Department of Computer Science and Engineering

A Consolidated Namespace for Network Applications, Developers, Administrators and Users

Miika Komu

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 166/2012

A Consolidated Namespace for Network Applications, Developers, Administrators and Users

Miika Komu

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on December 7th, 2012, at 12 noon.

Aalto University
School of Science
Department of Computer Science and Engineering
Data Communications Software

Supervising professor
Professor Antti Ylä-Jääski

Thesis advisors
Professor Sasu Tarkoma
Professor Andrei Gurtov

Preliminary examiners
Professor Maryline Laurent, Telecom SudParis, France
Associate Professor Anthony D. Joseph, University of California at Berkeley, USA

Opponent
Professor Hannu H. Kari, National Defence University, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 166/2012

ISBN 978-952-60-4904-5 (printed)
ISBN 978-952-60-4905-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-4905-2

Unigrafia Oy
Helsinki 2012

Finland

Publication orders (printed book):
miika@iki.fi

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Miika Komu

Name of the doctoral dissertation
A Consolidated Namespace for Network Applications, Developers, Administrators and Users

Publisher School of Science

Unit Department of Computer Science and Engineering

Series Aalto University publication series DOCTORAL DISSERTATIONS 166/2012

Field of research Networking

Manuscript submitted 11 September 2012 **Date of the defence** 7 December 2012

Permission to publish granted (date) 2 November 2012 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

The current Internet is founded on the TCP/IP architecture that was originally designed around machines rather than humans. In the original architecture, computers were always named using numerical IP addresses until symbolic names were added to the architecture. Today, most end-users access Internet services directly using symbolic host names with DNS extensions or indirectly by utilizing key words with search engines, but the pervasiveness of IP addresses remains a source of inflexibility as the TCP/IP architecture. The wild success of the Internet, moreover, has attracted significant financial investments, and has resulted in a heavy financial stake in the existing infrastructure and architecture. Consequently, even conservative improvements to its ossified design can face a difficult deployment path.

In this dissertation, we examine a number of legacy-compatible and evolutionary solutions to three of the challenges in the TCP/IP architecture. The first challenge of non-persistent addressing stems from the reuse of IP addresses at network, transport and application layers. While this simplified the naming model of the original TCP/IP architecture, it disrupts TCP streams when the topological location of a mobile device changes. As with other causes of non-persistent addressing, Internet transparency is lost as NAT devices are based on private address realms, and site renumbering is difficult as addresses are hard coded into various configurations. The second challenge is that heterogeneous addressing, as introduced by IPv6, complicates the addressing of hosts and the networking logic of applications. The third challenge is that the addressing model offers little support for security, which needs to be reinforced at the various layers of the networking stack. A consolidated namespace meets the requirements of these three high-level challenges.

From the surveyed solutions, we have narrowed down the number of alternatives to seven solutions that fulfill the requirements of a consolidated namespace and used the Host Identity Protocol (HIP) for empirical evaluation. As other work exists in this area, our work focuses on application layer aspects because it has remained relatively unexplored, especially in the context of HIP.

The concrete research problems are threefold. First, we revisit some aspects of the challenges for consolidated naming at the application layer to understand the impact of the problems. Then, we implement improvements on HIP to better meet the goals for consolidated naming for end-users, network application developers and network administrators. Thirdly, we design, develop and analyze technical improvements to HIP in order to facilitate its adoption and deployment.

Keywords mobility, multihoming, site renumbering, IPv6, security, HIP

ISBN (printed) 978-952-60-4904-5 **ISBN (pdf)** 978-952-60-4905-2

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2012

Pages 203 **urn** http://urn.fi/URN:ISBN:978-952-60-4905-2

A! Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Miika Komu

Väitöskirjan nimi
Lujitettu nimiavaruus verkkosovelluksille, kehittäjille, ylläpitäjille ja käyttäjille

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 166/2012

Tutkimusala tietoliikenne

Käsikirjoituksen pvm 11.09.2012 **Väitöspäivä** 07.12.2012

Julkaisuluvan myöntämispäivä 02.11.2012 **Kieli** Englanti

☐ **Monografia** ☒ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä

Nykyinen Internet pohjautuu TCP/IP-arkkitehtuuriin, joka suunniteltiin alunperin koneita silmällä pitäen ihmiskeskeisyyden sijasta. TCP/IP-arkkitehtuurissa tietokoneet nimettiin numeeristen IP-osoitteiden avulla kunnes symboliset nimet lisättiin arkkitehtuuriin. Nykyään useimmat ihmiset käyttävätkin Internet-palveluita suoraan palvelun symbolisella nimellä DNS-laajennoksien avulla tai epäsuorasti hakukoneiden hakusanoilla, mutta IP-osoitteiden läpitunkevuus tekee TCP/IP-arkkitehtuurin edelleen kankeaksi. Lisäksi Internetin menestystarina on houkutellut investointeja ympärilleen, mikä puolestaan on aiheuttanut taloudellista painetta olla kajoamatta Internetin infrastruktuuriin ja arkkitehtuuriin. Tämän vuoksi maltillisiakin muutoksia on vaikea ottaa käyttöön.

Tässä väitöskirjassa tutkitaan joukkoa taaksepäin yhteensopivia ja evolutiivisia ratkaisuja kolmeen eri tutkimusongelmaan TCP/IP-arkkitehtuurissa. Ensimmäisenä ongelmana ovat muuttuvat osoitteet, jotka ovat haastellisia koska samoja osoitteita kierrätetään verkko-, kuljetus- ja sovelluskerroksilla. Vaikka samojen osoitteiden käyttäminen kaikissa kerroksissa yksinkertaisti alunperin TCP/IP-arkkitehtuuria, tämä oikopolku katkaisee liikkuvan laitteen TCP-yhteydet, kun sen sijainti verkossa muuttuu. Muita syitä osoitteiden muutoksiin ovat niin sanottujen NAT-laitteiden tuomat yksityiset osoiteavaruudet, jotka eristävät laitteita toisistaan. Lisäksi myös osoitteistuksen uusimista vaativat organisaatiomuutokset ovat ongelmallisia, koska osoitteita käytetään monesti suoraan erilaisissa asetustiedostoissa. Toisena tutkimusongelmana on IPv6:n mukanaan tuoma heterogeeninen osoitteistus, joka monimutkaistaa laitteiden osoitteistusta ja sovellusten verkkologiikkaa. Kolmantena ongelmana on tietoturva, jota nykyinen Internetin osoitteistusmalli sellaisenaan tukee huonosti, joten sitä pitää tukea erikseen useammassa verkkopinon kerroksessa. Nimiavaruutta, joka ratkaisee kaikki kolme edellä mainittua korkean tason ongelmaa, kutsutaan tässä työssä lujitetuksi (eng. consolidated).

Tässä työssä sopivat lujitetut ratkaisut on rajattu seitsemään, joista Host Identity Protocol (HIP) on valittu empiirisiin kokeiluihin. Ongelmakenttää tarkastellaan sovelluskerroksen näkökannalta, koska tätä puolta ei ole tutkittu kattavasti etenkään HIP:n osalta.

Konkreettiset tutkimuskysymykset voidaan jakaa kolmeen osaan. Ensimmäiseksi tarkastellaan joitakin lujitettujen nimiavaruuksien käytännön haasteita sovelluskerroksen näkökannalta. Toiseksi toteutetaan parannuksia HIP-arkkitehtuuriin, jotta se vastaisi paremmin lujitetun nimiavaruuden vaatimuksia loppukäyttäjien, sovelluskehittäjien ja verkkojen ylläpitäjien kannalta. Kolmanneksi suunnitellaan, kehitetetään ja analysoidaan teknisiä parannuksia HIP-arkkitehtuuriin sen käyttöönotton ja leviämisen tehostamiseksi.

Avainsanat liikkuvuudenhallinta, osoitteistus, IPv6, tietoturva, HIP

ISBN (painettu) 978-952-60-4904-5 **ISBN (pdf)** 978-952-60-4905-2

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2012

Sivumäärä 203 **urn** http://urn.fi/URN:ISBN:978-952-60-4905-2

Preface

I owe my gratitude to Antti Ylä-Jääski, Andrei Gurtov, Pekka Nikander, Martti Mäntylä and late Kimmo Raatikainen, for making this work financially possible.

I would like to convey my thanks to the preliminary examiners, professors Anthony D. Joseph and Maryline Laurent, for their thorough review of my dissertation. I thank professor Hannu H. Kari for accepting the invitation to act as the opponent of the public defense of my thesis.

I wish to thank my co-authors, Sasu Tarkoma, Andrei Gurtov, Samu Varjonen, Janne Lindqvist, Andrey Lukyanenko, Essi Vehmersalo, Jukka Manner, Kristiina Karvonen, and Jaakko Kangasharju, for their efforts.

I would like to acknowledge all the brave students that participated in the various implementation efforts throughout the years, including Antti Partanen, Lauri Karila, Lauri Silvennoinen, Janne Lundberg, Niklas Karlsson, Juha-Matti Tapio, Diego Beltrami, Teresa Finez, Abhinav Pathak, Weiwei Hu, Xiang Liu Tao Wan, Bing Zhou, Xin Gu, Karthik Mallavarapu, Thomas Jansen, Tim Just, Paul Tötterman, Christof Mroz, David Martin, Henrik Ziegeldorf, Hanno Wirtz and Diego Biurrun.

I would like to thank Tobias Heer, Aapo Kalliola, Yu Xiao, Yrjö Raivio, Mohit Sethi, Stefan Götz, Mark Ain, Anna Lantee and Suvi Koskinen for their kind help and tips during the various stages of preparing this manuscript.

I am grateful for a number of colleagues who have challenged and enriched my ideas, including Jukka Ylitalo, Tuomas Aura, Teemu Koponen, Thomas Henderson, Jeff Ahrenholz, Robert Moskowitz, Ari Keränen, Gonzalo Camarillo, Jan Melen, Petri Jokela, Jani Hautakorpi, Marcelo Braun, Shinta Sugimoto, Cui Yong, Vern Paxon, Stuart Cheshire, Andrew McGregor, Tim Shepard, Anisul Huq, Catharina Candolin, Laura Takkinen, Antti Louko, Arto Karila, Mika Kousa, Dmitriy Kuptsov, An-

drei Khurri, Joakim Koskela, Boris Nechaev, Kristian Slavov, Antti Järvinen, Oleg Ponomarev, Tapio Levä, Yrjö Raivio, Sakari Luukkainen, Jukka Nurminen, Zhonghong Ou, and Matti Siekkinen.

Outside of work, I wish to express my gratitude to Soini Nuorala, Markku Juntunen and especially Ville Rasimus for keeping my feet firmly in the ground. I wish to thank my family and friends for their support throughout the years.

I dedicate this work to Suvi Koskinen who has been my lighthouse in the stormy waters of life.

Helsinki, November 19, 2012,

Miika Komu

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Miika Komu, Samu Varjonen, Andrei Gurtov and Sasu Tarkoma. Sockets and Beyond: Assessing the Source Code of Network Applications. Accepted for publication in *Linux Symposium*, Proceedings of the Linux Symposium, Technical Report, Ottawa, Canada, July 2012.

II Miika Komu, Sasu Tarkoma, Jaakko Kangasharju and Andrei Gurtov. Applying a Cryptographic Namespace to Applications. In *Dynamic Interconnection of Networks Workshop (DIN'05)*, Proceedings of the 1st ACM Workshop on Dynamic Interconnection of Networks (co-located with Mobicom 2005 Conference), Cologne, Germany, pp. 23-27, ISBN 1-59593-144-9, September 2005.

