov za deljenje naslovov IPv4 in klasifikacija

Identificiramo devet razredov mehanizmov za deljenje naslovov IPv4. Tabela C.2 prikazuje posamezne razrede skupaj z njihovimi lastnostmi v posameznih dimenzijah.

C.3.4 Analiza lastnosti

Da bi lahko identificirali kompromise med posameznimi prehodnimi mehanizmi, moramo najprej dobro razumeti lastnosti posameznih razredov. Za vsako lastnost znotraj posamezne dimenzije navedemo vse slabosti.

- *Dimenzija 1: lokacija funkcije deljenja naslovov IP*
  - *CPE in prehod*
    - Posredovanje vrat skozi dve stopnji NAPT-a.
  - *Prehod*
    - Omejen nadzor nad funkcijo NAPT.

        - Večja kompleksnost prehoda.
    - *CPE*
        - Možno samo statično dodeljevanje naslovov IPv4 in vrat.
        - Večja kompleksnost CPE-ja.
- *Dimenzija 2: hranjenje stanja v prehodu*
    - *Na sejo*
        - Potrebno sinhroniziranje stanja.
        - Ves promet mora skozi prehod (angl. hairpinning).
        - Strojne in programske zahteve zaradi hranjenja podatkov o sejah.
        - Težje omogočiti nadgradljivost.
    - *Na dodelitev*
        - Potrebno sinhroniziranje stanja.
        - Ves promet mora skozi prehod (angl. hairpinning).
        - Potrebna je dodatna signalizacija.
    - *Brez*
        - Odvisnost med naslavljanjem IPv4 in IPv6.
        - Potreba po dodatnih pravilih za preslikave.
        - Manj učinkovita poraba naslovnega prostora ponudnika.
        - Nezdružljivost z razpršenimi naslovnimi bloki.
- *Dimenzija 3: metoda za prehajanje skozi dostopovno omrežje*
    - *Usmerjanje*
        - Usmerjanje IPv4 ne spodbuja uvedbe IPv6.
    - *Tuneliranje*
        - Težave z MTU.
        - Težave s prestrezanjem prometa.
        - Večji paketi.
        - Ranljivosti vrste “routing loop”.

- *Dvojno prevajanje*
  - Težave z MTU.
  - Potrebno preračunavanje kontrolne vsote.
  - Potencialno omejena transparentnost do bita DF.
  - Potencialno omejena transparentnost do polja TOS.
  - Potencialno nepodprti fragmentirani "zero-checksum" paketi UDP.
  - Omejena transparentnost do protokola ICMP.
  - Izguba nekaterih polj v zaglavjih IPv4.
- *Obratno prevajanje zaglavij*
  - Težave z MTU.
  - Izguba nekaterih polj v zaglavjih IPv4.

- *Dimenzija 4: stopnja zahtevanja IPv6*
  - *IPv6 ni potreben*
    - Ni spodbujanja k uvedbi IPv6.
    - Potrebna administracija infrastrukture IPv4.
  - *IPv6 delno potreben*
    - IPv4 ostaja v končnih omrežjih.
    - Potrebna administracija infrastrukture IPv4.
  - *IPv6 potreben*
    - Nezdružljivost z IPv4-aplikacijami.
    - Nezdružljivost z aplikacijami, odvisnimi od protokola IP.
    - Podpora samo IPv6-napravam.
    - Potrebna je storitev DNS64.

- *Dimenzija 5: politika dodeljevanja naslovov IPv4 in vrat*
  - *Statično in dinamično*
    - V primeru, da je dodeljevanje dinamično, je potrebno hranjenje stanja v prehodu "na sejo".

Tabela C.1

Razredi mehanizmov in kompromisi med njimi (nesortirano)

Address-Plus-Port	*Carrier-Grade-NAT*
Povezljivost od točke do točke	Preprosti CPE-ji, enostvno upravljanje
Nadgradljivost	Tehnologija zrela in dostopna
S hranjenjem stanja	*Brez hranjenja stanja*
Prožno naslavljanje, ni odvisnosti v naslavljanju	Enostavno porazdeljevanje
Učinkovita uporaba naslovov	Enostavna visoka razpoložljivost
Podpora razpršenim naslovnim blokom	Neposredne poti med CPE-ji
Tuneliranje	*Dvojno prevajanje*
Ohrani pakete IPv4 nedotaknjene	Ni ranljivosti "routing loop"
Ni preračunavanja kontrolne vsote	Ni težav s prestrezanjem
Zrela in razširjena metoda	Ni povečane velikosti paketa
IPv6 zahtevan	*IPv6 ni zahtevan*
Spodbujanje prehoda	Enostavna uvedba
Manj administracije	Podpora starim aplikacijam
Statično dodeljevanje	*Dinamično dodeljevanje*
Obvladljivo stanje	Večje razmerje deljenja naslovov
Učinkovito beleženje	Bolj varno

- *Samo statično*
  - Nizko razmerje souporabe naslovov IPv4.
  - Varnostno tveganje zaradi manjše naključnosti vrat.

C.3.5 Analiza kompromisov

Po tem, ko smo določili ustrezen nabor dimenzij za klasifikacijo mehanizmov za deljenje naslovov IPv4 in ko smo opravili natančno analizo lastnosti znotraj vsake dimenzije, nam sedaj preostane, da izberemo najpomembnejše od teh in ugotovimo kompromise med njimi. Tabela C.1 prikazuje povzetek.

Tabela C.2

Razredi mehanizmov za deljenje naslovov IPv4

Dimenzija	*Razred 1*	*Razred 2*	*Razred 3*	*Razred 4*	*Razred 5*	*Razred 6*	*Razred 7*	*Razred 8*	*Razred 9*
Lokacija funkcije za deljenje naslovov	CPE in prehod	prehod	prehod	prehod	CPE	CPE	CPE	CPE	prehod
Hranjenje stanja v prehodu	na sejo	na sejo	na sejo	na sejo	brez	na dodelitev	brez	brez	na sejo
Metoda za prehod dostopovnega omrežja	usmerjanje	tuneliranje	tuneliranje	usmerjanje	tuneliranje	tuneliranje	dvojno prevajanje	obratno prevajanje zaglavij	dvojno prevajanje
Stopnja zahtevanja IPv6	IPv6 ni zahtevan	IPv6 delno zahtevan	IPv6 ni zahtevan	IPv6 zahtevan	IPv6 delno zahtevan	IPv6 delno zahtevan	IPv6 delno zahtevan	IPv6 delno zahtevan	IPv6 delno zahtevan
Politika dodeljevanja naslovov in vrat	statično in dinamično	statično in dinamično	statično in dinamično	statično in dinamično	samo statično	samo statično	samo statično	samo statično	statično in dinamično

C.4 AP64: nadgradljiv mehanizem za deljenje naslovov IPv4 v avtohtonih omrežjih IPv6

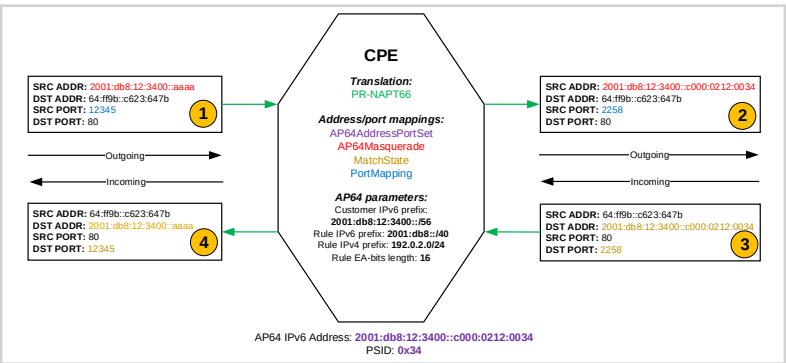
Predlagamo nov mehanizem, ki bo podpiral čista IPv6-omrežja, omogočal prehode brez stanja in izogibanje vsem težavam, ki jih povzročajo tuneliranje, dvojno prevajanje in obratno prevajanje zaglavij. To so tri ključne lasnosti mehanizma AP64.

C.4.1 Definicija

Mehanizem znotraj naše klasifikacije zaseda naslednje mesto:

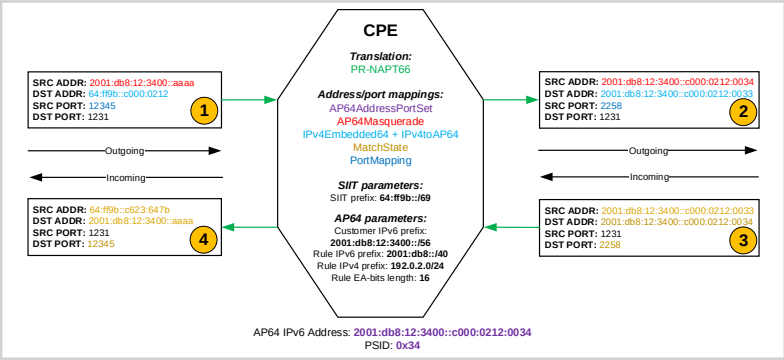
- lokacija funkcije deljenja naslovov IP: *CPE*,
- hranjenje stanja v prehodu: *brez stanja*,
- metoda za prehajanje skozi dostopovno omrežje: *usmerjanje*,
- stopnja zahtevanja IPv6: *IPv6 zahtevan*,
- politika dodeljevanja naslovov IPv4 in vrat: *samo statično*.

To kombinacijo lastnosti definiramo kot nov razred v klasifikaciji, *razred 10*. Slika C.17, slika C.18 in slika C.19 predstavljajo delovanje posameznih komponent mehanizma. Prvi dve sliki prikazujeta dva različna načina delovanja naprave CPE in zadnja slika prikazuje delovanje prehoda.