III Miika Komu, Sasu Tarkoma and Andrey Lukyanenko. Mitigation of Unsolicited Traffic Across Domains with Host Identities and Puzzles. In *15th Nordic Conference on Secure IT Systems (NordSec 2010)*, Springer Lecture Notes in Computer Science, Volume 7127, pp. 33-48, ISBN 978-3-642-27936-2, Espoo, Finland, October 2010.

IV Janne Lindqvist, Essi Vehmersalo, Miika Komu and Jukka Manner. Enterprise Network Packet Filtering for Mobile Cryptographic Identities. *International Journal of Handheld Computing Research (IJHCR)*, Volume 1, Issue 1, pp. 79-94, ISSN 1947-9158, January 2010.

V Miika Komu and Janne Lindqvist. Leap-of-Faith Security is Enough for IP Mobility. In *Consumer Communications and Networking Conference (CCNC'09)*, Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference, Las Vegas, pp. 830-834, ISBN 978-1-4244-2308-8, February 2009.

VI Kristiina Karvonen, Miika Komu and Andrei Gurtov. Usable Security Management with Host Identity Protocol. In *Computer Systems and Applications (AICCSA'09)*, Proceedings of the Seventh ACS/IEEE International Conference on Computer Systems and Applications, Rabat, pp. 279 - 286, ISBN 978-1-4244-3807-5, May 2009.

List of Abbreviations

P2P-SIP Peer-to-Peer Session Initiation Protocol 7
PI Provider-independent 20
PA Provider-allocated 20
PCP Port Control Protocol 33
PKI Public Key Infrastructure 43
PLA Packet Level Authentication 47
PRNG Pseudo-Random Number Generator 65
QoS Quality of Service 20
RFC Request for Comments 17
RTCWEB Real Time Communication on the Web 34
SA Security Association 45
SAVA Source Address Validation Architecture 42
SCTP Stream Control Transmission Protocol 26
SEND SEcure Neighbor Discovery 41
SHIM6 Site Multihoming for IPv6 27
SMTP Simple Mail Transfer Protocol 69
SIP Session Initiation Protocol 25
SLP Service Location Protocol 33
SRV Service Record 34
SSO Single Sign-On 43
SP Security Policy 45
SPI Security Parameter Index 19
SSL Secure Sockets Layer 9
SSH Secure Shell 8
STUN Session Traversal Utilities for NAT 33
TLS Transport Layer Security 8
TPM Trusted Platform Module 43
TTL Time To Live 17
TTP Trusted Third Party 43
TURN Traversal Using Relay NAT 33
UDP User Datagram Protocol
UIA Unmanaged Internet Architecture 66
ULA Unique Local Address 29
uPnP Universal Plug and Play 32
URI Universal Resource Identifier 35
URL Universal Resource Locator 15
VLAN Virtual LAN 41
VPN Virtual Private Network 8
WLAN Wireless LAN 16
WPA2 Wi-Fi Protected Access version 2 42

1. Introduction

The Internet has grown beyond its original expectations, but its design architecture has remained relatively static. Particularly its IP-based addressing model has remained the same despite the fact that network devices have evolved. For instance, modern smart phones are equipped with multiple network access technologies, and portable devices (e.g. laptops and tablets) traverse between multiple networks. On the other hand, IPv4 address depletion has led to the introduction of NAT devices that have created problems for end-to-end connectivity. IPv6 was designed to reintroduce end-to-end connectivity, but the protocol has been adopted slowly, possibly because it makes firewall rules and network application development more complex. The complexity can also be visible to the end-user, for instance, as latency.

Many of the individual challenges in the Internet addressing architecture have been solved by a number of different, complex and possibly incompatible "band-aid" solutions. Sometimes application developers redundantly solve some of the challenges at the application layer because the network stack or utility libraries are lacking the required functionality.

In this dissertation, we propose to extend the addressing architecture of the Internet in order to consolidate it for application developers, network administrators and end-users. The goal is to explore the limits of the TCP/IP architecture in a backward compatible manner while designing for forward compatibility. To find a balance between a "band-aid" and "clean-slate" solution, we suggest a kind of "hip arthroplasty" for the architecture of the Internet to meet the present challenges in a consolidated way.

By consolidation[1] of the addressing architecture, we refer to the tack-

[1]The term was briefly used, e.g., in RFC [52, p. 4] but here we extend the coverage of the term beyond site renumbering

ling of three challenges: non-persistent, heterogeneous and insecure addressing. To mention a few examples, non-persistence means that the addresses of hosts are topologically dependent, and that the basic TCP/IP architecture does not provide generic support for topologically-independent addressing for mobile or multihoming capable devices. Non-persistence also includes a renumbering of a site when changing the Internet Service Provider (ISP) [52] or Internet transparency [51], that is to say, all devices cannot successfully reach the other devices because Network Address Translation (NAT) and firewall middleboxes block some of the data flows. Heterogeneous addressing is caused by the introduction of IPv6; applications have to be written to support two address families. Insecure addressing refers to the weak security properties of IP addresses, so additional measures are needed for security. As an additional challenge, we also take into account the deployment of protocol architectures from a technical perspective [221, 220].

We argue that the challenges originate from the design of the network-layer addresses. In spite of higher-level naming as supported by DNS, applications have to use addresses and, hence, inherit their limitations. In this dissertation, we present a number of alternative solutions to IP addressing, but the experimentation is based on one particular architecture. *Host Identity Protocol (HIP)* was chosen as the empirical evaluation tool since, in a nutshell, this standardized protocol offers a cryptographic identity for the end-hosts that isolates the application and transport layers from the fluctuations of the underlying network topology in a secure way [164, 174]. The protocol facilitates IPv6 interoperability at the application and network layers [98, 250, 117], and can restore end-to-end connectivity in the presence of NAT devices [129, 234]. Thus, HIP provides a consolidated namespace to meet the three presented high-level challenges, and the approach is reasonably realistic to deploy in practice as it is compatible with legacy applications. The impact of the namespace introduced by HIP is analyzed at the higher levels of the networking stack from the point of view of end-users, firewall administrators and application developers.

1.1 Problem and Scope

The challenges studied in this dissertation are related to the lack of a consolidated addressing model for the Internet. In this dissertation, we

survey a number of different evolutionary solutions to these challenges[2]. However, we narrow the focus to a single solution HIP in the end, and most of the individual articles are also related to this particular architecture.

Compared to some other approaches, a distinct characteristic of HIP is that it provides a new namespace that is visible to the applications. For this reason, we have chosen to focus on the application-layer aspects of HIP in this particular dissertation. This way, we also supplement a number of other dissertations that analyze HIP from different perspectives: mobility mechanisms [249, 134], power consumption on hand-held devices [124], HIP-aware middleboxes [95] and HIP applied to cellular networks [97].

It should be explicitly mentioned that Publication I is also part of another dissertation [232]. This other dissertation is based on the HIP namespace as well, but focuses on connectivity, network hand-off mechanisms from IPv4 to IPv6 and securing name resolution. In contrast, the focus here is to analyze the effects of the new namespace on applications, developers, firewall administrators and users.

While the viability of using HIP to solve the addressing problems will be argued later in this dissertation, the main contribution is related to the analyzing of the artifacts of applying and using HIP at the application layer. Based on this, we postulate three high-level research problems for this dissertation:

Problem I. Revisit the challenges for a consolidated namespace at the application layer.

Problem II. Improve and evaluate HIP as a consolidated namespace from the viewpoint of network application developers, network administrators and end-users.

Problem III. Understand the technical deployment issues related to HIP.

Problem I questions the challenges related to consolidated addressing and revisits the different aspects of it at the application layer. Thus, this problem acts as a "reality check" and is mostly investigated in Publication I.

Problem II takes a step in a more concrete direction and chooses a particular consolidated naming solution from the alternatives presented in

[2]A number of related surveys exist [57, 180, 101, 133]

Chapter 2. In this research problem, we explore and improve HIP architecture empirically in order for it to better meet the challenges of consolidated naming. To add further practical value, this problem is analyzed from the standpoint of different interest groups in Chapter 3. The contributions to this research problem can be attributed to multiple publications as follows. Publication II introduces a new programmable HIP Application Programming Interface (API) for developers, Publication III presents a use case for HIP to protect end-users from unwanted traffic, Publication IV proposes a firewall to support mobile devices that should ease the burden of network administrators and Publication VI shows a usability evaluation of HIP on end-users.

Problem III acts as another reality check on the proposed solution for a consolidated namespace as offered by HIP. Namely, this problem challenges how feasible it is to deploy HIP from a technical perspective. Again, the contributions in answering this research problem originate from multiple individual publications. Publication I gives practical insight on the API deployment in general, not only HIP. Then, Publication III describes another deployment model where HIP is deployed only at the server side, thus avoiding the hurdles of the client-side deployment. Finally, Publication V proposes a transition path for HIP that reduces the infrastructural dependencies, which are often considered a deployment obstacle.

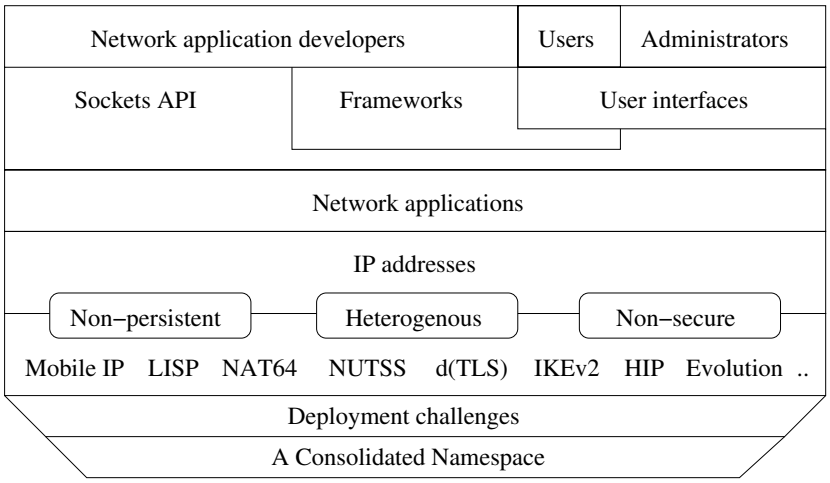

Figure 1.1. A visualization of the challenges in the TCP/IP stack and some solutions

Figure 1.1 visualizes the research challenges of this dissertation using a stack. The layer on the top of the figure represents the target groups of this work, that is, network application developers, users and adminis-

trators. The developers program either directly, using the Sockets API, or indirectly through libraries or network application frameworks, whereas the users and administrative personnel typically utilize applications with graphical or command line user interfaces. However, libraries, frameworks and user interfaces eventually use the Sockets API for network communication, and it exposes IP addresses directly to networks applications, introducing them to non-persistent, heterogeneous, insecure and deployment related challenges. As listed in the bottom part of the figure, these individual challenges can be met using different TCP/IP extensions operating at the various layers of the networks stack. Some of the extensions solve multiple challenges whereas others only a few. Both the taxonomy for consolidated naming and the solution alternatives are described in detail in the next chapter.

The research problems focus on different areas of Figure 1.1. In Problem I, we analyze consolidated naming in the context of the Sockets API and network application frameworks. In Problem II, we investigate and improve HIP to better meet the requirements for consolidated naming from the viewpoint of end-users, network developers and administrators. Research Problem III focuses on the deployment aspects of HIP.

For the sake of completeness, other alternative solutions to HIP will be presented and compared later. However, alternatives will be constrained to backward compatible or incrementally deployable architectures in order to make a fair comparison. In other words, we focus on evolutionary architectures that try to minimize economic impact and extend the current life span of the present TCP/IP architecture instead of, e.g., so-called clean-slate architectures [195, 179, 181]. Consequently, a number of research problems and related solutions are beyond of the scope of this work [110, 118, 14, 131, 78]. For instance, clean-slate networking based on, e.g., the content/data-oriented networking paradigm can require pervasive changes in network applications, stacks and infrastructure. Delay Tolerant Networkings (DTNs) [256, 70, 111] can require a total rewrite of the network logic of the application, and wireless sensor networks [31, 247] do not always implement a full TCP/IP stack. Network mobility and mobile ad-hoc networks will not be considered as the thesis makes no contributions in this field of research. The multicast addressing model is also outside the scope of this work because it is not globally deployed as a network-layer solution. Our standpoint is technical; a complete economic analysis of HIP is not relevant here.

1.2 Methodology

The methodology is heavily inclined towards empirical experimentation in the collection of publications. With the exception of a purely statistical analysis in Publication I, we implemented and analyzed a proof-of-concept prototype in the remaining publications. The methodology for the quantitative analysis of the prototypes includes usability testing, software performance and complexity measurements, and mathematical modeling.

Qualitative analysis is present in all of the publications. Typically, the design is scrutinized based on the practical insight learned by prototyping and then solutions for the short-comings of the design are discussed. As an example of the qualitative aspects, Publication V includes a qualitative analysis related to deployment (backward and forward compatibility of the design). Table 1.1 summarizes the different methodologies used in the publications.

Methodology	PI	PII	PIII	PIV	PV	PVI
Prototyping		✓	✓	✓	✓	✓
Performance		✓	✓	✓	✓	
Statistics	✓					✓
Modeling			✓			
Qualitative	✓	✓	✓	✓	✓	✓
Usability						✓
Complexity	✓	✓			✓	

Table 1.1. The methodology employed to analyze the networking software and concepts in the publications

Simulation was not used as a method in the publications because large-scale scalability was not the focus of this work. However, all of the performance measurements were conducted on commodity hardware to understand the performance impact on individual hosts.

1.3 Contributions

The contributions of the individual publications can be summarized as follows:

Publication I analyzes statistically open-source software in Ubuntu to explore how the applications of today resolve and utilize host names and IP addresses, and how applications employ transport protocols and security. A key finding of this publication is the recurring security problems in the initialization of the OpenSSL library. An-

other finding related to this dissertation is a multihoming issue in all of the four investigated network application frameworks. The contributions of this publication can have a real-world impact for improving the open-source software in various Linux-based distributions. The authors have also reported the UDP multihoming problem to the HIP working group at the IETF, and it is mentioned in RFC5338 [99, p. 10].

Publication II argues that low-level security transparent to applications, such as HIP and Internet Protocol security (IPsec), can be transformed into something more tangible for application developers. We support our argument by extending the Sockets API to support the cryptographic namespace of HIP by implementing the extensions in the Linux kernel and embedding them successfully into an existing application. We were also among the first ones to empirically experiment and report so-called referral problems in HIP and to implement user-specific identifiers for HIP. This publication underwent further evolution in the Internet Engineering Task Force (IETF) community and eventually was published as an experimental standard [128]. The publication inspired the author to collaborate with the SHIM6 working group to design another API that was published also as an RFC [127].

Publication III proposes a cross-layer application of the computational puzzles and the cryptographic namespace in HIP to combat against email spam. In this publication, we integrated HIP puzzle control into a spam filter to introduce a computational cost for senders of spam. As a counter measure, spammers could, however, change their identity to escape identity and puzzle tracking. We addressed this problem as an game-theoretic problem in order to find an optimal solution for inbound email servers. A shortcoming of the proposed solution is the lack of universal HIP adoption; however, the results are generalizable to new application scenarios without legacy burdens, such as Peer-to-Peer Session Initiation Protocol (P2P-SIP) using HIP [122].

Publication IV presents the design, implementation and performance evaluation of a transparent and HIP-aware firewall. The core idea in the design is that the HIP-aware firewall tracks the identities of the client-side devices instead of their IP addresses. This is a

relatively simple and secure way to authenticate mobile and multi-access clients because the IP addresses of the devices can change frequently. The proposed design resembles a Virtual Private Network (VPN) solution, but is based on end-to-end architecture instead of end-to-middle and supports easy network address renumbering at the service side. This approach eases the life of network administrators because they do not need separate access control lists for IPv4 and IPv6. The publication also analyzes and proposes solutions to some challenges related to deploying the solution, including management of the identities and fine-grained access control to services. The outcomes of this experiment are referenced by the HIP experiment report [98, pp. 25].

Publication V investigated if the cryptographic namespace of HIP could be managed without deploying the cryptographic keys in the Domain Name System (DNS). The extra records may introduce management complexity and still be subject to forging until Domain Name System Security Extensions (DNSSEC) are adopted globally. The chosen approach was based on the leap-of-faith security model that was also a recipe for success for Secure Shell (SSH). We implemented the model for HIP using an interposition library that translated application traffic based IP addresses into Host Identifiers on the fly. The implemented library prototype did not require changes in the applications and could fall back on non-HIP connectivity when a peer did not support HIP. We measured the prototype for software complexity and performance. Finally, we analyzed the design for forward compatibility and for security issues inherited from the chosen model. The results indicate that HIP can be managed without any support from DNS, similarly to other substitute technologies such as MobileIP, VPNs and Transport Layer Security (TLS). The publication is referenced by RFC5338 [99, p. 9] and the IETF experiment report [98, pp. 11] on HIP.

Publication VI evaluated how end-users perceive the cryptographic namespace of HIP. While HIP can be visible to the applications, this does not necessarily imply that the users actually observe the use of HIP. To raise the awareness of the user of HIP-based security, we implemented a graphical prompt for the user to explicitly approve all HIP-based connections, a HIP plug-in for the Firefox web

browser and a HIP-aware web site. The entire system was evaluated for usability in two groups of test users. In tests, we applied different combinations of security: no security, leap-of-faith HIP, normal HIP and HIP combined with Secure Sockets Layer (SSL). We employed familiar security indicators in the browser, and most users then noticed when connectivity to the web site was secured, despite the prototype being rather unpolished. The findings of the publications were reported to IETF and, consequently, the end-user GUI is briefly mentioned in RFC5338 [99, p. 9] and referenced in the HIP experiment report [98, pp. 11].

As the contributions of the publications to the research problems are somewhat convoluted as discussed in Section 1.1, the order of the publications deviates from the logical order of the problems. Instead, the publications are organized to ease readability. PI is the most generic publication and gives an introduction to the Sockets API. Next, PII extends the Sockets API to provide a generic API for HIP-aware applications. Then, PIII extends the HIP-specific API to support the use case of mitigation of spam. PIV introduces another use case for HIP, that is, to provide infrastructure-based access control for the services of mobile clients. Finally, PV experiments with use of HIP without the dependency on DNS infrastructure, which is then tested with end-users in PVI.

Table 1.2 summarizes the contributions of the individual publications from the viewpoint of the challenges and the target user groups. The challenges related to non-persistent, heterogeneous and insecure addressing are covered by the first research challenge. The target user groups are related to the third research problem.

Challenge/target group	PI	PII	PIII	PIV	PV	PVI
Non-persistent addressing	✓	✓		✓		
Heterogeneous addressing		✓		✓		
Insecure addressing	✓	✓	✓	✓	✓	✓
End-users						✓
Network app. developers	✓	✓				
Network administrators	✓		✓	✓		

Table 1.2. Contributions of the publications for the addressing challenges and target user groups

1.4 Author's Contributions

The author of this dissertation was involved in all activities of the publications, starting from the conception of the idea, spanning design, implementation, measurement and analysis of the results, and ending in writing of the publication. Here, the author's main contributions to the publications are highlighted.

Publication I: The author contributed many of the ideas in this publication and manually inspected half of the frameworks. A co-author handled collection of the statistics.

Publication II: The author did most of the work for this publication, including design, implementation, experimentation and writing.

Publication III: The idea and design for this publication was conceived by the author. The author integrated the puzzle mechanism into the spam filter and measured puzzle performance. The theoretical analysis originated from the co-authors but was coordinated by the author.

Publication IV: The author designed and conducted the measurements, and participated in the writing of the journal article.

Publication V: The initial prototype was designed by the author and implemented by a student under the instruction of the author. However, the author rewrote the prototype several times to optimize it before the actual measurements, which were also conducted by the author. The author also wrote a large part of the text in the publication.

Publication VI: The author chose to be the second author in this publication even though the workload was split evenly between the first and second author. The author's contribution to this publication was mainly technical even though he participated in the design of the implementation and the design of the usability tests. The author participated also in the actual usability tests in the role of a note taker. The graphical user interfaces were implemented by another developer under the instruction of the author.

During his post-graduate studies, the author also published a number of other research papers, participated in the implementation and interoperability testing of the various HIP-related standards [164, 172, 165, 116,

142, 141, 171, 171, 170] and became a co-author of four IETF standards [99, 128, 127, 129]. The author also contributed HIP-related kernel code that was adopted into the vanilla Linux kernel[3]. The author has been involved with HIP for Linux implementation[4] activities since 2002, which has been used by many other researchers for their own research[5].

1.5 Structure of the Thesis

Chapter 2 describes the background and related work. It describes the challenges relating to the TCP/IP architecture and organizes them into a taxonomy for consolidated addressing. Also, different solutions to meet the various challenges are presented, and HIP is described at the end separately as it has an important role in the collection of articles. Chapter 3 describes and discusses the findings of articles at a high level, and also points out future directions. It should be noted that the results of the performance measurements and other technical details are not repeated in this section; rather, the goal is to position the work from the standpoint of consolidated naming and the target user groups. Finally, the work is concluded in Chapter 4.

[3] http://lwn.net/Articles/144899/

[4] https://launchpad.net/hipl

[5] Please refer to hipl-users and hip-dev mailing lists at http://www.freelists.net/

2. Challenges and Solutions in the TCP/IP Architecture

The TCP/IP architecture was conceived in an era when end-users consisted of a trusted circle of people, and network devices were too large to be portable. At the time, short term needs prevailed and TCP/IP architecture was designed with these factors in mind. While this undoubtedly contributed to the success story of the Internet, these assumptions do not hold true anymore. For instance, practically every Windows desktop machine is equipped with anti-virus software and a firewall. Also, hand-held devices are a part of our everyday lives. Despite the early pioneering work to improve network architectures [209, 200, 201, 55], the restrictions of the original TCP/IP architecture are still present and the remaining challenges to meet modern requirements are fulfilled by different extensions to the original architecture.

This chapter presents a survey of a number of problems originating from restrictions in the TCP/IP architecture and their solutions. We focus on issues related to network addressing from the viewpoint of the network and upper layers. These issues are further organized into a taxonomy and divided into challenges related to non-persistent, heterogeneous and insecure addresses. Within the problem scope of this dissertation, we then enumerate a number of industrial and research solutions that address the challenges within the scope of this dissertation[1]. In addition, technical challenges related to deployment barriers will also be discussed briefly. We also present the architecture of HIP in more detail because it was chosen as the vehicle for experimentation in the collection of articles. Finally, the last section presents a comparison of the different approaches and explains how the collection has improved HIP in order to prepare for the next chapter that presents the contributions in detail.

[1] With the exception of Plutarch, all chosen solutions have been also implemented

2.1 Non-Persistent Addressing

The TCP/IP architecture of the Internet was originally designed around the contemporary restrictions of large computers that were difficult to move around. However, advances in electronics followed Moore's law, resulting in cheaper and smaller electronics for consumers, and portable devices, such as laptops and cellular phones, became pervasive. Consequently, the original restriction on static hosts was no longer true even though it is still present in the design of TCP/IP networking stack.