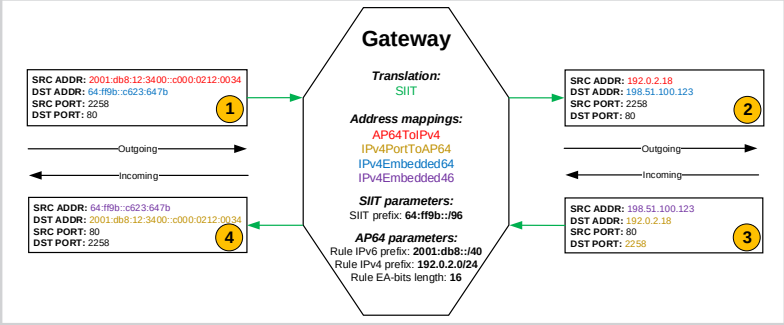
Slika C.17

AP64 CPE, ki manipulira z odhodnimi in dohodnimi paketi, ki izvirajo s končne naprave uporabnika, usmerjeni pa so proti IPv4-internetu.



Slika C.18

AP64 CPE, ki manipulira z odhodnimi in dohodnimi paketi, ki izvirajo s končne naprave uporabnika, usmerjeni pa so proti drugi končni napravi drugega naročnika.



Slika C.19

AP64 prehod, ki manipulira z odhodnimi in dohodnimi paketi, ki izvirajo s končne naprave uporabnika, usmerjeni pa so proti IPv4-internetu.

C.5 Ogrodje za teoretično analizo in primerjavo zmogljivosti

Ocenjevanje zmogljivosti prehodnih mehanizmov je zahteven problem. Težav je več: implementacije je težko pridobiti, pogosto so slabe kvalitete (tečejo v okviru uporabniškega okolja in ne v okviru jedra), potrebujejo različne operacijske sisteme, kar privede do razlik v meritvah itd.

Namesto da bi torej ocenjevali zmogljivost posameznega mehanizma kot celote, smo vse mehanizme za deljenje naslovov IPv4, ki jih obravnavamo v tej disertaciji, preučili in tako prišli do nabora osnovnih operacij nad paketi. Ideja je naslednja: če ugotovimo zmogljivosti posameznih operacij, lahko nato "s seštevanjem" izračunamo tudi teoretično oceno zmogljivosti dejanskih mehanizmov oz. razredov.

Tako smo prišli do nabora naslednjih osnovnih operacij: NAPT44 [20], NAPT64 [30],

NAPT66 (Poglavje 4), NAT64 [117] in TUNNEL [66].

V okviru vrednotenja zmogljivosti mehanizmov nas zanimajo naslednje metrike:

- *CPE Čas*: Koliko procesorskega časa porabi CPE za procesiranje paketa, ki bodisi inicializira novo sej ali je del obstoječe seje?
- *CPE Prostor*: Koliko prostora potrebuje CPE za hranjenje informacij o sejah?
- *Prehod Čas*: Koliko procesorskega časa porabi prehod za procesiranje paketa, ki bodisi inicializira novo sej ali je del obstoječe seje?
- *Prehod Prostor*: Koliko prostora potrebuje prehod za hranjenje informacij o sejah?

Na podlagi meritev časov procesiranja posameznih operacij in na podlagi ocen prostorske kompleksnosti za posamezne razrede mehanizmov, smo izpeljali ocene zmogljivosti razredov prehodnih mehanizmov – performančne indekse, ki jih prikazuje Tabela C.3. Ti omogočajo tudi medsebojno primerjavo zmogljivosti posameznih razredov.

C.6 Zaključek

V tej disertaciji smo se ukvarjali s problemom prehoda interneta na protokol IPv6. Najprej smo pregledali zgodovinske vidike prehoda na IPv6 in analizirali trenutno stanje prehoda. Ugotovili smo, da se vsaj do Aprila 2013 prehod interneta na IPv6 ni niti dobro začel. Zgolj 1,13 % odstotka uporabnikov storitev Google dostopa do njih preko protokola IPv6 in samo 0,28 % oz. 0,50 % vsega prometa v dveh večjih vozliščih predstavlja promet IPv6. Ugotovili smo tudi, da je 15,76 % avtonomnih sistemov pripravljenih na IPv6 in da 5,27 %, 10,80 % in 24 % spletnih strežnikov na seznamih Alexa Top 1M, Alexa Top 1000 and Alexa Top 500 omogoča dostop preko protokola IPv6. Nadalje smo identificirali dve glavni skupini prehodnih mehanizmov: mehanizme za uvedbo protokola IPv6 in mehanizme za deljenje naslovov IPv4. Sistematično smo pregledali vse mehanizme za uvedbo IPv6 tako, da smo za vsakega navedli kratek opis, podali časovno premico glede dogajanja v IETF, podatke o razširjenosti, opis delovanja in razne druge opombe. Nadaljevali smo z analizo mehanizmov za deljenje naslovov IPv4, saj so danes bistveno pomembnejši za prehod na IPv6 kot mehanizmi za uvedbo protokola IPv6.

Tabela C.3

Performance indexes and their components for all IPv4 address sharing mechanisms.

Razred	*CPE operacije*	*Prehod operacije*	*CPE Čas*	*CPE Prost.*	*Prehod Čas*	*Preh. Prost.*	*Perf. indeks*
Razred 5	NAPT44, TUNNEL	TUNNEL	2.536	100	1475	0	*1578*
Razred 7	NAPT44, NAT64	NAT64	2.850	100	1789	0	*1892*
Razred 8	NAPT44, NAT64	NAT64	2.850	100	1789	0	*1892*
Razred 10	NAPT66	NAT64	1.078	100	1789	0	*1890*
Razred 6	NAPT44, TUNNEL	TUNNEL	2.536	100	1475	1000	*2578*
Razred 2	TUNNEL	NAPT44	1.475	0	1061	10000	*11062*
Razred 3	TUNNEL	NAPT44	1.475	0	1061	10000	*11062*
Razred 1	NAPT44	NAPT44	1.061	100	1061	10000	*11162*
Razred 4	IPv6 baseline	NAPT64	1	0	2103	10000	*12104*
Razred 9	NAT64	NAPT64	1.789	0	2103	10000	*12105*

Predstavili smo novo klasifikacijo mehanizmov za deljenje naslovov IPv4, ki je bila nato uporabljena za analizo njihovih lastnosti. Naš cilj je bil predstaviti kompromise povezane z izborom posameznega mehanizma na razumljiv, a konsistenten način. Najprej smo določili 5-dimenzionalen prostor prehodnih mehanizmov za deljenje naslovov IPv4. Nato smo sistematično pregledali vse mehanizme, ki so bili doslej predlagani in jih klasificirali v devet ločenih razredov s pomočjo naše klasifikacije. Nadalje smo analizirali slabosti povezane s posameznimi lastnostmi mehanizmov vzdolž petih dimenzij naše klasifikacije. Končno smo združili analizo lastnosti v kvalitativno analizo kompromisov, s poudarkom na prednostih posameznih lastnosti. Ugotovili smo več stvari. Najprej, da dilema "CGN proti A+P" ni enaka dilemi "NAPT-v-CPE" proti "NAPT-v-prehodu", kar je glede na naše izkušnje, pogosta napačna predstava.

Nato smo ugotovili, da lahko gledamo na prevajanje naslovov in omejitev vrat (angl. port-restriction) kot na dve ločeni funkciji, ki jih lahko izvajamo neodvisno na različnih lokacijah znotraj mehanizma. Poleg tega se je pomembno zavedati, da prevajanje med različnimi družinami IP v kontekstu metod za prehajanje dostopovnih omrežij ni povezano s klasičnim prevajanjem (NAPT). Nazadnje smo tudi spoznali, da je edini (obstoječi) mehanizem, ki resnično spodbuja uvedbo IPv6, Stateful NAT64 (razred 4). To pa zato, ker za svoje delovanje potrebuje čisto omrežje IPv6, kar v resnici želimo v končni fazi v internetu tudi doseči. Vsi ostali mehanizmi zahtevajo tudi dodelitev naslova IPv4 vsaj do neke mere v končnih omrežjih, kar pomeni, da bo potrebno veliko napora, da nekoč te naslove odstranimo iz teh omrežij.

Uporabili smo pet dimenzij naše klasifikacije kot prostor možnih mehanizmov za deljenje naslovov IPv4 in identificirali nov nabor lastnosti, ki skupaj določajo nov uporaben mehanizem – AP64. Določili smo mehanizem, njegovo arhitekturo in funkcije ter predlagali nov razred za njegovo klasifikacijo – razred 10. Poleg tega smo obrazložili vse potrebne tehnične podrobnosti za njegovo implementacijo. Poleg tega smo utemeljili, zakaj je ta mehanizem unikaten in uporaben v prihodnjih letih izvajanja prehoda na IPv6. Glede na to, da potrebuje čista IPv6 omrežja, je zelo spodbuden glede uvajanja IPv6, tako kot to velja za mehanizme razreda 4. Ker pa je AP64 obenem mehanizem A+P (z razliko od razreda 4, ki vsebuje mehanizme CGN), je bistveno bolj nadgradljiv, saj prehod ne hrani nikakršnega stanja o sejah ali dodelitvah. Poleg tega pa uporaba navadnega usmerjanja kot metode za prehajanje dostopovnega omrežja odstrani vse skrbi glede težav z MTU-jem, fragmentacijo, prestrezanjem prometa itd.