The TCP/IP stack still remains constrained by its original design, which was effectively a design compromise to make the addressing model simpler. Namely, TCP was designed to reuse the same namespace as the IP layer. While the obvious benefit of this shortcut was to avoid managing an additional namespace, the main drawback is that this makes TCP connections more static. As TCP connections are created based on the same addresses used by the underlying network layer, the connections break when the address changes or is removed. In contrast, UDP is more suitable than TCP for tolerating address changes due to its connectionless nature. However, it can also be used in a connection-oriented way, exposing it to the same constraints as TCP.

In general, the TCP/IP architecture is challenged in the temporal dimension of addressing as it was designed to assume stable addresses. This is not only problematic from the viewpoint of initial connectivity but especially in sustaining of on-going data flows.

In this section, we look firstly at the challenges related to the transient nature of addresses in the TCP/IP architecture from the standpoint of the application layer. While the lifetime of addresses, and perhaps the quality of service related to the use of the address, is directly visible to the application, we describe a more fine-grained taxonomy of the related challenges. Challenges related to long-term disconnectivity as tackled by DTN are beyond the scope of this dissertation. Then, we fit some example solutions in each category. Finally, as some solutions fit multiple categories as is the case with HIP, we provide a summary of the problems solved by each solution.

2.1.1 Challenges

Originally, the IP address was defined to only be used at the network layer, but TCP reused the addresses as its connection identifiers [220, p.

15]. Correspondingly, the Sockets API, the de-facto low-level programming interface for network applications, was designed before DNS and is therefore heavily encumbered with the use of IP addresses [52, p. 15].

While the reuse of IP addresses at multiple layers offers relief from address management issues, it is a layer violation that results in undesired dependencies between the layers. IP addresses are confined to the local network topology and effectively define "where" a host is located, whereas transport-layer identifiers define "who" the connection end-point is [160, p. 6][2]. Consequently, the transport layer becomes dependent on the location of the end-host and its data flows are interrupted when the end-host changes its location. The problem is further aggravated by applications that should be using application-layer identifiers (defining "what"[3].), e.g., FQDN-based identifiers, but instead employ IP addresses directly. Such use can result in connections to incorrect or even malign hosts because IP addresses are typically ephemeral by nature and, in private address realms, overlapping. To further aggravate the problem, applications have also few means of discovering when IP addresses are stale because the Sockets API does not attach any lifetime to the data structures associated with IP addresses [220, p. 11].

For the reasons just described, it could be argued that the basis of the TCP/IP architecture is founded on the assumption of stable or persistent addresses. Paradoxically, addresses are nowadays *non-persistent*, especially due to the advancements in modern, mobile end-user equipment and dynamic network environments. As TCP/IP is universally deployed and adopted, changing its fundamental nature is economically challenging and, thus, various network technologies to reintroduce the persistent addressing model have emerged. However, many of the solutions may tackle only a single problem emerging from non-persistent addressing, and it is not always guaranteed that such point solutions interoperate with each other seamlessly and efficiently. Hence, we roughly categorize the different approaches according to mobility, multihoming, renumbering and Internet transparency challenges.

In the first *mobility* challenge, a single device or an entire network of devices changes its attachment to the network, which typically occurs due to physical movement of the device(s). Network mobility [167] is not

[2]IP addresses are also used for routing which defines "how" to get there

[3]Due to the coupled role of addresses, Fully Qualified Domain Names (FQDNs) could be considered as the new "who", and Universal Resource Locators (URLs) as the new "where" [71, p. 21] due to the pervasiveness of the web

within the scope of this dissertation and the focus is instead on host mobility [155, p. 25]. In host mobility, the problem is dual-fold: when a host moves to different network, it cannot no longer be reached by other hosts and its existing data flows are terminated. Although both of these mobility-related issues are important for applications based on Peer-to-Peer (P2P)-networking, initial reachability is more of a concern for servers whereas the sustaining of existing connections can be considered more important for the clients in the client-server paradigm. It should be noted that mobile end-hosts are also challenging from the viewpoint of infrastructure because network-level firewalls typically authenticate based on access-control lists based on fixed IP addresses.

The second *multihoming* challenge results in multiple alternative paths between two end-hosts and can also be considered dual-fold. Site multihoming occurs transparently from end-host applications, aside from the latency or throughput changes. In contrast, end-host multihoming is more explicit and visible for end-hosts and applications even though many developers are unaware of the end-host multihoming issue as they assume that a single network interface always implies a single address [220, p. 12].

In contrast to mobility[4], the multihoming challenge does not require the physical movement of the host, but the challenge rather stems from the availability of more than one address for a single host, either on the same or different network interfaces. It has impact not only on the client side but also on the server side. At the client side, many hand-held devices support multiple access technologies such as Wireless LAN (WLAN), 3G and Bluetooth. A multihoming problem emerges when the client inadvertently chooses to initiate communications from a “wrong” address, and this may result in a firewall on the path or the server blocking the traffic. At the server side, a misconception is that an application bound to a specific IP address in order to filter incoming data will also send outgoing data from the same address [220, p. 15]. Finally, besides initiation of data flows, the multihoming challenge manifests itself during communications at both the client and server side. When the path between an active pair of addresses breaks the data flow, it would be useful to automatically switch to a functional pair of addresses. Alternatively, two data paths could be simultaneously utilized to maximize throughput. However,

[4]As a common denominator, Ylitalo [249, p. 22] scopes multihoming as a subset of mobility

such functionality remains unsupported by the TCP/IP stack.

The third challenge is *site renumbering*. As many client-side networks are frequently renumbered with, e.g., Dynamic Host Configuration Protocol (DHCP), renumbering issues surface in services that rely on hard-coded IP addresses internally. For instance, corporate mergers or a change in the ISP affects the IP address prefix of a site and requires the site to be renumbered. Due to human error, stale addresses might still be left in various locations after the renumbering. For instance, hard-coded addresses might be discovered in firewall access-control lists of the firewall and configuration files of web servers as described in Request for Comments (RFC) 5887 [52, p. 34]. The RFC further explains that the problems with cached or hard coded IP address literals may be partially attributed to the fact that the DNS look-up functions in the Sockets API do not pass the Time To Live (TTL) values to the application. As human error can result in downtime of services and economic loss, companies tend to avoid site renumbering despite it being adequately documented in the RFC.

The fourth challenge is *Internet transparency* [51, p. 2], which refers to two aspects of the original Internet design: all hosts were universally addressable and intermediate hosts did not essentially modify packets, which resulted in end-to-end connectivity where all hosts were reachable by others. Unfortunately, this transparency was compromised by the evolution of the Internet by the advent of new types of middleboxes. To curb the depletion of IPv4 addresses, NATs middleboxes introduced private address realms that hindered universal addressability, and firewalls were invented to filter undesirable data flows on behalf of the application layer. However, the middleboxes did not come without trade-offs because NATs complicate communications of P2P-software, and firewalls enforce Hypertext Transfer Protocol (HTTP) to serve as the new narrow waist for the Internet [191]. Consequently, the Internet is evolving into an end-to-middle architecture, favoring the client-server paradigm, and making the deployment of new transport and network-layer protocols challenging. It is also worth noting that end-hosts need to be able to address all middleboxes for complete Internet transparency.

Related to Internet transparency, RFC 6250 [220, p. 6] describes two misconceptions about applications and reachability. First, reachability with NATs and firewalls is not always asymmetric as clients can reach servers although the reverse, however, may not hold true. The RFC men-

tions *callbacks*, which are present in the File Transfer Protocol (FTP) for example, as one source of the problem With active data transfer in the FTP [192, p. 4], the client passes its IP address as a callback to the server, which then creates a new TCP connection with the client. If the client is behind a NAT, the creation of the TCP connection fails, resulting in a failure of the file transfer. A second issue described by the RFC is that reachability is not always transitive. As an example scenario, host A can reach B that can reach C, but this does not guarantee that A can reach C as the routes may be different between each node, with different firewalls or NATs between. This problem is also referred to as a *referral* problem. For instance, HTTP [74, p. 62] avoids the referral trap with its clever design of redirection. When a web server receives a request from a client that needs to be redirected to another server, the server instructs the client to connect directly to the other server instead of bluntly passing the client's address as a referral to the other server for connecting back, which would be problematic in the presence of NATs.

2.1.2 Solutions

As discussed in the previous section, the root cause of the inflexibility of the TCP/IP architecture is that the transport and network layers share the same namespace, convoluting the semantics of the layers. Thus, it is only natural to decouple the namespaces, either in a strict way or loosely, to facilitate host mobility and multihoming. In general, this architectural approach is sometimes referred to as the *identifier-locator split* [220, p. 14] and the Internet Architecture Board (IAB) has acknowledged it is a viable way to modularize the TCP/IP architecture [160, p. 7], [151, p. 4],[152, p. 7].

The paradigm of the identity-locator split introduces one level of indirection in naming by introducing location-independent identities for the end-hosts that mask the details of the locators, that is, the location-dependent addresses. Backward-compatible approaches conforming to the paradigm typically try to restore persistent addressing of hosts with "surrogate" addresses [152, p. 58] to be used in place of routable addresses at the application layer. Despite the syntax of the two addresses being essentially the same, the semantic difference is nevertheless usually emphasized by calling the surrogate addresses identifiers. In some of the literature [164, p. 3], an *identity* refers to an abstract entity whereas an *identifier* is a concrete realization of the identity with a certain predefined presentation

format. In the context, we also use the term *persistent identifier* [107, p. 20] to emphasize the invariant nature of stable surrogate addresses.

The identifier-locator split is merely an architectural paradigm that has to be concretely realized with a concrete protocol and, thus, different design alternatives have been proposed. According to RFC 4177 [107, pp. 8, 15], the split can be implemented by modifying existing protocol elements, or by adding new "shim" protocol elements between the application and transport layers, or between the transport and network layers. The RFC further describes the split being realized at the end-hosts or middleboxes (such as site-exit or edge routers). RFC 6115 [152, pp. 11,21] refers to the end-host based approach as *core-edge elimination* and the middlebox-based approach as *core-edge separation* (later referred to with the shorter "elimination" and "separation" terms).

Regarding the syntax and semantics with the identifiers, several alternatives exist according to RFC 4177 [107, pp. 18-20]. The identifier and locator namespaces can be *overlapping* or *disjoint*. With the former, a drawback is that context [200, p. 2] of the use may be needed in order to disambiguate between the overlapping namespaces. With the latter, a drawback is the extra complexity of additional bindings between identifiers and locators. The identifiers can be *structured* (typically hierarchical) or *unstructured*. The term "flat' is also used to refer to unstructured identifiers.

Whether an identifier is unique or not depends on the scope. Three nested classes of network identifiers are described in RFC 4177 [107, pp. 20-23] according to their scope. Starting with the smallest scope, *ephemeral identifiers* are created by two end-hosts during a communication session to distinguish it from others. As an example of this, two hosts using IPsec-based protection use a Security Parameter Index (SPI) [121, p. 12] number to indicate the symmetric key used to protect the packet. *Opportunistic identifiers* are bound either temporally or topologically and do not always guarantee that sequential use of an identifier will result in a connection to the same host. For instance, the present Internet with its private address realms can be categorized into this group because the same, possibly private, IP address can result in communication with a different host, depending on which network the connecting host is located. These two classes of identifiers are jointly labeled as *non-persistent* in the context of this dissertation.