Na koncu smo predstavili tudi ogrodje za vrednotenje zmogljivosti mehanizmov za deljenje naslovov IPv4. Po tem, ko smo neuspešno izvedli dva večja eksperimenta glede ocenjevanja zmogljivosti mehanizmov, smo se naučili, da je izjemno težko mehanizme vrednotiti in primerjati "kot celote", saj so zelo kompleksni, njihove implementacije je težko pridobiti, pogosto so te tudi zelo različnih kvalitet in tečejo na različnih operacijskih sistemih. Zato smo namesto tega razstavili razrede mehanizmov na pet osnovnih operacij nad paketi (NAPT44, NAPT64, NAPT66, NAT64 in TUNNEL) ter ovrednotili njihovo zmogljivost. Eksperimentalno smo izmerili čase procesiranja za vsako operacijo tako, da smo uporabili "nedeljive" implementacije, ki tečejo v okviru jedra (in ne uporabnika) in ki so čim bolj preizkušene. Ugotovili smo, da so operacije, ki izvajajo prevajanje IPv6-v-IPv4 najbolj drage, saj sta operaciji NAPT64 in NAT64 potrebovali 19 in $16,16\mu s$ za procesiranje posameznega paketa, medtem ko je ope-

racija TUNNEL potrebovala $13,13\mu s$, operaciji NAPT44 in NAPT66 pa $8,3\mu s$ oz. $11,05\mu s$. S primerjanjem teh rezultatov z dvema osnovnima scenarijema (brez operacij), smo izračunali performančne indekse za vseh pet osnovnih operacij. S pomočjo teh indeksov in ocen prostorske kompleksnosti, ki smo jih ocenili na podlagi lastnosti posameznih mehanizmov, smo lahko teoretično ocenili in nato med seboj primerjali zmogljivosti vseh razredov mehanizmov za deljenje naslovov IPv4. Najprej smo potrdili, da so mehanizmi A+P bistveno bolj učinkoviti kot mehanizmi CGN, kar tudi dokazuje, da je naš razred 10 bistvo učinkovitejši od konkurenčnih mehanizmov, kot so mehanizmi v razredu 4. Poleg tega smo pokazali, da so mehanizmi CGN, ki uporabljajo prevajanje NAPT64 bistveno dražji od ostalih mehanizmov CGN. Na koncu smo tudi ugotovili, da so mehanizmi A+P, ki uporabljajo tuneliranje (razred 5) bolj učinkoviti od tistih, ki uporabljajo dvojno prevajanje.

Med izvajanjem raziskav smo naleteli tudi na nekatere izzive. Denimo, mogoče bi bilo oporekati, da prostor mehanizmov, ki ga napenjajo naše dimenzije, ni popoln. Popolnost tega prostora utemeljujemo tako, da gremo sistematično skozi vse dimenzije in za vsako posebej podamo utemeljitev, zakaj ne obstaja nobena druga možna vrednost, ki je doslej nismo še predvideli. Verjamemo, da je to največ, kar lahko naredimo, saj je zelo težko napovedati prihodnje dosežke na tem področju. Ne glede na to verjamemo, da tudi če takšen dosežek "pokvari" naš prostor, ni posebno težko narediti posodobitev in vključiti novo vrednost znotraj neke dimenzije v popravljeno klasifikacijo.

Zavedamo se tudi, da četudi predlagamo teoretično ogrodje za oceno zmogljivosti, še vedno izvajamo eksperimente nad implementacijami osnovnih operacij. Implementacije pa nikoli niso optimalne in največ, kar lahko naredimo je, da zagotovimo, da so kar najbolj primerljive, torej da vse tečejo v okviru jedra, na istem operacijskem sistemu in da so čim bolj preizkušene.

Uporabnost mehanizma AP64, ki ga predlagamo, bi lahko izpodbijali s trditvijo, da ne podpira aplikacij, ki ne podpirajo protokola IPv6, kot je denimo Skype. To drži in velja tudi za druge mehanizme, ki zahtevajo čista IPv6 robna omrežja. Ker naslova IPv4 ni, take aplikacije enostavno ne delujejo. To je razlog, zakaj namigujemo, da bi lahko bil AP64 uporaben v letih, ki prihajajo, ko bodo vse široko uporabljane aplikacije podpirale protokol IPv6. Poleg tega, kot obstajajo primeri uporabe za Stateful NAT64 (razred 4) tudi danes, gotovo obstajajo tudi primeri upoprabe za AP64, saj sta si zelo podobna.

Podajamo naslednje zamisli za nadaljevanje dela, ki je bilo opravljeno v okviru te

disertacije. Kot prvo, verjamemo da je vredno pregledati ostale "prazne škatle" predlaganega prostora mehanizmov, ki bi lahko vodile do novih potencialno uporabnih mehanizmov. Točno ta pot nas je namreč odkrila do odkritja AP64. Kljub temu predlaganje mehanizma v znanstveni publikaciji ni dovolj za pridobitev kritične ocene s strani standardizacijske skupnosti, torej s strani IETF-a. Objava mehanizma AP64 v obliki dokumenta Internet Draft in nato njegovo predstavljanje in zastopanje na bodočih srečanjih IETF bi lahko bilo uspešno, vendar to ni več predmet akademskega raziskovanja. Eden izmed načinov, kako bi lahko naslovili težavo s Skype-om, je, da razmislimo o uvedbi platformno-odvisnega agenta, podobno kot to omogočajo mehanizmi razreda 9. Vendar pa to ne bi bilo v skladu z našo osnovno predpodstavko, da končnih naprav na noben način ne smemo spreminjati, da bi lahko delovale znotraj nekega mehanizma. Ne glede na to bi bile lahko takšne rešitve smotrne, če bi denimo internetni ponudniki pokazali dovolj zanimanja zanje. Dodatna ideja za nadaljnje raziskovanje je izboljšanje ogrodja za ocenjevanje zmogljivosti tako, da bi poiskali dodatne implementacije osnovnih operacij in nato izbrali najboljšo, ki bi jo vključili v ogrodje.

BIBLIOGRAPHY

[1] Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. ISO/IEC 7498-1, November 1994. URL http://www.iso.org/iso/catalogue_detail.htm?csnumber=20269. accessed 10-April-2013.

[2] Internet host count history. URL http://www.isc.org/solutions/survey/history. accessed 10-April-2013.

[3] V. Cerf, Y. Dalal, and C. Sunshine. Specification of Internet Transmission Control Program. RFC 675, December 1974. URL http://www.ietf.org/rfc/rfc675.txt.

[4] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. URL http://www.ietf.org/rfc/rfc791.txt. Updated by RFCs 1349, 2474.

[5] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. RFC 1518 (Historic), September 1993. URL http://www.ietf.org/rfc/rfc1518.txt.

[6] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519 (Proposed Standard), September 1993. URL http://www.ietf.org/rfc/rfc1519.txt. Obsoleted by RFC 4632.

[7] S. Bradner and A. Mankin. IP: Next Generation (IPng) White Paper Solicitation. RFC 1550 (Informational), December 1993. URL http://www.ietf.org/rfc/rfc1550.txt.

[8] S. Bradner and A. Mankin. The Recommendation for the IP Next Generation Protocol. RFC 1752 (Proposed Standard), January 1995. URL http://www.ietf.org/rfc/rfc1752.txt.

[9] M. McGovern and R. Ullmann. CATNIP: Common Architecture for the Internet. RFC 1707 (Informational), October 1994. URL http://www.ietf.org/rfc/rfc1707.txt.

[10] R. Callon. TCP and UDP with Bigger Addresses (TUBA), A Simple Proposal for Internet Addressing and Routing. RFC 1347 (Informational), June 1992. URL http://www.ietf.org/rfc/rfc1347.txt.

[11] R. Hinden. Simple Internet Protocol Plus White Paper. RFC 1710 (Informational), October 1994. URL http://www.ietf.org/rfc/rfc1710.txt.

[12] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 1883 (Proposed Standard), December 1995. URL http://www.ietf.org/rfc/rfc1883.txt. Obsoleted by RFC 2460.

[13] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. URL http://www.ietf.org/rfc/rfc2460.txt. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[14] R. Gilligan and E. Nordmark. Transition Mechanisms for IPv6 Hosts and Routers. RFC 1933 (Proposed Standard), April 1996. URL http://www.ietf.org/rfc/rfc1933.txt. Obsoleted by RFC 2893.

[15] IPv6 Statistics. URL http://www.google.com/intl/en/ipv6/statistics/. accessed 21-March-2013.

[16] Lars Eggert. IPv6 Deployment Trends. URL http://eggert.org/meter/ipv6. accessed 21-March-2013.

[17] William Jackson. Will IPv6 ever have a killer app?, March 2013. URL http://gcn.com/blogs/cybereye/2013/03/will-ipv6-have-a-killer-app.aspx. accessed 21-March-2013.

[18] Geoff Huston. IPv4: How long do we have? *The Internet Protocol Journal*, 6(4), December 2003. URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-4/ipv4.html.

[19] Gregor Maier, Fabian Schneider, and Anja Feldmann. NAT usage in residential broadband networks. In *Proceedings of the 12th international conference on Passive and active measurement (PAM)*, pages 32–41, Atlanta, GA, March 20–22, 2011. URL http://dl.acm.org/citation.cfm?id=1987510.1987514.

[20] Pyda Srisuresh and Kjeld Borch Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001. URL http://www.ietf.org/rfc/rfc3022.txt.

[21] Policies for IPv4 address space management in the Asia Pacific region, May 2011. URL http://www.apnic.net/policy/add-manage-policy#9.10. accessed 21-March-2013.

[22] Nejc Škoberne, Olaf Maennel, Iain Phillips, Randy Bush, Jan Žorz, and Mojca Ciglarič. IPv4 Address Sharing Mechanism Classification and Tradeoff Analysis. *IEEE/ACM Transactions on Networking*, PP(99):1–1, April 2013. URL http://dx.doi.org/10.1109/TNET.2013.2256147.

[23] B. Carpenter. IPng White Paper on Transition and Other Considerations. RFC 1671 (Informational), August 1994. URL http://www.ietf.org/rfc/rfc1671.txt.

[24] J. Curran. An Internet Transition Plan. RFC 5211 (Informational), July 2008. URL http://www.ietf.org/rfc/rfc5211.txt.

[25] Active IETF Working Groups. URL http://datatracker.ietf.org/wg/. accessed 19-November-2013.

[26] Free Pool of IPv4 Address Space Depleted. URL http://www.nro.net/news/ipv4-free-pool-depleted. accessed 19-November-2013.

[27] Carolyn Duffy Marsan. Biggest mistake for IPv6: It's not backwards compatible, developers admit, March 2009. URL http://www.networkworld.com/news/2009/032509-ipv6-mistake.html. accessed 10-April-2013.

[28] G. Tsirtsis and P. Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). RFC 2766 (Historic), February 2000. URL http://www.ietf.org/rfc/rfc2766.txt. Obsoleted by RFC 4966, updated by RFC 3152.

[29] C. Aoun and E. Davies. Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966 (Informational), July 2007. URL http://www.ietf.org/rfc/rfc4966.txt.

[30] Marcelo Bagnulo, Philip Matthews, and Iljitsch van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146 (Proposed Standard), April 2011. URL http://www.ietf.org/rfc/rfc6146.txt.

[31] Simon Perreault. Ecdysis: Open-Source Implementation of a NAT64 Gateway. URL http://ecdysis.viagenie.ca.

[32] Julio Cossío, Jorge Cano, Alberto Leiva, Juan Arturo Nolazco, and Martha Sordia. Jool – An implementation of RFC6146 (stateful NAT64). URL https://github.com/NICMx/NAT64.

[33] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (INTERNET STANDARD), November 1987. URL http://www.ietf.org/rfc/rfc1035.txt. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.

[34] J. Arkko and A. Keranen. Experiences from an IPv6-Only Network. RFC 6586 (Informational), April 2012. URL http://www.ietf.org/rfc/rfc6586.txt.

[35] Randy Bush, Mark Townsley, and Dan Wing. An IPv4 End of Life Plan: A Shared Vision for IPv6. In *Asia Pacific Regional Internet Conference on Operational Technologies (APRICOT)*, New Delhi, India, February 21–March 2, 2012. URL http://meetings.apnic.net/__data/assets/pdf_file/0016/45241/120229.apops-v4-life-extension.pdf.

[36] Scott O. Bradner. *IPng: Internet Protocol Next Generation*, page 45. Addison-Wesley, September 1996.

[37] Daniel J. Bernstein. The IPv6 Mess. URL http://cr.yp.to/djbdns/ipv6mess.html. accessed 02-April-2013.

[38] Jan Žorž and Ole Trøan. IPv4 on-life support (or) The vision of way forward and tradeoffs in transition to IPv6 mechanisms space. In *10th PLNOG Meeting*, Warszawa, Poland, February 28–March 1, 2013. URL http://www.data.proidea.org.pl/plnog/8edycja/materialy/prezentacje/Jan_Zorz_The_vision_and.pdf.

[39] F. Baker, X. Li, C. Bao, and K. Yin. Framework for IPv4/IPv6 Translation. RFC 6144 (Informational), April 2011. URL http://www.ietf.org/rfc/rfc6144.txt.

[40] Geoff Huston. IPv6: IPv6/IPv4 Comparative Statistics. URL http://bgp.potaroo.net/v6/v6rpt.html. accessed 07-April-2013.

[41] DE-CIX Traffic statistics. URL http://www.de-cix.net/about/statistics/. accessed 07-April-2013.

[42] AMS-IX Statistics. URL https://www.ams-ix.net/technical/statistics. accessed 07-April-2013.

[43] Danny Goodwin. Google Search Market Share Slips as Bing & Yahoo Gain, January 2013. URL http://searchenginewatch.com/article/2236637/Google-Search-Market-Share-Slips-as-Bing-Yahoo-Gain. accessed 10-April-2013.

[44] Percentage of Alexa Top 1000 websites currently reachable over IPv6. URL http://www.worldipv6launch.org/measurements/. accessed 07-April-2013.

[45] Mike Leber. Global IPv6 Deployment Progress Report. URL http://bgp.he.net/ipv6-progress-report.cgi. accessed 07-April-2013.

[46] Emile Aben. Measuring World IPv6 Day - Long-Term Effects, July 2011. URL https://labs.ripe.net/Members/emileaben/measuring-world-ipv6-day-long-term-effects. accessed 07-April-2013.

[47] Nadi Sarrar, Gregor Maier, Bernhard Ager, Robin Sommer, and Steve Uhlig. Investigating IPv6 traffic: what happened at the world IPv6 day? In *Proceedings of the 13th international conference on Passive and Active Measurement*, pages 11–20, Vienna, Austria, March 12–14, 2012. URL http://dx.doi.org/10.1007/978-3-642-28537-0_2.

[48] Iljitsch van Beijnum. World IPv6 Launch gets 27 percent of pageviews on IPv6, June 2012. URL http://arstechnica.com/information-technology/2012/06/world-ipv6-launch-gets-27-percent-of-page-views-on-ipv6/. accessed 21-March-2013.

[49] Eric Vyncke. IPv6 Deployment Aggregated Status. URL http://www.vyncke.org/ipv6status/. accessed 07-March-2013.

[50] IPv6 Industry Survey Results, June 2012. URL http://www.ipv6.bt.com/Downloads/2012_IPv6_Survey.pdf. accessed 07-April-2013.

[51] Maarten Botterman. IPv6 Deployment Survey. In *RIPE 65*, Amsterdam, Netherlands, September 24–28, 2012. URL https://ripe65.ripe.net/presentations/127-RIPE_65_Global_IPv6_Deployment_Survey_2012_highlights.pdf.

[52] Sheila Frankel, Richard Graveman, John Pearce, and Mark Rooks. Guidelines for the Secure Deployment of IPv6. Technical Report 800-119, National Institute of Standards and Technology, December 2010. URL http://csrc.nist.gov/publications/nistpubs/800-119/sp800-119.pdf.

[53] R. Gilligan and E. Nordmark. Transition Mechanisms for IPv6 Hosts and Routers. RFC 2893 (Proposed Standard), August 2000. URL http://www.ietf.org/rfc/rfc2893.txt. Obsoleted by RFC 4213.

[54] Erik Nordmark and Robert E. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213 (Proposed Standard), October 2005. URL http://www.ietf.org/rfc/rfc4213.txt.

[55] Jim Bound, Laurent Toutain, Octavio Medina, Francis Dupont, Hossam Afifi, and Alain Durand. Dual Stack Transition Mechanism (DSTM). Internet-Draft (work in progress), draft-ietf-ngtrans-dstm-08, June 2012. URL http://tools.ietf.org/html/draft-ietf-ngtrans-dstm-08.

[56] R. Callon and D. Haskin. Routing Aspects of IPv6 Transition. RFC 2185 (Informational), September 1997. URL http://www.ietf.org/rfc/rfc2185.txt.

[57] M. Nakamura and J. Hagino. SMTP Operational Experience in Mixed IPv4/v6 Environments. RFC 3974 (Informational), January 2005. URL http://www.ietf.org/rfc/rfc3974.txt.

[58] Y. Shirasaki, S. Miyakawa, T. Yamasaki, and A. Takenouchi. A Model of IPv6/IPv4 Dual Stack Internet Access Service. RFC 4241 (Informational), December 2005. URL http://www.ietf.org/rfc/rfc4241.txt.

[59] T. Chown, S. Venaas, and C. Strauf. Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues. RFC 4477 (Informational), May 2006. URL http://www.ietf.org/rfc/rfc4477.txt.

[60] G. Tsirtsis and H. Soliman. Problem Statement: Dual Stack Mobility. RFC 4977 (Informational), August 2007. URL http://www.ietf.org/rfc/rfc4977.txt.

[61] G. Tsirtsis, V. Park, and H. Soliman. Dual-Stack Mobile IPv4. RFC 5454 (Proposed Standard), March 2009. URL http://www.ietf.org/rfc/rfc5454.txt.

[62] H. Soliman. Mobile IPv6 Support for Dual Stack Hosts and Routers. RFC 5555 (Proposed Standard), June 2009. URL http://www.ietf.org/rfc/rfc5555.txt.

[63] J. Korhonen, J. Soininen, B. Patil, T. Savolainen, G. Bajko, and K. Iisakkila. IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS). RFC 6459 (Informational), January 2012. URL http://www.ietf.org/rfc/rfc6459.txt.

[64] D. Wing and A. Yourtchenko. Happy Eyeballs: Success with Dual-Stack Hosts. RFC 6555 (Proposed Standard), April 2012. URL http://www.ietf.org/rfc/rfc6555.txt.

[65] Sander Steffann, Iljitsch van Beijnum, and Rick van Rein. A Comparison of IPv6 over IPv4 Tunnel Mechanisms. Internet-Draft (work in progress), draft-steffann-tunnels-03, April 2013. URL http://tools.ietf.org/html/draft-steffann-tunnels-03.

[66] A. Conta and S. Deering. Generic Packet Tunneling in IPv6 Specification. RFC 2473 (Proposed Standard), December 1998. URL http://www.ietf.org/rfc/rfc2473.txt.

[67] D. Thaler. IP Tunnel MIB. RFC 2667 (Proposed Standard), August 1999. URL http://www.ietf.org/rfc/rfc2667.txt. Obsoleted by RFC 4087.

[68] A. Terzis, J. Krawczyk, J. Wroclawski, and L. Zhang. RSVP Operation Over IP Tunnels. RFC 2746 (Proposed Standard), January 2000. URL http://www.ietf.org/rfc/rfc2746.txt.

[69] D. Thaler. IP Tunnel MIB. RFC 4087 (Proposed Standard), June 2005. URL http://www.ietf.org/rfc/rfc4087.txt.

[70] J. De Clercq, D. Ooms, S. Prevost, and F. Le Faucheur. Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE). RFC 4798 (Proposed Standard), February 2007. URL http://www.ietf.org/rfc/rfc4798.txt.

[71] R. Graveman, M. Parthasarathy, P. Savola, and H. Tschofenig. Using IPsec to Secure IPv6-in-IPv4 Tunnels. RFC 4891 (Informational), May 2007. URL http://www.ietf.org/rfc/rfc4891.txt.

[72] C. Shen, H. Schulzrinne, S. Lee, and J. Bang. NSIS Operation over IP Tunnels. RFC 5979 (Experimental), March 2011. URL http://www.ietf.org/rfc/rfc5979.txt.

[73] S. Krishnan, D. Thaler, and J. Hoagland. Security Concerns with IP Tunneling. RFC 6169 (Informational), April 2011. URL http://www.ietf.org/rfc/rfc6169.txt.

[74] Gabi Nakibly and Fred L. Templin. Routing Loop Attack Using IPv6 Automatic Tunnels: Problem Statement and Proposed Mitigations. RFC 6324 (Informational), August 2011. URL http://www.ietf.org/rfc/rfc6324.txt.