Persistent identifiers are global in their scope and unique across con-

current and parallel sessions according to RFC 4177 [107]. They can be used for initiating communications at any time and anywhere, and are guaranteed to always result in communication with the same host or no communication at all. For instance, MAC addresses are an example of centrally assigned, unique identifiers at the data-link layer. In contrast, public keys generated to identify SSH hosts can be considered statistically unique identifiers at the application layer.

In the context of this work, we clarify two properties for persistent identifiers. Firstly, we specify that such identifiers must facilitate end-host mobility, end-host multihoming, site renumbering and Internet transparency. Secondly, we note that persistent identifiers are impacted by the referral issues when the identifiers are unstructured and used in the context of name directories supporting only structured name look-ups. In such a scenario, the identifiers may need some additional information in order to be successfully searched for through the structured directory. Here we define the referral issue as not rendering a persistent into a non-persistent identifier, and rather describe the referral issues separately.

Besides address agility, the identity-locator split holds also the promise of improving the *routing scalability* of the "core" of the Internet, or Default-Free Zone (DFZ) to be more exact, which is currently facing some addressing-related challenges. For example, many companies prefer Provider-independent (PI) addresses over Provider-allocated (PA) addresses to facilitate easier migration from one ISP to another [152, p. 5],[160, p. 5]. The trade-off here is that PI addresses do not aggregate as well as PA addresses; thus PI addresses create larger routing tables and challenge routing scalability [32, p. 8]. Essentially, the identity-locator split can be used to reap the benefits of PI and PA addresses. The identity namespace offers the same topology-independent functionality as PI addresses, and the locator namespace supports aggregation as it can be based on PA addresses. For applications, the unresolved scalability problems may cause degradation of Quality of Service (QoS) in the future. The solutions (based on the identity-locator split) can limit the degradation and affect the way how applications identify other hosts.

The benefits for the routing scalability in the core-edge elimination approach are only fully realized when most of the hosts on a site are upgraded to support the elimination. In contrast, only the edge routers have to be upgraded in the separation approach to enable it for the whole site[5].

[5]Communications with legacy sites is another issue in core-edge separation

The elimination approach can improve routing scalability because it allows routing tables to grow with the number of ISPs rather than edge networks [112, p. 2]. However, the separation does not support Internet transparency because it creates a separate locator namespace solely for the routers. Consequently, end-hosts cannot address routers anymore because they are confined to the identity namespace [112, p. 4] in the separation approach.

Protocols implementing the identity-locator split are typically based either on *tunneling* or *address rewriting* [107, pp. 15-16]. The tunneling approach is also referred as *map-and-encap*. With tunneling, the packets are encapsulated with an extra header: the inner header contains the identifiers and the outer header the locators. The header is added when the host responsible for the identity-locator split sends the packet, and is correspondingly removed at the destination by the responsible host. In address rewriting schemes, the responsible hosts translate identifiers to locators when sending and translate locators back to identifiers at the destination. The translation can involve the whole address or just portions of it, such as the prefix.

The tunneling approach requires a mapping infrastructure from where to look up the locators corresponding to the identifiers. This approach can result in lost packets, especially when combined with the core-edge separation. When an edge router receives an outgoing packet with the identifiers, it has to look up the corresponding destination locator and, thus, may have to drop the packet until the look-up is completed. As the tunneling scheme adds an extra header, it changes the Maximum Transfer Unit (MTU) and fragmentation processing [160, p. 18]. Tunneling can also interfere with geolocation based on IP addresses [220, p. 14]. However, a benefit of the tunneling is that it is stateless as each packet contains all the necessary information to process it.

As an alternative to tunneling, address rewriting typically requires some extra state at the middleboxes. Typically, packets do not have be dropped because the translation is known beforehand. Translation does not affect MTU as no extra header is added. It should be noted that some translation schemes alter the semantics of IP addresses, e.g., by splitting a single address into identifier and locator portions. Such schemes require changes to application logic as most of them generally treat addresses as opaque tokens without additional semantics [160, p. 18].

Next, we describe some protocols that try to introduce persistent identi-

fiers in order to facilitate mobility, multihoming, site renumbering or internet transparency. We survey different approaches based on the identity-locator split in addition to other application, transport and network-level solutions.

Mobility Solutions

The mobility-related terminology in this section refers to RFC 3753 [155] unless separately mentioned. Network mobility refers to relocating to an entire network without interrupting packet delivery. Network mobility [67])[6] will not be further examined here, and the interested reader may refer to other sources [66, 62, 185, 64]. In *host mobility*, or *terminal mobility*, a host retains its connectivity with other hosts despite changing its network attachment point. For example, this can occur when a laptop moves from outside the range of one WLAN network to another. Session mobility refers to migration of an application-layer session between two different devices.

In the temporal dimension, connectivity can be divided into sustaining initial connectivity versus the sustaining of on-going communications. Typically, the former requires the help of immovable infrastructure to relocate the moving host. As an example of a solution to this problem, DNS includes extensions [235] that allow hosts to update their new location in the DNS. In order to provide a stable "anchor" point, the infrastructure is also required to sustain the on-going communications of two hosts that move at the same time. When only one end-point of the communications moves at a time, support from the infrastructure is optional because the moving host can inform its other communications partners directly about its new whereabouts. This process, independently of whether it includes network intermediaries or not, is usually referred to as *handoff* or *handover*.

A *horizontal handoff* occurs when a device moves between homogeneous access technologies capable of supporting handoffs. This can occur in bridged Ethernet networks [224], or with a laptop that moves between two WLAN access points [162, 193] or when a cellular phone transitions between two base stations. In contrast, *vertical handoff* occurs when a mobile devices switches between different network types, such as WLAN and 3G. Some mobility mechanisms function only within a single address

[6]In contrast, site renumbering will be discussed later. While renumbering and network mobility could be solved with same technologies, the solutions differ in practice

domain, and this is called *local mobility* or *micro mobility*. In contrast, *global mobility*, or *macro mobility*, works across different domains.

Mobile IP is a classic example of a protocol supporting global mobility, albeit it has both a standardized version [183, 182] as well as a research variant [44] that is more optimized for local mobility. Originally, Mobile IP was not *address-family agnostic* [249, p. 22] at the network layer because it had one protocol supporting IPv4 networking [183] and another IPv6 [184]. However, Dual Stack Mobile IPv4 (DSMIPv4) [227] or Dual Stack Mobile IPv6 (DSMIPv6) extensions [212] will facilitate both IP versions within a single protocol[7].

In Mobile IP terminology, a moving host is denoted as *mobile node* and its communication partners are referred to as *correspondent nodes*. In general, Mobile IP supports host mobility by introducing a persistent, surrogate address for mobile nodes. The address identifies the node, and it is referred to as *home address*. Correspondingly, the mobile node is located using its *care-of-address*, which refers to its current IP address in the topological network topology. The home address and care-of-address are bound together by a network intermediary called *home agent* that relays traffic from the correspondent to the mobile node, thus maintaining the illusion for the correspondent node that the mobile node has a persistent address [149, p. 6]. The home agent [163] also relays the reverse direction[8]. When a home agent becomes unreachable, the mobile node can switch to an alternative home agent [138][9]. It should be noted that Mobile IP has also extensions to support network mobility [62, 150].

In contrast, MobileIPv6 supports direct communications between the mobile and correspondent node (route optimization) in the case where the correspondent supports MobileIPv6 protocol. Another difference in MobileIPv6 is that IPsec-based security is mandatory, which is typically managed with the help of the Internet Key Exchange (IKE)v2 [120].

While Mobile IP could be described as an industrial end-to-middle tunneling solution, a number of end-to-end solutions have been proposed by academia. For example, Location Independent Networking for IPv6 (LIN6) [139] splits an IPv6 address into two: the first 64 bits identify

[7] Another issue is IPv6 interoperability at the application layer, which is the topic of Section 2.2

[8] Early versions of Mobile IPv4 employed a so-called triangular (asymmetric) routing where the reverse direction did not involve the home agent. Also, another optimization called *foreign agents* is not usually deployed at all in practice

[9] This can disrupt existing transport-layer sessions unless the home agents are at the same LAN

the network attachment point (locator) and the last 64 bits identify the end-host. In essence, LIN6 introduces a new logical shim layer between the transport and network layers to serve two purposes. Firstly, the upper layers are isolated from the network layer changes because the shim layer translates the locator portion to a LIN6-specific prefix before delivering an incoming packet from the network to the transport layer. This identifier is referred to as the "generalized identifier", and the application can utilize it, e.g., for access control as it is immutable. Secondly, the shim layer can manage end-host mobility because it can translate the LIN6-specific prefix to a suitable locator before sending a packet to the network. LIN6 is dependent on extra infrastructure, called *mapping agents*, that the LIN6-capable end-hosts update regarding their current identity-locator bindings.

A later elimination approach called Identifier-Locator Network Protocol (ILNP) [21, 20][10] splits an IPv6 address into identifier and locator portions similarly to LIN6. In contrast to LIN6, ILNP does not provide a generalized identifier to upper layers, but rather exposes the possibly outdated locator portion. To avoid changing application semantics to interpret only the identifier part, ILNP instead proposes extensions to the Sockets API that use FQDN names to hide the details of addresses entirely from the applications. As another difference, ILNP reuses DNS rather than requiring new mapping infrastructure, and stores the mapping information in new DNS resource records. To secure the publishing of identity-locator bindings, the approach assumes the use of secure dynamic DNS updates [239] for the site adopting the approach. End-hosts can also inform each other directly about their changed network locations using ICMP(v6) messages, but DNS is utilized as a fixed re-contact point when two end-hosts relocate simultaneously. While the approach primarily targets IPv6 with its concrete ILNPv6 proposal, a separate ILNPv4 protocol is sketched that uses IPv4 options to carry the identifiers and requires modifications to the Address Resolution Protocol (ARP). ILNP can reuse IPsec to secure its data plane but requires changes to IPsec processing to ignore the locator portion of IPv6 addresses.

Internet Indirection Infrastructure (i3) [217] is another research-oriented architecture facilitating end-host mobility based on core-edge separation and the identity-locator split. As the name suggests, i3 achieves mobil-

[10] ILNP is founded on an even earlier approach called GSE which will be described in Section 2.1.2

ity by introducing an indirection infrastructure that hides the location of the end-hosts. The infrastructure consists of an application-based overlay based on a Distributed Hash Table (DHT). For two applications to communicate over the overlay, the server-side chooses an arbitrary identifier for itself and publishes a hash of its identifier along with its IP address in the DHT. Then the client-side application can deliver data to the server through the application-layer overlay using the published identifier. This way, the infrastructure decouples senders from receivers and allows applications to update their current location to the overlay. Based on the decoupling, i3 can further support anycast and multicast.

Forwarding directive, Association and Rendezvous Architecture (FARA) also decouples the identity from the location by abstracting the functionality of the different layers of the networking stack [60]. The architecture does not require any new namespace to be introduced, but it relies on overlay-based infrastructure to assist in mobility. As a concrete realization of the architecture, M-FARA [188] is based on the IP namespace but implements its own transport protocols and network intermediaries to support mobility.

It should be noted that many of the network-layer solutions for mobility remain marginally adopted and deployed. Thus, many application layer solutions have emerged to tolerate mobility at some level. For instance, many web browsers, including Mozilla Firefox, support pausing and resuming of downloads. Email client software such as Mozilla Thunderbird can tolerate disconnectivity and automatically reconnect to the email server. Internet telephony as supported by Session Initiation Protocol (SIP) includes session mobility [206]. Most web services identify HTTP sessions[11] with browser cookies and, thus, can tolerate IP address changes for non-streaming applications. In web browsers, the use of persistent HTTP, i.e., the reuse of the same TCP connection is less common nowadays [47, p. 3], perhaps to tolerate mobility better. Finally, more generic application layer solutions based on, e.g., libraries [252], by introducing a new session layer [210] and overlays [9] have also emerged but have not yet been embraced by application developers.