[75] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056 (Proposed Standard), February 2001. URL http://www.ietf.org/rfc/rfc3056.txt.

[76] G. Armitage, P. Schulter, M. Jork, and G. Harter. IPv6 over Non-Broadcast Multiple Access (NBMA) networks. RFC 2491 (Proposed Standard), January 1999. URL http://www.ietf.org/rfc/rfc2491.txt.

[77] C. Huitema. An Anycast Prefix for 6to4 Relay Routers. RFC 3068 (Proposed Standard), June 2001. URL http://www.ietf.org/rfc/rfc3068.txt.

[78] Yakov Rekhter, Robert G. Moskowitz, Daniel Karrenberg, Geert Jan de Groot, and Eliot Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996. URL http://www.ietf.org/rfc/rfc1918.txt.

[79] P. Savola and C. Patel. Security Considerations for 6to4. RFC 3964 (Informational), December 2004. URL http://www.ietf.org/rfc/rfc3964.txt.

[80] G. Huston. 6to4 Reverse DNS Delegation Specification. RFC 5158 (Informational), March 2008. URL http://www.ietf.org/rfc/rfc5158.txt.

[81] B. Carpenter. Advisory Guidelines for 6to4 Deployment. RFC 6343 (Informational), August 2011. URL http://www.ietf.org/rfc/rfc6343.txt.

[82] V. Kuarsingh, Y. Lee, and O. Vautrin. 6to4 Provider Managed Tunnels. RFC 6732 (Informational), September 2012. URL http://www.ietf.org/rfc/rfc6732.txt.

[83] Emile Aben. 6to4 - How Bad is it Really?, December 2010. URL https://labs.ripe.net/Members/emileaben/6to4-how-bad-is-it-really. accessed 12-April-2013.

[84] Gunter Van de Vede, Ole Trøan, and Tim Chown. Non-Managed IPv6 Tunnels considered Harmful. Internet-Draft (work in progress), draft-vandevelde-v6ops-harmful-tunnels-01, August 2010. URL http://tools.ietf.org/html/draft-vandevelde-v6ops-harmful-tunnels-01.

[85] Ole Trøan and Gunter Van de Velde. Lightweight 4over6 Port-set Allocation: Using PCP To Coordinate Between the CGN and Home Gateway. Internet-Draft (work in progress), draft-tsou-pcp-natcoord-08, October 2012. URL http://tools.ietf.org/html/draft-tsou-pcp-natcoord-08.

[86] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default Address Selection for Internet Protocol Version 6 (IPv6). RFC 6724 (Proposed Standard), September 2012. URL http://www.ietf.org/rfc/rfc6724.txt.

[87] R. Despres. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd). RFC 5569 (Informational), January 2010. URL http://www.ietf.org/rfc/rfc5569.txt.

[88] W. Townsley and O. Troan. IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) – Protocol Specification. RFC 5969 (Proposed Standard), August 2010. URL http://www.ietf.org/rfc/rfc5969.txt.

[89] T. Tsou, C. Zhou, T. Taylor, and Q. Chen. Gateway-Initiated IPv6 Rapid Deployment on IPv4 Infrastructures (GI 6rd). RFC 6654 (Informational), July 2012. URL http://www.ietf.org/rfc/rfc6654.txt.

[90] Steinar H. Gunderson. Global IPv6 Statistics. In *RIPE 57*, Dubai, UAE, October 26–30, 2008. URL http://meetings.ripe.net/ripe-57/presentations/Colitti-Global_IPv6_statistics_-_Measuring_the_current_state_of_IPv6_for_ordinary_users_.7gzD.pdf.

[91] B. Carpenter and C. Jung. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. RFC 2529 (Proposed Standard), March 1999. URL http://www.ietf.org/rfc/rfc2529.txt.

[92] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007. URL http://www.ietf.org/rfc/rfc4861.txt. Updated by RFC 5942.

[93] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007. URL http://www.ietf.org/rfc/rfc4862.txt.

[94] F. Templin, T. Gleeson, and D. Thaler. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 5214 (Informational), March 2008. URL http://www.ietf.org/rfc/rfc5214.txt.

[95] F. Templin, T. Gleeson, M. Talwar, and D. Thaler. Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). RFC 4214 (Experimental), October 2005. URL http://www.ietf.org/rfc/rfc4214.txt. Obsoleted by RFC 5214.

[96] F. Templin. Transmission of IPv4 Packets over Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) Interfaces. RFC 5579 (Informational), February 2010. URL http://www.ietf.org/rfc/rfc5579.txt.

[97] C. Huitema. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380 (Proposed Standard), February 2006. URL http://www.ietf.org/rfc/rfc4380.txt. Updated by RFCs 5991, 6081.

[98] Daniel Maier, Oliver Haase, and Jürgen Wäsch. NAT hole punching revisited. In *IEEE 36th Conference on Local Computer Networks (LCN)*, pages 147–150, Bonn, Germany, 2011. URL http://dx.doi.org/10.1109/LCN.2011.6115173.

[99] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489 (Proposed Standard), March 2003. URL http://www.ietf.org/rfc/rfc3489.txt. Obsoleted by RFC 5389.

[100] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard), October 2008. URL http://www.ietf.org/rfc/rfc5389.txt.

[101] Hiang-Ming Huang, Quincy Wu, and Yi-Bing Lin. Enhancing Teredo IPv6 tunneling to traverse the symmetric NAT. *IEEE Communications Letters*, 10(5):408–410, May 2006. URL http://dx.doi.org/10.1109/LCOMM.2006.1633339.

[102] D. Harrington and W. Hardaker. Transport Security Model for the Simple Network Management Protocol (SNMP). RFC 5591 (Draft Standard), June 2009. URL http://www.ietf.org/rfc/rfc5591.txt.

[103] D. Thaler. Teredo Extensions. RFC 6081 (Proposed Standard), January 2011. URL http://www.ietf.org/rfc/rfc6081.txt.

[104] NETMARKETSHARE – Market Share Statistics for Internet Technologies. URL http://www.netmarketshare.com. accessed 14-April-2013.

[105] Sebastian Zander, Lachlan L.H. Andrew, Grenville Armitage, Geoff Huston, and George Michaelson. Investigating the IPv6 teredo tunnelling capability and performance of internet clients. *Computer Communication Review*, 42(5):13–20, September 2012. URL http://doi.acm.org/10.1145/2378956.2378959.

[106] Geoff Huston. Testing Teredo, April 2011. URL https://labs.ripe.net/Members/gih/testing-teredo. accessed 14-April-2013.

[107] R. Despres, B. Carpenter, D. Wing, and S. Jiang. Native IPv6 behind IPv4-to-IPv4 NAT Customer Premises Equipment (6a44). RFC 6751 (Experimental), October 2012. URL http://www.ietf.org/rfc/rfc6751.txt.

[108] A. Durand, P. Fasano, I. Guardini, and D. Lento. IPv6 Tunnel Broker. RFC 3053 (Informational), January 2001. URL http://www.ietf.org/rfc/rfc3053.txt.

[109] M. Blanchet and F. Parent. IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP). RFC 5572 (Experimental), February 2010. URL http://www.ietf.org/rfc/rfc5572.txt.

[110] Tunnels By Country. URL http://www.tunnelbroker.net/usage/tunnels_by_country.php. accessed 14-April-2013.

[111] Account Growth Last 20 Months. URL http://www.tunnelbroker.net/usage/accounts_by_month.php. accessed 14-April-2013.

[112] AICCU – Automatic IPv6 Connectivity Client Utility. URL http://www.sixxs.net/tools/aiccu/. accessed 14-April-2013.

[113] Tunnel Information and Control protocol (TIC). URL http://www.sixxs.net/tools/tic/. accessed 14-April-2013.

[114] Jeroen Massar. AYIYA: Anything In Anything. Internet-Draft (work in progress), draft-massar-v6ops-ayiya-02, July 2004. URL http://tools.ietf.org/html/draft-massar-v6ops-ayiya-02.

[115] Jeroen Massar. SixXS Heartbeat Protocol. Internet-Draft (work in progress), draft-massar-v6ops-heartbeat-01, June 2005. URL http://tools.ietf.org/html/draft-massar-v6ops-heartbeat-01.

[116] E. Nordmark. Stateless IP/ICMP Translation Algorithm (SIIT). RFC 2765 (Proposed Standard), February 2000. URL http://www.ietf.org/rfc/rfc2765.txt. Obsoleted by RFC 6145.

[117] Xing Li, Congxiao Bao, and Fred Baker. IP/ICMP Translation Algorithm. RFC 6145 (Proposed Standard), April 2011. URL http://www.ietf.org/rfc/rfc6145.txt.

[118] Marcelo Bagnulo, Andrew Sullivan, Philip Matthews, and Iljitsch van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147 (Proposed Standard), April 2011. URL http://www.ietf.org/rfc/rfc6147.txt.

[119] Congxiao Bao, Christian Huitema, Marcelo Bagnulo, Mohamed Boucadair, and Xing Li. IPv6 Addressing of IPv4/IPv6 Translators. RFC 6052 (Proposed Standard), October 2010. URL http://www.ietf.org/rfc/rfc6052.txt.

[120] X. Li, C. Bao, D. Wing, R. Vaithianathan, and G. Huston. Stateless Source Address Mapping for ICMPv6 Packets. RFC 6791 (Proposed Standard), November 2012. URL http://www.ietf.org/rfc/rfc6791.txt.

[121] Nathan Lutchansky. TAYGA – Simple, no-fuss NAT64 for Linux, . URL http://www.litech.org/tayga/. accessed 15-April-2013.

[122] X. Li, C. Bao, M. Chen, H. Zhang, and J. Wu. The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition. RFC 6219 (Informational), May 2011. URL http://www.ietf.org/rfc/rfc6219.txt.

[123] Xiaoyu Zhao and Yan Ma. Linux based NAT-PT gateway implementation. In *Proceedings of International Conferences on Info-tech and Info-net (ICII)*, volume 5, pages 258–263, Beijing, China, October 29–November 1, 2001. URL http://dx.doi.org/10.1109/ICII.2001.983528.

[124] Joo-Chul Lee, Myung-Ki Shin, and Hyong-Jun Kim. Implementation of NAT-PT/SIIT, ALGs and consideration to the mobility support in NAT-PT environment. In *59th IEEE Vehicular Technology Conference (VTC)*, volume 5, pages 2714–2718, Los Angeles, CA, May 17–19, 2004. URL http://dx.doi.org/10.1109/VETECS.2004.1391414.

[125] Lukasz Tomicki. Network Address Translation, Protocol Translation IPv4/IPv6. URL http://tomicki.net/naptd.php. accessed 15-April-2013.

[126] J. Hagino and K. Yamamoto. An IPv6-to-IPv4 Transport Relay Translator. RFC 3142 (Informational), June 2001. URL http://www.ietf.org/rfc/rfc3142.txt.

[127] P. Srisuresh, G. Tsirtsis, P. Akkiraju, and A. Heffernan. DNS extensions to Network Address Translators (DNS_ALG). RFC 2694 (Informational), September 1999. URL http://www.ietf.org/rfc/rfc2694.txt.

[128] The KAME project. URL http://www.kame.net/. accessed 16-April-2013.

[129] faithd – FAITH IPv6/v4 translator daemon. URL http://www.freebsd.org/cgi/man.cgi?query=faithd&sektion=8. accessed 16-April-2013.

[130] Nathan Lutchansky. Portable Transport Translator Daemon (pTRTd), . URL http://www.litech.org/ptrtd/. accessed 16-April-2013.

[131] Feico Dillema. totd – DNS proxy and translator for IPv6 and IPv4. URL http://manpages.ubuntu.com/manpages/lucid/man8/totd.8.html. accessed 16-April-2013.

[132] H. Kitamura. A SOCKS-based IPv6/IPv4 Gateway Mechanism. RFC 3089 (Informational), April 2001. URL http://www.ietf.org/rfc/rfc3089.txt.

[133] B. Huang, H. Deng, and T. Savolainen. Dual-Stack Hosts Using "Bump-in-the-Host" (BIH). RFC 6535 (Proposed Standard), February 2012. URL http://www.ietf.org/rfc/rfc6535.txt.

[134] K. Tsuchiya, H. Higuchi, and Y. Atarashi. Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS). RFC 2767 (Informational), February 2000. URL http://www.ietf.org/rfc/rfc2767.txt. Obsoleted by RFC 6535.

[135] S. Lee, M-K. Shin, Y-J. Kim, E. Nordmark, and A. Durand. Dual Stack Hosts Using "Bump-in-the-API" (BIA). RFC 3338 (Experimental), October 2002. URL http://www.ietf.org/rfc/rfc3338.txt. Obsoleted by RFC 6535.

[136] Bill Huang, Hui Deng, and Teemu Savolainen. Dual Stack Hosts Using "Bump-In-the-Host" (BIH). In *IETF 78*, Maastricht, Netherlands, July 25–30, 2010. URL http://www.ietf.org/proceedings/78/slides/behave-0.pdf.

[137] David Fernández. Transition Mechanisms BIA, TRT & SOCKS. In *Global IPv6 Summit*, Madrid, Spain, March 13–15, 2002. URL http://www.ipv6-es.com/02/docs/david_fernandez_3.pdf.

[138] Hossam Afifi and Laurent Toutain. Methods for IPv4-IPv6 transition. In *Proceedings of IEEE International Symposium on Computers and Communications (ISCC)*, pages 478–484, Red Sea, Egypt, July 6–8, 1999. URL http://dx.doi.org/10.1109/ISCC.1999.780953.

[139] Laurent Toutain and Hossam Afifi. Dynamic Tunneling: A new method for the IPv4-IPv6 Transition. Internet-Draft (work in progress), draft-ietf-ngtrans-dti-00, December 1998. URL http://tools.ietf.org/html/draft-ietf-ngtrans-dti-00.

[140] Ra'ed AlJa'afreh, John Mellor, Mumtaz Kamala, and Basil Kasasbeh. Bi-directional Mapping System as a New IPv4/IPv6 Translation Mechanism. In *Tenth International Conference on Computer Modeling and Simulation (UKSIM)*, pages 40–45, Cambridge, UK, April 1–3, 2008. URL http://dx.doi.org/10.1109/UKSIM.2008.90.

[141] Yang Xia, Bu Sung Lee, Chai Kiat Yeo, and Vincent Lim Sok Seng. An IPv6 Translation Scheme for Small and Medium Scale Deployment. In *Second International Conference on Advances in Future Internet (AFIN)*, pages 108–112, Venice, Italy, July 18–25, 2010. URL http://dx.doi.org/10.1109/AFIN.2010.25.

[142] Peng Wu, Yong Cui, Mingwei Xu, Jianping Wu, Xing Li, Chris Metz, and Shengling Wang. PET: Prefixing, Encapsulation and Translation for IPv4-IPv6 Coexistence. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, Miami, FL, December 6–10, 2010. URL http://dx.doi.org/10.1109/GLOCOM.2010.5683446.

[143] Wenming Shi, Chuanhe Huang, Qinggang Wang, Yan Chen, Yiming Huang, and Yong Cheng. A Novel IPv4/IPv6 Translation Mechanism Based on NAT-PT. In *The 9th International Conference on Advanced Communication Technology (ICACT)*, volume 2, pages 1037–1041, Gangwon-Do, South Korea, February 12–14, 2007. URL http://dx.doi.org/10.1109/ICACT.2007.358535.

[144] Eun-Young Park, Jae-Hwoon Lee, and Byoung-Gu Choe. An IPv4-to-IPv6 dual stack transition mechanism supporting transparent connections between IPv6 hosts and IPv4 hosts in integrated IPv6/IPv4 network. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1024–1027, Paris, France, June 20–24, 2004. URL http://dx.doi.org/10.1109/ICC.2004.1312656.

[145] Kai Wang, Ann-Kian Yeo, and A. L. Ananda. DTTS: a transparent and scalable solution for IPv4 to IPv6 transition. In *10th International Conference on Computer Communications and Networks (ICCCN)*, pages 248–253, Scottsdale, AZ, October 15–17, 2001. URL http://dx.doi.org/10.1109/ICCCN.2001.956257.

[146] Xiaoyu Zhao, Xiaohong Deng, Tao Zheng, Daqing Gu, and Eric Burgey. Implementing Public IPv4 Sharing in IPv6 Environment. In *5th International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, pages 251–255, Valencia, Spain, September 20–25, 2010. URL http://dx.doi.org/10.1109/ICCGI.2010.20.

[147] Yuncheng Zhu, Maoke Chen, Hong Zhang, and Xing Li. Stateless Mapping and Multiplexing of IPv4 Addresses in Migration to IPv6 Internet. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, New Orleans, LA, USA, November 30–December 4, 2008. URL http://dx.doi.org/10.1109/GLOCOM.2008.ECP.433.

[148] Hatim Mohamad Tahir, Azman Taa, and Norshakinah Bt Md. Nasir. Implementation of IPv4 Over IPv6 using Dual Stack Transition Mechanism (DSTM) on 6iNet. In *2nd IEEE International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA)*, volume 2, pages 3156–3162, Damascus, Syria, April 24–28, 2006. URL http://dx.doi.org/10.1109/ICTTA.2006.1684921.

[149] Ra'ed AlJa'afreh, John Mellor, and Irfan Awan. Implementation of IPv4/IPv6 BDMS Translation Mechanism. In *Second UKSIM European Symposium on Computer Modeling and Simulation (EMS)*, pages

512–517, Liverpool, UK, September 8–10, 2008. URL http://dx.doi.org/10.1109/EMS.2008.71.

[150] Ioan Raicu and Sherali Zeadally. Evaluating IPv4 to IPv6 transition mechanisms. In *10th International Conference on Telecommunications (ICT)*, volume 2, pages 1091–1098, Tahiti, Papeete, French Polynesia, February 23–March 1, 2003. URL http://dx.doi.org/10.1109/ICTEL.2003.1191589.

[151] Ra'ed AlJaafreh, John Mellor, and Irfan Awan. Evaluating BDMS and DSTM Transition Mechanisms. In *Second UKSIM European Symposium on Computer Modeling and Simulation (EMS)*, pages 488–493, Liverpool, UK, September 8–10, 2008. doi: http://dx.doi.org/10.1109/EMS.2008.60.

[152] Ra'ed AlJa'afreh, John Mellor, and Irfan Awan. A Comparison Between the Tunneling Process and Mapping Schemes for IPv4/IPv6 Transition. In *International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 601–606, Bradford, UK, May 26–29, 2009. URL http://dx.doi.org/10.1109/WAINA.2009.209.

[153] J. Hanumanthappa, D. H. Manjaiah, and C. V. Aravinda. An Innovative Simulation, Comparison Methodology amp; Framework for Evaluating the Performance Evaluation of a Novel IPv4/IPv6 Transition Mechanisms: BD-SIIT vs DSTM. In *First International Conference on Integrated Intelligent Computing (ICIIC)*, pages 258–263, Bangalore, India, August 5–7, 2010. URL http://dx.doi.org/10.1109/ICIIC.2010.55.

[154] Shaneel Narayan and Sotharith Tauch. IPv4-v6 transition mechanisms network performance evaluation on operating systems. In *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, volume 5, pages 664–668, Chengdu, China, July 9–11, 2010. URL http://dx.doi.org/10.1109/ICCSIT.2010.5564141.

[155] Jiann-Liang Chen, Yao-Chung Chang, and Chien-Hsiu Lin. Performance investigation of IPv4/IPv6 transition mechanisms. In *The 6th International Conference on Advanced Communication Technology (ICACT)*, volume 2, pages 545–550, Phoenix Park, Korea, February 9–11, 2004. URL http://dx.doi.org/10.1109/ICACT.2004.1292930.

[156] Jivika Govil, Jivesh Govil, Navkeerat Kaur, and Harkeerat Kaur. An examination of IPv4 and IPv6 networks: Constraints and various transition mechanisms. In *IEEE Southeastcon*, pages 178–185, Huntsville, AL, April 3–6, 2008. URL http://dx.doi.org/10.1109/SECON.2008.4494282.