End-host Multihoming Solutions

In mobility scenarios, the host may be disconnected for a short period of time (*break-before-make*) during a handoff. However, multihoming scenar-

[11] According to some measurements [154, p. 1], HTTP traffic can amount to nearly 60% of Internet traffic

ios can involve the simultaneous availability of multiple network paths that may result in a smoother handoff because the host may be able to prepare an alternative path while sustaining communications using the active path (*make-before-break*). Similarly to mobility, multihoming can be realized at multiple layers of the networking stack. For instance, multihoming is present with multiple email accounts; sending an email using an unsubscribed email address will bounce off a mailing list. While the challenge persists at different levels, we focus on transport and network level solutions in this section.

As with mobility, multihoming solutions can be realized at end-host or intermediate hosts. When multihoming is implemented at network intermediaries such as routers, it is usually referred to as *site multihoming*. The purpose of such multihoming is to make use of network redundancy in order to dynamically switch from one network provider to another when a network fault occurs, but it can also be utilized for load sharing and other traffic engineering purposes [5, pp. 2-3]. For IPv4, RFC 4116 [6, pp. 4-7] documents a number of methods to facilitate site multihoming. The use of multiple Autonomous System (AS) numbers is another option, but this approach is problematic as the numbers are a finite resource. A more scalable way is to make use of PA addresses and advertise new routes on the Internet with the Border Gateway Protocol (BGP) when switching between ISPs. However, the drawback of this is that the site needs to renumber its network when changing its primary service provider. Site renumbering is the topic of the next section, and we focus on approaches dedicated solely to end-host multihoming in the remainder of this section.

Scalable multihoming is an important goal for the Internet architecture [152, p. 7]. When the goal is merely to sustain on-going communications for the sake of multihoming, non-persistent identifiers are sufficient [107, pp. 21-22]. At best, the Stream Control Transmission Protocol (SCTP) [177] is an example of the use of opportunistic identifiers[12]. SCTP makes no attempt to uniquely identify hosts but merely facilitates end-host multihoming, and also end-host mobility with later extensions [216], with routable IP addresses. SCTP introduces a new transport protocol [214] with separate API extensions [215] that applications can utilize. In contrast to TCP, SCTP can also support multiple streams within a single session and offers a messaging oriented API that avoids application-

[12]Individual streams inside SCTP sessions have stream IDs that are effectively ephemeral identifiers

level framing of messages.

Multipath TCP (MPTCP) [76] extends TCP to support end-host multihoming for both failure tolerance and load balancing purposes. Similarly to SCTP, it makes no attempt to introduce a new namespace. As it is based on IP addresses, it can be characterized as being based on opportunistic identifiers. However, a crucial difference to SCTP is that it can work with unmodified legacy applications, despite its optional extensions for the Sockets API [202].

As another approach, the identity-locator split has been proposed to decouple the existing transport and network layers from each other to facilitate a more generic form of multihoming [107, p. 13-14]. As an example, Site Multihoming for IPv6 (SHIM6) [175] is an IPv6-specific end-to-end solution that introduces a shim layer to the end-host stack. The shim layer does not introduce a new namespace but rather reuses routable IPv6 addresses as both identifiers and locators. This design choice allows a SHIM6-capable host to communicate with a SHIM6-incapable host. Consequently, SHIM6 does not require any changes to IPv6 applications even though it has an optional API for SHIM6-aware applications [127].

Site Renumbering Solutions

Site renumbering occurs typically when the site changes its network provider, e.g., to obtain more competitive prices[13]. The new provider will offer a different IP address range, and the site needs to be renumbered to correspond to the new prefix. Ideally, only DNS-based names would have been used internally by the site and by external sites to reference the various services of the local site. Then, the transition can be accomplished by merely updating DNS records and waiting for the cached entries to expire. However, this is not often the case, and IP addresses are embedded in various application, service, and infrastructure configurations [52, p. 34], and downtime is avoided at any cost.

In a nutshell, site renumbering can interrupt existing transport-layer connections, and it can also cause certain hosts to be completely unreachable due to address misconfigurations. The former problem can be solved using some of the end-host mobility or multihoming protocols described in the previous two sections and will not be further discussed in this section. The latter problem is more severe, and this section focuses on solutions that are applicable to it. Instead of ephemeral or opportunistic identifiers,

[13]At least in Finland, it is possible to change your cellular operator without changing your phone number. Unfortunately, this does work with ISPs at all

many of these solutions require more heavyweight means in the form of persistent identifiers to ensure that services remain reachable.

Another characteristic of the protocols is that many of them support at least site multihoming in addition to site renumbering. Many of the protocols also advertise themselves as solutions to routing scalability. In this work, we treat problems with routing scalability as a symptom or side effect of the missing renumbering support rather than the root of the problem; the actual source of the problem is a non-scalable choice to either support site multihoming or site renumbering, such as is the case with PI addresses. Thus, "routing scalability" will not be discussed in its own section but rather as a benign property of individual solutions to site renumbering.

The scalability of the Internet is challenged by an increasing amount of prefixes that aggregate poorly. According to RFC 4984 [160], there are multiple sources that are attributed to the problem. For example, some companies avoid renumbering costs by using PI addresses instead of PA addresses. Historical, non-aggregatable address allocations as well as multihoming and traffic-engineering tricks further aggravate the problem. Moore's law does not help to curb the costs for high-end routers because they employ a different type of memory than what is used in commodity hardware such as cellular phones. To recap, the RFC states that routers are becoming more expensive, routes are becoming flatter despite routing still being based on aggregatable algorithms[14], and the pursuit of local benefits has resulted in global scalability related costs.

RFC 4192 [28] describes procedures for how IPv6 renumbering can be realized manually without introducing service breaks. The renumbering includes prefix-related modifications to switches, routers, firewalls, DNS, DHCP, end-hosts and application configurations. However, RFC 5887 [52] states that renumbering is still difficult in practice due to the very nature of IP addresses. Despite IPv6 supporting multiple addresses and unique local IPv6 addresses [104] offering some relief during renumbering[15], manual renumbering procedures are still far from seamless. For example, TTL values in DNS should be set smaller well before the transition to minimize caching related problems. The TTL values are not visible to the applications because the Sockets API does not support such a concept as it was designed before DNS. Until browsers are restarted, many

[14]RFC 4984 points out that only a few feasible approaches to non-topological routing have been proposed [45, 8, 136]

[15]In addition to the various protocols for dynamic service discovery

of them employ so-called “DNS pinning”, which caches the addresses of the server to avoid security issues. In addition, routers may have to be restarted as they cache addresses, and server-side applications have to bind to multiple addresses for the duration of the transition period. Finally, the RFC also mentions that 34 investigated standards out of 257 had some dependencies to IP addresses in the protocol specification itself.

Independently of whether PI or PA addresses have been used for multihoming or site renumbering purposes in an edge network, injecting non-aggregatable addresses has introduced scalability challenges for global routing in DFZ [123, p. 1]. As a compromise, readdressing can be avoided in IPv4 by employing private addresses within the site while sustaining scalability with PA addresses [6, p. 7]. In IPv6, the same can be accomplished with Unique Local Addresss (ULAs) [106]. While these approaches support renumbering within an intranet, they need additional support to facilitate external communications. Without additional support (e.g. VPN tunneling), ULAs cannot be used for external communications at all. By default in IPv4, NAT devices drop incoming data flows unless some special arrangements are in place. For both incoming and outgoing data flows, NAT devices typically do not support support survivability of transport-layer connections [52, p. 17][16].

Evolution [152, p. 52-56] is an incentive-based strategy that starts with a number of shorter term traffic engineering schemes that require no coordination between sites and ends in a coordinated longer term plan. Instead of promoting a sudden “revolution” in routing architecture, the approach is phased so that each step gradually improves routing scalability and brings immediate benefits for its early adopters [123]. The approach starts from the intra-AS changes and ends with inter-AS changes to ultimately accomplish core-edge separation. The intra-AS changes involve algorithmic improvements to the software in local routers to achieve better Forwarding Information Base (FIB) aggregation locally.

In the second iterative step of the Evolution approach, new infrastructural indirection elements are introduced. The local AS deploys new routers, called Aggregation Point Routers (APRs), to facilitate *virtual aggregation* [30, p. 1] within its own AS. The idea is that the APRs split the entire IPv4 namespace artificially into large, virtual fragments, and each APR advertises its responsible block to legacy routers inside the AS. This

[16]NATs also conflict with Internet transparency, but this will be discussed in more detail in Section 2.1.2

way, the legacy routers of the AS survive with fewer route prefixes, and the complexity of finding more accurate routes is shifted to new APRs.

As a trade-off, the APRs naturally introduce path stretch in the sense of an extra hop. Another drawback is that the approach requires tunneling to avoid routing loops [30, p. 3]. The tunnel starts from the APR and is terminated at the egress site-exit router, which uses the extra information on the tunnel to make the precise forwarding decision.

While the intra-AS tunnels allow a single AS to evolve independently of others, it has some unnecessary overhead because each AS has to decapsulate and encapsulate packets in a hop-by-hop fashion, instead of merely forwarding the tunneled packets towards their ultimate destination. Fortunately, the overhead involved with this map-and-encap approach can be mitigated in the final phase of the Evolution, where pairwise agreements are expected to emerge between different ASes that employ virtual tunneling. The incentive is that two ASes can start advertising and peering virtual routes directly for each other so that the tunnels do not have to be terminated in the middle but rather are established between the two ASes. With time, the number of agreements grow and eventually the Internet gradually evolves towards a separate virtual routing name space.

In general, virtual aggregation introduces a new parallel routing namespace, and this mapping between two namespace needs to be managed somehow. The Evolution approach suggests reusing and extending BGP for backward compatibility and minimizing costs related to the mapping infrastructure. In contrast, the Location-Identifier Separation Protocol (LISP) working group in the IETF proposes a more revolutionary approach and directly focuses on the last phase of the Evolution approach. Cisco-driven LISP is a core-edge separation protocol and it implements the identity-locator split in border routers. LISP introduces a new protocol for the map-and-encap scheme [72]. For distributing mappings, multiple alternatives have been proposed. For instance, NERD [148] proposed a full mapping table for each mapping host, and the approach officially adopted by the working group, ALT [83], is based on partial tables organized into a hierarchical overlay. To reduce initial latency, ALT can be used for piggybacking data packets.

Besides improving renumbering and routing scalability, LISP-capable sites also support site multihoming without modifications to end-hosts[17].

[17]LISP extensions to support end-host mobility are being pursued at the IETF [73] that require similar modifications at the end-host as in the various approaches for core-edge separation

As trade-offs, LISP requires new mapping infrastructure, i.e., proxies, to interoperate with the legacy networks and the overhead for testing aliveness with other connected LISP routers as described in RFC 6115 [152, pp. 9]. The RFC also lists other protocols with different technical details based on core-edge separation and the map-and-encap schemes (such as Six/one [236], IVIP [240] and IRON/RANGER [219]), but LISP suffices here as a prime example of protocols in this category.

Global, Site, and End-system address elements (GSE) [176] serve as an early example of a locator-rewriting approach based on core-edge separation. The approach splits an IPv6 address into three pieces. The prefix of the address is denoted as *routing goop*, which can be changed by routers, and it essentially identifies a network. In the middle, *site-topology partition* can locally be used for defining different subnets for a site. The remaining part, *end-system designator*, is a globally unique end-host identifier generated from the Media Access Control (MAC) address of the host, for instance. GSE supports site renumbering because the end-hosts are identified with the end-system designator, and GSE-capable routers adjust the routing goop according to the local topology. The approach also proposes two new DNS records, one for the routing goop, and another for the site-topology partition and end-system designator.