[157] J. Gnana Jayanthi and S. Albert Rabara. Transition and mobility management in the integrated IPv4 and IPv6 network - A systematic review. In *International Conference On Electronics and Information Engineering (ICEIE)*, volume 1, pages 162–166, Kyoto, Japan, August 1–3, 2010. URL http://dx.doi.org/10.1109/ICEIE.2010.5559898.

[158] Mallik Tatipamula, Patrick Grossetete, and Hiroshi Esaki. IPv6 integration and coexistence strategies for next-generation networks. *IEEE Communications Magazine*, 42(1):88–96, January 2004. URL http://dx.doi.org/10.1109/MCOM.2004.1262167.

[159] David Geer. To Boost Adoption, IPv6 Proponents Back Once-Shunned Technology. *Computer*, 41(10):16–19, October 2008. URL http://dx.doi.org/10.1109/MC.2008.436.

[160] Michael Mackay, Christopher Edwards, Martin Dunmore, Tim Chown, and Graca Carvalho. A scenario-based review of IPv6 transition tools. *IEEE Internet Computing*, 7(3):27–35, May 2003. URL http://dx.doi.org/10.1109/MIC.2003.1200298.

[161] Xianhui Che and Dylan Lewis. IPv6: Current Deployment and Migration Status. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, 1(2):22–29, June 2010. URL http://www.aece.ro/local_repository/?file=2011_3_8_10_ipv6_current_deployment_and_migration_status.pdf.

[162] Dan Wing. Network Address Translation: Extending the Internet Address Space. *IEEE Internet Computing*, 14(4):66–70, August 2010. URL http://dx.doi.org/10.1109/MIC.2010.96.

[163] Daniel G. Waddington and Fangzhe Chang. Realizing the transition to IPv6. *IEEE Communications Magazine*, 40(6):138–147, June 2002. URL http://dx.doi.org/10.1109/MCOM.2002.1007420.

[164] Jianping Wu, Yong Cui, Xing Li, and Chris Metz. The Transition to IPv6, Part 1: 4over6 for the China Education and Research Network. *IEEE Internet Computing*, 10(3):80–85, May 2006. URL http://dx.doi.org/10.1109/MIC.2006.67.

[165] Yong Cui, Jianping Wu, Xing Li, Mingwei Xu, and Chris Metz. The Transition to IPv6, Part II: The Softwire Mesh Framework Solution. *IEEE Internet Computing*, 10(5):76–80, September 2006. URL http://dx.doi.org/10.1109/MIC.2006.113.

[166] Geoff Huston. NAT++: Address Sharing in IPv4. *The Internet Protocol Journal*, 13(2):2–15, June 2010. URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_13-2/132_address.html.

[167] Andreas Ripke, Rolf Winter, Marcus Brunner, Jürgen Quittek, and Holger Zuleger. The Impact of Port-Based Address-Sharing on Residential Broadband Access Networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, Miami, USA, December 6–10, 2010. URL http://dx.doi.org/10.1109/GLOCOM.2010.5683449.

[168] Yong Cui, Jiang Dong, Peng Wu, Jianping Wu, Chris Metz, Yiu L. Lee, and Alain Durand. Tunnel-Based IPv6 Transition. *IEEE Internet Computing*, 17(2):62–68, March 2013. URL http://dx.doi.org/10.1109/MIC.2012.63.

[169] Yong Cui, Peng Wu, Mingwei Xu, Jianping Wu, Yiu L. Lee, Alain Durand, and Chris Metz. 4over6: network layer virtualization for IPv4-IPv6 coexistence. *IEEE Network*, 26(5):44–48, September 2012. URL http://dx.doi.org/10.1109/MNET.2012.6308074.

[170] Chongfeng Xie and Qiong Sun. Problem Statement for Operational IPv6/IPv4 Co-Existence. In *IETF 80*, Prague, Czech Republic, March 27–April 1, 2011. IETF. URL http://www.ietf.org/proceedings/80/slides/v6ops-15.pdf.

[171] Xiaohong Deng, Lan Wang, Tao Zheng, Daqing Gu, and Eric Burgey. Analysis on IPv6 Transition Solutions and Service Tests. In *The Seventh International Conference on Networking and Services (ICNS)*, pages 85–91, Venice, Italy, May 22–27, 2011. URL http://www.thinkmind.org/download.php?articleid=icns_2011_4_40_10185.

[172] Peng Wu, Yong Cui, Jianping Wu, Jiangchuan Liu, and Chris Metz. Transition from IPv4 to IPv6: A State-of-the-Art Survey. *IEEE Communications Surveys Tutorials, IEEE*, 15(3):1407–1424, July 2013. URL http://dx.doi.org/10.1109/SURV.2012.110112.00200.

[173] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational), December 2002. URL http://www.ietf.org/rfc/rfc3439.txt.

[174] Randy Bush, Gabor Bajko, Mohamed Boucadair, Steven M. Bellovin, Luca Cittadini, Olaf Maennel, Reinaldo Penno, Teemu Savolainen, and Jan Zorz. The Address plus Port (A+P) Approach to the IPv4 Address Shortage. RFC 6346 (Experimental), August 2011. URL http://www.ietf.org/rfc/rfc6346.txt.

[175] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), March 1997. URL http://www.ietf.org/rfc/rfc2131.txt. Updated by RFCs 3396, 4361, 5494, 6842.

[176] Matt Smith and Ray Hunt. Network security using NAT and NAPT. In *10th IEEE International Conference on Networks (ICON)*, pages 355–360, Mumbai, India, December 18–21, 2002. URL http://dx.doi.org/10.1109/ICON.2002.1033337.

[177] Geoff Huston. Anatomy: A Look Inside Network Address Translators. *The Internet Protocol Journal*, 7(3):2–32, September 2004. URL http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html.

[178] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), January 2007. URL http://www.ietf.org/rfc/rfc4787.txt.

[179] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382 (Best Current Practice), October 2008. URL http://www.ietf.org/rfc/rfc5382.txt.

[180] D. MacDonald and B. Lowekamp. NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN). RFC 5780 (Experimental), May 2010. URL http://www.ietf.org/rfc/rfc5780.txt.

[181] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger. IANA-Reserved IPv4 Prefix for Shared Address Space. RFC 6598 (Best Current Practice), April 2012. URL http://www.ietf.org/rfc/rfc6598.txt.

[182] P. Srisuresh and B. Ford. Unintended Consequences of NAT Deployments with Overlapping Address Space. RFC 5684 (Informational), February 2010. URL http://www.ietf.org/rfc/rfc5684.txt.

[183] Shane Alcock. Research into the Viability of Service-Provider NAT, August 2008. URL http://www.wand.net.nz/~salcock/someisp/flow_counting/result_page.html.

[184] Mat Ford, Mohamed Boucadair, Alain Durand, Pierre Levis, and Phil Roberts. Issues with IP Address Sharing. RFC 6269 (Informational), June 2011. URL http://www.ietf.org/rfc/rfc6269.txt.

[185] Mohamed Boucadair, Nejc Škoberne, and Wojciech Dec. Analysis of Port Indexing Algorithms. Internet-Draft (work in progress), draft-bsd-softwire-stateless-port-index-analysis-00, September 2011. URL http://tools.ietf.org/html/draft-bsd-softwire-stateless-port-index-analysis-00.

[186] F. Brockners, S. Gundavelli, S. Speicher, and D. Ward. Gateway-Initiated Dual-Stack Lite Deployment. RFC 6674 (Proposed Standard), July 2012. URL http://www.ietf.org/rfc/rfc6674.txt.

[187] Alain Durand, Ralph Droms, James Woodyatt, and Yiu L. Lee. Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion. RFC 6333 (Proposed Standard), August 2011. URL http://www.ietf.org/rfc/rfc6333.txt.

[188] Naoki Matsuhira. Stateless Automatic IPv4 over IPv6 Tunneling with IPv4 Address Sharing. Internet-Draft (work in progress), draft-matsuhira-sa46t-as-03, April 2012. URL http://tools.ietf.org/html/draft-matsuhira-sa46t-as-03.

[189] Mohamed Boucadair, Satoru Matsushima, Yiu Lee, Olaf Bonness, Isabel Borges, and Gang Chen. Motivations for Carrier-side Stateless IPv4 over IPv6 Migration Solutions. Internet-Draft (work in progress), draft-ietf-softwire-stateless-4v6-motivation-04, August 2012. URL http://tools.ietf.org/html/draft-ietf-softwire-stateless-4v6-motivation-04.

[190] Sheng Jiang, Rémi Després, Reinaldo Penno, Yiu Lee, Gang Chen, and Maoke Chen. IPv4 Residual Deployment via IPv6 - a Stateless Solution (4rd). Internet-Draft (work in progress), draft-ietf-softwire-4rd-04, October 2012. URL http://tools.ietf.org/html/draft-ietf-softwire-4rd-04.

[191] Information technology – UPnP Device Architecture – Part 1: UPnP Device Architecture Version 1.0. ISO 29341-1:2011, September 2011. URL http://www.iso.org/iso/catalogue_detail.htm?csnumber=57494.

[192] Stuart Cheshire and Marc Krochmal. NAT Port Mapping Protocol (NAT-PMP). Internet-Draft (work in progress), draft-cheshire-nat-pmp-05, September 2012. URL http://tools.ietf.org/html/draft-cheshire-nat-pmp-05.

[193] Dan Wing, Stuart Cheshire, Mohamed Boucadair, Reinaldo Penno, and Paul Selkirk. Port Control Protocol (PCP). Internet-Draft (work in progress), draft-ietf-pcp-base-28, October 2012. URL http://tools.ietf.org/html/draft-ietf-pcp-base-28.

[194] Chris Donley, Lee Howard, Victor Kuarsingh, John Berg, and Jinesh Doshi. Assessing the Impact of Carrier-Grade NAT on Network Applications. Internet-Draft (work in progress), draft-donley-nat444-impacts-05, October 2012. URL http://tools.ietf.org/html/draft-donley-nat444-impacts-05.

[195] Xiaoyu Zhao, Lan Wang, and Daqing Gu. UPnP Extensions for Public IPv4 Sharing in IPv6 Environment. In *The Sixth International Conference on Networking and Services (ICNS)*, pages 81–85, Cancun, Mexico, March 7–13, 2010. URL http://dx.doi.org/10.1109/ICNS.2010.20.

[196] Olaf Maennel, Randy Bush, Luca Cittadini, and Steven M. Bellovin. A Better Approach than Carrier-Grade-NAT. Technical Report CUCS-041-08, Columbia University, September 2008. URL http://mice.cs.columbia.edu/getTechreport.php?techreportID=560.

[197] Yi-Hsuan Feng, Nen-Fu Huang, Rong-Tai Liu, and Meng-Huan Wu. Flow Digest: A State Replication Scheme for Stateful High Availability Cluster. In *IEEE International Conference on Communications (ICC)*, pages 1298–1303, Glasgow, Scotland, June 24–28, 2007. IEEE. URL http://dx.doi.org/10.1109/ICC.2007.219.

[198] J.C. Mogul and S.E. Deering. Path MTU discovery. RFC 1191 (Draft Standard), November 1990. URL http://www.ietf.org/rfc/rfc1191.txt.

[199] J. Postel. The TCP Maximum Segment Size and Related Topics. RFC 879, November 1983. URL http://www.ietf.org/rfc/rfc879.txt. Updated by RFC 6691.

[200] Wojciech Dec, Rajiv Asati, Congxiao Bao, Hui Deng, and Mohamed Boucadair. Stateless 4Via6 Address Sharing. Internet-Draft (work in progress), draft-dec-stateless-4v6-04, October 2011. URL http://tools.ietf.org/html/draft-dec-stateless-4v6-04.

[201] Mark J. Handley, Eric Rescorla, and IAB. Internet Denial-of-Service Considerations. RFC 4732 (Informational), December 2006. URL http://www.ietf.org/rfc/rfc4732.txt.

[202] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. URL http://www.ietf.org/rfc/rfc792.txt. Updated by RFCs 950, 4884.

[203] Rodrigo Fonseca, George Manning Porter, Randy H. Katz, Scott Shenker, and Ion Stoica. IP Options are not an option. Technical Report UCB/EECS-2005-24, Electrical Engineering and Computer Sciences, University of California at Berkeley, December 2005. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-24.pdf.

[204] Nejc Škoberne and Mojca Ciglarič. Practical Evaluation of Stateful NAT64/DNS64 Translation. *Advances in Electrical and Computer Engineering*, 11(3):49–54, August 2011. URL http://dx.doi.org/10.4316/AECE.2011.03008.

[205] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. URL http://www.ietf.org/rfc/rfc3261.txt. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141.

[206] A. Ramaiah, R. Stewart, and M. Dalal. Improving TCP's Robustness to Blind In-Window Attacks. RFC 5961 (Proposed Standard), August 2010. URL http://www.ietf.org/rfc/rfc5961.txt.

[207] Sarah Banks, Andrea Colmegna, and Tim Spets. TR-069 Amendment 4: CPE WAN Management Protocol. Broadband Forum Technical Report, July 2011. URL http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf.

[208] Ikuhei Yamagata, Yasuhiro Shirasaki, Akira Nakagawa, Jiro Yamaguchi, and Hiroyuki Ashida. NAT444. Internet-Draft (work in progress), draft-shirasaki-nat444-06, July 2012. URL http://tools.ietf.org/html/draft-shirasaki-nat444-06.

[209] Simon Perreault, Ikuhei Yamagata, Shin Miyakawa, Akira Nakagawa, and Hiroyuki Ashida. Common requirements for Carrier Grade NATs (CGNs). Internet-Draft (work in progress), draft-ietf-behave-lsn-requirements-09, August 2012. URL http://tools.ietf.org/html/draft-ietf-behave-lsn-requirements-09.

[210] Xiaoyu Zhao, Lan Wang, Daqing Gu, and Mohammed Boucadair. A Lightweight AplusP Approach for Public IPv4 Address Sharing in IPv6 Environments. In *5th International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, pages 256–261, Valencia, Spain, September 20–25, 2010. URL http://dx.doi.org/10.1109/ICCGI.2010.21.

[211] Ole Trøan, Wojciech Dec, Xing Li, Congxiao Bao, Satoru Matsushima, and Tetsuya Murakami. Mapping of Address and Port with Encapsulation (MAP). Internet-Draft (work in progress), draft-ietf-softwire-map-02, September 2012. URL http://tools.ietf.org/html/draft-ietf-softwire-map-02.

[212] Tetsuya Murakami, Ole Trø an, and Satoru Matsushima. IPv4 Residual Deployment on IPv6 infrastructure - protocol specification. Internet-Draft (work in progress), draft-murakami-softwire-4rd-01, September 2011. URL http://tools.ietf.org/html/draft-murakami-softwire-4rd-01.

[213] Qiong Sun, Chongfeng Xie, Yong Cui, Jianping Wu, Peng Wu, Cathy Zhou, and Yiu L. Lee. Stateless 4over6 in access network. Internet-Draft (work in progress), draft-sun-softwire-stateless-4over6-00, September 2011. URL http://tools.ietf.org/html/draft-sun-softwire-stateless-4over6-00.

[214] Gabor Bajko, Teemu Savolainen, Mohamed Boucadair, and Pierre Levis. Port Restricted IP Address Assignment. Internet-Draft (work in progress), draft-bajko-pripaddrassign-04, March 2012. URL http://tools.ietf.org/html/draft-bajko-pripaddrassign-04.

[215] Qiong Sun, Mohamed Boucadair, Xiaohong Deng, Cathy Zhou, and Tina Tsou. Request to move Connection of IPv6 Domains via IPv4 Clouds (6to4) to Historic status. Internet-Draft (work in progress), draft-troan-v6ops-6to4-to-historic-01, March 2013. URL http://tools.ietf.org/html/draft-troan-v6ops-6to4-to-historic-01.

[216] Yong Cui, Qiong Sun, Mohamed Boucadair, Tina Tsou, Yiu L. Lee, and Ian Farrer. Lightweight 4over6: An Extension to the DS-Lite Architecture. Internet-Draft (work in progress), draft-cui-softwire-b4-translated-ds-lite-09, October 2012. URL http://tools.ietf.org/html/draft-cui-softwire-b4-translated-ds-lite-09.

[217] Xiaohong Deng, Mohamed Boucadair, Cathy Zhou, Tina Tsou, and Gabor Bajko. NAT offload extension to Dual-Stack lite. Internet-Draft (work in progress), draft-zhou-softwire-b4-nat-04, October 2011. URL http://tools.ietf.org/html/draft-zhou-softwire-b4-nat-04.

[218] Reinaldo Penno, Alain Durand, and Alex Clauberg. Stateless DS-Lite. Internet-Draft (work in progress), draft-penno-softwire-sdnat-02, March 2012. URL http://tools.ietf.org/html/draft-penno-softwire-sdnat-02.

[219] Congxiao Bao, Xing Li, Yu Zhai, and Wentao Shang. dIVI: Dual-Stateless IPv4/IPv6 Translation. Internet-Draft (work in progress), draft-xli-behave-divi-04, October 2011. URL http://tools.ietf.org/html/draft-xli-behave-divi-04.

[220] Xing Li, Congxiao Bao, Wojciech Dec, Ole Trøan, Satoru Matsushima, and Tetsuya Murakami. Mapping of Address and Port using Translation (MAP-T). Internet-Draft (work in progress), draft-ietf-softwire-map-t-00, October 2012. URL http://tools.ietf.org/html/draft-ietf-softwire-map-t-00.

[221] Xing Li, Congxiao Bao, Wojciech Dec, Rajiv Asati, Chongfeng Xie, and Qiong Sun. dIVI-pd: Dual-Stateless IPv4/IPv6 Translation with Prefix Delegation. Internet-Draft (work in progress), draft-xli-behave-divi-pd-01, September 2011. URL http://tools.ietf.org/html/draft-xli-behave-divi-pd-01.

[222] Tetsuya Murakami, Gang Chen, Hui Deng, and Wojciech Dec. 4via6 Stateless Translation. Internet-Draft (work in progress), draft-murakami-softwire-4v6-translation-00, July 2011. URL http://tools.ietf.org/html/draft-murakami-softwire-4v6-translation-00.

[223] Masataka Mawatari, Masanobu Mawashima, and Cameron Byrne. 464XLAT: Combination of Stateful and Stateless Translation. Internet-Draft (work in progress), draft-ietf-v6ops-464xlat-08, September 2012. URL http://tools.ietf.org/html/draft-ietf-v6ops-464xlat-08.

[224] O. Troan and R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633 (Proposed Standard), December 2003. URL http://www.ietf.org/rfc/rfc3633.txt. Updated by RFC 6603.

[225] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335 (Best Current Practice), August 2011. URL http://www.ietf.org/rfc/rfc6335.txt.

[226] Weilin Xu, Yigang Yang, Huiting Liu, Guanglin Xu, and Hua Zhang. NAPT66 – Stateful IPv6-to-IPv6 Network Address Port Translation. URL https://code.google.com/p/napt66/. accessed 16-May-2013.

[227] Peter Orosz and Tamas Skopko. Performance Evaluation of a High Precision Software-based Timestamping Solution for Network Monitoring. *International Journal on Advances in Software*, 4(1&2): 181–188, 2011. URL http://www.thinkmind.org/download.php?articleid=soft_v4_n12_2011_13.

[228] Pablo Neira Ayuso et al. Netfiltler – firewalling, NAT, and packet mangling for Linux. URL http://www.netfilter.org/. accessed 25-August-2013.

[229] Terry Moës. IPv6 Address Translation in a Linux Kernel. Master Thesis, 2011. URL http://downloads.sourceforge.net/project/nfnat66/MOES%20Terry%20-%20Master%20Thesis%20-%20NAT66.pdf.