GSE does not fully conform to Internet transparency because the routing goop inside the site may be different from the outside, effectively requiring a split DNS. At the transport layer, the routing goop and site-topology partition should be excluded from the pseudo-checksum calculations. RFC 4984 [160, pp. 18-20] also notes that the end-system designators have changed the semantics of some applications that compare addresses for equality because they have to compare only the designator part. The RFC further criticizes GSE for an undefined locator-failure discovery and raises the question of compatibility with Cryptographically Generated Address (CGA) [23] addresses. The issues with GSE are further analyzed by others [255].

Internet Transparency Solutions

While waiting for IPv6 to become ubiquitous, the IPv4-based Internet is no longer transparent any more since different kinds of middleboxes have broken the original end-to-end nature [201] of the Internet. Firewalls are examples of such boxes, and some protocols, such as Skype for Internet telephony, have been known to work their way around the firewalls. NAT

devices also include the functionality of a firewall to filter out traffic flows originating from the Internet, but NATs also support address aggregation with private address realms. The aggregation requires the NAT middleboxes to couple transport and network layers tightly[18] as the translation information is stored in transport-layer ports. As NATs constrain connectivity to the client-server model, and private address realms make unique identification of hosts based on IPv4 addresses impossible, development of peer-to-peer applications has become more complicated. Consequently, several solution to work around the issues with NATs have emerged.

RFC 2775 [51] describes two practical solutions to achieve better interoperability with private address realms. In the first approach, *split (horizon) DNS* can be used to give a more concise view of a site from the standpoint of FQDNs. The idea is that the DNS returns private addresses for host name queries originating from the private address realm of the site and public addresses from outside. Thus, the host names are consistent but map to different addresses depending on the location of the querying host. Nevertheless, this approach does not comply with Internet transparency from the viewpoint of addressing and presents issues for applications caching addresses. The second approach is not problem free either. Application-Level Gateways (ALGs) can modify application-layer protocols to retrofit them with NATs. For example, an ALG is needed when FTP is used in active mode because the FTP server creates a new TCP connection back to the client located behind a NAT device. ALGs can be problematic, especially when they are not properly implemented [53, p. 12]. The complexity of ALGs have been investigated, and alternative types of NAT architectures have been proposed [40] but never adopted.

By default, NATs block transport-layer connections originating from the Internet unless the operator of the NAT manually opens certain ports. However, this requires some expertise on the part of the user, and the port can be forwarded to only a single host inside the private address space due to multiplexing reasons. For instance, only a single host can serve the standard HTTP port for the outside.

To avoid the manual tweaking of the NAT device, two approaches have emerged. First, applications can employ a Universal Plug and Play (uPnP) library to request the opening of certain ports in NATs [80]. The protocol is widely supported by different NAT device vendors although it can

[18] As also mentioned by others [249, pp. 53-54], transport and network layers are not only coupled at end-hosts but also at intermediary hosts as a result of NATs

sometimes be disabled by default. It is also supported by various manufactures of home multimedia devices and smart phones because it supports broadcast-based discovery of local-area services[19]. As a second approach, IETF is standardizing Port Control Protocol (PCP) [243]. Among other things, it improves upon uPnP as it can infiltrate through multiple cascading NAT devices.

A vendor-specific solution called "Back to my Mac" (BTMM) service [59] supports NAT traversal for OS X with the help of uPnP [80] and other protocols. BTMM introduces topologically-independent, ULA-based addresses [104] that can be used for addressing hosts behind NATs. The approach employs DNS to support end-host mobility and easier naming of hosts, and a combination of Kerberos, IKE and IPsec to facilitate security. BTMM inherits its restriction from uPnP; NAT traversal fails when uPnP is disabled from the NAT or in the presence of cascading NATs. Also, BTMM supports only IPv6-capable applications.

In uPnP and PCP, the idea is that the application requests the NAT to open and forward a port to it. As the success of this depends on the protocol support in the NAT(s), alternative approaches have emerged. The protocol family of Session Traversal Utilities for NAT (STUN) [198], Traversal Using Relay NAT (TURN) [153], and Interactive Connectivity Establishment (ICE) [197] works around the issue and penetrates through the NAT box(es). The trick is that two communicating applications simultaneously send transport-layer datagrams to each other in order to create state in their NAT middleboxes, thus bypassing ingress filtering in the NATs [213]. The success of this NAT penetration procedure is not always guaranteed because the deployed legacy NATs do not operate in a uniform way [22, 87], and especially TCP [213] penetration has somewhat smaller chances than UDP in practice. However, the UDP-based transport has a higher chance of succeeding than TCP [88, 77].

As a brief summary of the protocol family, an end-host uses STUN to learn its address-port mappings from the STUN infrastructure deployed in the Internet. The ICE protocol uses and extends the STUN protocol format to support penetration of on-path NATs. The end-host may also utilize a TURN relay to guarantee successful NAT traversal in the event the penetration fails. While ICE restores end-to-end connectivity, it can be argued that it does not support Internet transparency in an ar-

[19] Another service discovery protocol from the IETF is Service Location Protocol (SLP) [92]

chitecturally clean way as it does not untangle addresses from the ports. ICE is pervasive in the sense that it requires the application to use its APIs. ICE also changes the semantics of the application-layer protocol because the application has to exchange the address-port mappings, trigger the ICE penetration procedure, and demultiplex STUN traffic from its own application-layer traffic. Nevertheless, the protocol family has been adopted to SIP [41], P2P-SIP [114] and Real Time Communication on the Web (RTCWEB) [12], to mention a few examples.

Teredo [105] is less pervasive than ICE because the application does not need modifications for it[20]. In Teredo, the application uses a special surrogate IPv6 address when NAT traversal is desired. The local Teredo software tunnels transport-layer traffic sent to the virtual address over UDP and tries to penetrate NAT devices transparently from the application. The penetration procedures are similar to ICE although Teredo has a somewhat lower probability of successfully establishing end-to-end connectivity. Teredo supports Internet transparency in an architecturally cleaner way because Teredo introduces a new virtual namespace that hides the address and port translation details from the application. However, a Teredo-based address is not persistent because the address is formed in a topological-dependent manner. As further limitations, the Teredo requires an IPv6-capable application at both ends, and the tunneling obviously involves a small penalty for the MTU.

To restore Internet transparency, an early research-oriented approach called 4+4 [228] extends the NAT-based architecture of the Internet. In this approach, a host is uniquely identified both using its private IPv4 address and the public IPv4 address of its NAT. At the network layer, 4+4 employs stateless IPv4-over-IPv4 tunneling to store both of the address types, and 4+4-upgraded NAT devices translate the addresses.

The described implementation intercepts DNS requests at the client-side host, injects new requests for 4+4-specific Service Record (SRV) records to detect 4+4-capable servers. For 4+4-capable servers, the implementation caches the mappings between a private and public address, but it modifies the DNS responses so that the originating legacy application receives the private address. This way, a site can choose its internal addressing convention, and the changing of the ISP can become easier. Also, private address spaces aggregate better and thus improve routing scala-

[20]On older Windows versions, the application had to enable Teredo explicitly using a socket option

bility.

As another research solution to Internet transparency, mobility and multihoming, Delegation-oriented Architecture (DoA) [237] proposes an architecture based on core-edge elimination. The idea is to introduce a new shim layer at the end-host between the existing transport and network layers that translates upper-layer identifiers to routable locators. The approach is based on tunneling; the identifiers are stored in an additional header inserted between the transport and network layers. The architecture revives Internet transparency with its persistent identifiers, and the tunneling approach shields application port numbers from modification of NATs. Further, DoA-capable NATs are required to rewrite only the IP addresses in packet headers and use the additional identifier information from DoA headers as additional demultiplexing tokens.

DoA requires the identifiers to be globally unique but not necessarily cryptographically derived. The identifiers are flat and such non-hierarchical identifiers are difficult to look up from the DNS, so the architecture relies on a DHT service for storing them. DoA introduces extensions to the Sockets API to accommodate the 160-bit identifiers as they cannot fit existing structures.

However, what really makes the DoA architecture distinct from many other proposals is the support for secure, loose source-address routing. To accomplish this, an end-host announces its own identifier as well a chain of its middlebox identifiers in DHT. In order to the reach the end-host, other hosts have to follow the chain in the specified order. Each of the middleboxes includes cryptographic information in the traversed packets so that the end-host can verify the chain. This facilitates both on-path and off-path intermediaries that can support, for instance, virus scanning and firewall services. Off-path intermediaries can be useful for Distributed Denial of Service (DDoS) prevention as the intermediaries take the hit instead of the end-host. However, the trade-off of such delegation is a fairly complex DHT registration procedure and involves extra interaction between the intermediary hosts.

NUTSS [89] architecture is also based on core-edge elimination, but unlike in DoA, NUTSS-based applications should be using more aggregatable Universal Resource Identifier (URI)-based names instead of flat names. A fully-fledged NUTSS introduces a new API for network applications, but the implementation also supports a legacy-compatible mode where a shim layer translates routable IP addresses from the application

layer into NUTSS-based identifiers. Similarly to DoA, this architecture also achieves mobility, multihoming and Internet transparency by providing applications with persistent identifiers. However, it also holds the promise of supporting anycast and multicast.

In NUTSS, transport-layer connections are ephemeral and recreated on demand. In fact, NUTSS acts as a network-application framework that abstracts away the lower-layer details and paves the way for extensibility because a NUTSS-capable end-host can negotiate support for different protocols (IPsec, TLS, IPv6) using the control plane of NUTSS. The control plane supports steering of middleboxes; similarly to DoA, the end-host registers explicitly with middleboxes that can reside on or off path[21]. In contrast to DoA, NUTSS-based middleboxes handle either the control or data plane. Control plane boxes are handled by off-path "p-boxes" that are organized into an overlay and can be used to negotiate, for instance, access-control policies. M-boxes are typically on the path and they forward the data plane when it is permitted by the p-boxes. As the p-boxes and m-boxes communicate with each other, they can also support distributed firewalls for those problematic cases where the routes used between two end-hosts are asymmetric. NUTSS also suggests the use of STUN to establish direct end-to-end communications in the face of legacy NAT devices.

The crux of the NUTSS architecture is the deployment of the new infrastructure. To mitigate this, the authors propose a three phase deployment plan. In the first stage, public p-boxes are deployed and a few end-host applications employ NUTSS with the NAT traversal capability as the driving "killer application". Then individual networks deploy their own p-boxes, and end-hosts learn about their existence via DHCP extensions. Finally, legacy middleboxes are replaced with m-boxes that also act as proxies for communications for the remaining legacy end-hosts.

2.2 Heterogeneous Addressing

One of the fallacies in distributed computing is to assume that the network is homogeneous [84]. This applies to addressing, especially now that IPv4 and IPv6 will have to co-exist during the undetermined transition period to IPv6. The global adoption of IPv6 has not been the only sug-

[21]The explicit negotiation between all intermediaries in both DoA and NUTSS appears somewhat reminiscent to circuit-switched networks

gested way to deal with heterogeneous addressing since some approaches advocate the replacing of IP addresses with DNS-based names in applications. A few heretics have embraced heterogeneity instead of the homogenization of application layer addressing.

2.2.1 Challenges

Heterogeneous addressing as introduced, for instance, by the separation of IPv4 and IPv6 is problematic for a few reasons. For instance, access control rules double, both at the end-hosts and middleboxes, leaving more room for human error. IPv4 address literals are easily forgotten in software configurations of a site and are discovered only when the site transitions completely to IPv6 (or renumbers itself otherwise). Writing of networking software is more complicated as developers have to deal with dual versions for DNS and for the actual data transfers. The development is especially hard because the Internet is still in a transition phase: a client may discover IPv6 addresses for a server, but it is not guaranteed that the client-side network is capable of supporting IPv6 or that the individual service at the server supports IPv6. The client can try each compatible address pair sequentially until it finds a working combination, but the user may have aborted the connection by then due to the additional timeouts. Thus, developers may be tempted to avoid IPv6 altogether, or they have to employ parallelization for network connections, which creates further unnecessary traffic on the Internet.

It could be said that heterogeneous addressing involves a challenge similar to multihoming with its multiple addresses. However, heterogeneous addressing is even more pervasive at the application layer because the Sockets API does not insulate applications from the different address formats. Naturally, network application frameworks and other middleware can be used for insulation, but a host of software has already been written without them.

Finally, it is worth mentioning that the heterogeneity does not only stem from the introduction of IPv6. For instance, wireless sensor networks are comprised of constrained devices with limited memory, processor and battery lifetime. In such environments, the IP-based stack has been considered too inefficient and purpose-built stacks have been utilized[22]. However, some attempts to assimilate sensor networks with IPv6 have already

[22]For instance, the proprietary Z-wave protocol for sensor networks is not based on TCP/IP

emerged [140, 207].

2.2.2 Solutions

While it is possible to implement software agile enough to accommodate both IP versions, it appears that a substantial portion of the networking software is still oblivious to IPv6, and address literals are hard coded into software configurations. Therefore, it can be argued, based on existing practices, that agility for different address families is difficult to achieve in the present TCP/IP architecture. Thus, mono-cultural attempts to unify address formats have emerged, for instance with IPv6-mapped [103, p. 10] addresses as a replacement for IPv4 addresses, but these have all failed [159].

The IETF community has invested a lot of energy in global IPv6 adoption. As a result, a number of different transitioning solutions have been introduced. To mention a few such approaches that would facilitate IPv6 at the application layer, there is the example of Teredo, which has been described earlier, as well as various other tunneling mechanisms, including manually configured IPv6-over-IPv6 tunnels, 6-to-4 Tunneling Protocol and its extension IPv6 Rapid Deployment on IPv4 Infrastructures (6RD), Multiprotocol Label Switching (MPLS) (MPLS)-based tunneling, Network Address Translation/Protocol Translation (NAT-PT) and its enhancement NAT64. Instead of iterating through all of these standards, we point the interested reader to a overview [108, p. 22-47], and we focus here instead on NAT64 as it has lately received a lot of attention in the IETF.

In NAT64 [26], the client-side host (or its network) supports only IPv6 and the server supports only IPv4. To interoperate between these two incompatible address families, the client has to be located behind a NAT device supporting the proposed extensions and to use a DNS server (or proxy) supporting DNS64 [27] extensions. When the client looks up the address of the server over IPv6 from the DNS, it notes that the server has only an A record configured and decides to synthesize an AAAA response based on a special IPv6 prefix and the address of the server. This avoids the address incompatibility issue, and the NAT64 device transforms IPv6 packets originating from the client to IPv4 packets for the server, and vice versa. The other benefits of this approach are that it introduces IPv6 to the client-side networks, which NAT deployment would typically prevent, and also, for example, homogenizes the client-side access control lists to IPv6. As a drawback, the approach fails with the software that employs

IPv4 address literals [17, p. 14].

As an alternative to assimilating applications to use IPv6 addresses, the NUTSS architecture, as described in the previous section, proposes URIs as identifiers in its Sockets API extensions. Others have also proposed DNS-based identification [81, 58]. One of most recent ones, Name-based Sockets (NBS) [230] is still somewhat immature, but is being standardized in the IETF. Unlike NUTSS, it takes a bit of a more conservative approach and uses only the FQDN portion as the persistent host identifiers. In a nutshell, NBS suggest new Sockets API structures that can hold names instead of addresses, and the name resolution occurs inside NBS module, not in the application. Thus, NBS approach offers a solution for homogeneous naming by reusing the existing DNS namespace.

The NBS architecture can support also mobility and multihoming. It defines its own control plane for mobility management that tries to minimize round trips by piggybacking it into data-plane packets. This design choice limits it to IPv6 because IPv4 options appear to be dropped by a number of existing firewalls [157, p. 339]. Other design constraints exist as well, for example, since modifying of all application to use NBS extensions is not trivial, and some hosts do not have names in the DNS [220, p. 16]. RFC4177 [107, pp. 20, 31] mentions a few other constraints with FQDN-based names. Firstly, services typically employ load balancing by attaching a single FQDN to multiple addresses that belong to different hosts. This means that the FQDN does not uniquely identify a single host. Secondly, FQDN-based identifiers may not survive business mergers or acquisitions as the domain name may change[23].

In contrast to the attempts to unify applications to use either FQDNs or IPv6 addresses, the abstract Plutarch [61] architecture embraces heterogeneous addressing. Plutarch does not try to modify existing Internet architecture but rather nests network end-hosts and intermediaries into two abstract entities. The smallest entity is a *context* that refers to a set of hosts with homogeneous networking capabilities, such as addresses, packet formats, transport protocols or a common service for name look up and storage. As two concrete examples, the context can be a private address realm or an autonomous system. Then, an *interstitial function* chains two heterogeneous contexts together to enable inter-context communications. As examples of interstitial functions, uPnP is its real-

[23]While the reuse of FQDN-based identifiers avoid new infrastructure, it could be further argued FQDNs are domain specific and a host should not preserve its FQDN it moves to another domain

ization for private address realms, and BGP routing procedures for autonomous systems. The interstitial function translates addresses and names, transport-layer protocols or even acts as an ALG to modify data for more suitable formats, for instance for hand-held devices, according to the needs of the underlying context. While this is nothing new, the novelty of Plutarch is that a universal programmable API is proposed for end-hosts to chain contexts with interstitial functions with the ambitious aim of achieving communications across heterogeneous networks. The authors present a sketch of the API for Plutarch but admit it is still a straw-man approach (as it remains unimplemented).

2.3 Insecure Addressing

Insecure addressing is present at all layers of the network stack. In this section, we briefly introduce the challenges and a number of solutions. The solutions are categorized into client-side and server-side authentication, communication privacy and availability. It is worth noting that we merely scratch the surface; many interesting topics, such as anonymity, object/data-centric security, intrusion detection and quantum cryptography are worth another dissertation, and hence will be beyond the scope of this particular dissertation.

2.3.1 Challenges

Insecure addressing is present in the network and link layer in the TCP/IP architecture. IP addresses, as well as MAC addresses, can be forged because typically they do not include a secure verification of the ownership. When the attacker is in the same network as the victim, the attacker can change its own address to correspond to the MAC or IP address of the victim host. As an alternative, the attacker can intervene in the look-up procedures. With MAC addresses, the attacker can employ ARP spoofing. With IP addresses, the attacker could impersonate the DHCP or DNS server.

An attacker could also try to circumvent IP-based access control measures of an application running on a multihoming victim host. Let us consider that the victim host has a “trusted” and “untrusted” network interface, both of which are configured with their own unique IP addresses (correspondingly referred here as the trusted and untrusted address). In

such a scenario, the application developer binding the application to the trusted address may assume that the application receives traffic solely from the trusted network interface. However, this assumption is false when the underlying host employs a so-called *weak end system model* [42, p. 63]. For instance, the attacker's host may abuse this loophole by sending traffic destined to the trusted address, but by routing the packet through the untrusted interface of the victim host [220, p. 15, 19]. Thus, the attacker can bypass the IP-based access control measure of the victim's application because the underlying networking stack is based on the weak end system model.

2.3.2 Solutions

While IPv4 is prone to ARP spoofing, the issue is resolved in IPv6 using SEcure Neighbor Discovery (SEND) extensions [16]. However, neither IPv6 or the extensions are widely deployed. As MAC and IPv6 addresses still remain easy to forge, they are far from ideal as authentication tokens, and the problem is pushed up in the networking stack in many scenarios. As transport-layer security is not really deployed in the Internet, the authentication is usually implemented in the application layer. The layer where security is implemented also defines the granularity, that is, application-layer solutions typically enjoy finer grained granularity than solutions operating on a lower layer.

Instead of modifying individual applications to support security, low-level solutions can be used to protect entire legacy networks more efficiently, albeit not without trade-offs. For instance, IP-based firewalls or Virtual LAN (VLAN) tagging in routers can be used to isolate networks from each other. However, they offer little protection against "insider attacks" [34, p. 12-13], i.e, a user can be tricked into installing malware on the underlying computer that bypasses the firewall and to further infect the entire network. In other words, such firewalls offer only topological protection. This can be challenging with mobile hosts without permanent IP addresses and, in practice, it is common to use a VPN or a web proxy to access a firewalled intranet. Also, another challenge with firewalls are multihoming sites that can result in asymmetric paths.

Contrary to low-layer security, a benefit of application-layer security is that the application is aware of security and can convey this information to the user. Typically, it is also easier for the user to carry the security credentials, such as passwords, However, application-layer security inherits

all of the weaknesses of the lower layers. Implementing security redundantly at multiple layers offers more protection but can have a negative impact on performance.

In practice, a number of factors limit the impact of attacks against addresses, at least when the attacker is off the communication path. For instance, the NAT devices drop new traffic flows arriving at the private address realm by default. Then, since the identity and location of a host are coupled, a malign host cannot claim to be the victim host, at least when it resides in another network than the victim. Source address spoofing can be difficult to achieve as many routers and firewalls drop packets originating from the incorrect network. However, a compromised router or a WLAN access point allows man-in-the-middle attacks, and the above mentioned measures cannot protect against such on-path attacks.

Client-side Authentication

A client-side host can be authenticated by the server or by any of the on-path middleboxes, such as local wireless access point, router or firewall. Section 2.1.2 also mentioned DoA and NUTSS, two research-oriented approaches that supported off-path intermediaries, so they will not be further discussed here.

As MAC-based and IP-address based access control can be easy to circumvent, it is possible fortify them with other means. For instance, a host can generate a private-public key pair and hash the public part of the key along with some additional parameters to generate a self-certifying IPv6 address. This technique is called CGA [23] and is employed in IPv6 by SEND [16]. While SEND protects the ownership of an address, it does not prevent a host from claiming an unreserved address, possibly even belonging to an entirely different topology. However, such exploitation can be prevented with Source Address Validation Architecture (SAVA) [246] by verifying the source addresses of egress packets at the varying levels of routers or by requiring the end-hosts to cryptographically verify the packets they send.

At home, it is common to employ password-based authentication for the wireless access point using Wi-Fi Protected Access version 2 (WPA2)-based security [1]. For organizational and corporate use, a myriad of protocols exist [3, 166, 196, 46, 7, 18] both for wired and wireless authentication. In such environments, the user is usually directed to the web portal to input user credentials. Further, the authentication may

be valid across multiple web-based services if the sites collaborate with Single Sign-On (SSO). Again, various different schemes to implement SSO exist [2, 93, 222].

A few other techniques could be mentioned as well. In Identity-based Cryptography (IBC), any publicly-known string, such as email or even an IP address, can be used to represent the user's public key for signing or encryption [25]. Must of the complexity of public key management and certification is hidden into a Trusted Third Party (TTP) service. In contrast, purpose-built keys [43] require no infrastructure but are merely ephemeral identifiers based on public keys that merely ensure that two communicating end-points remain the same throughout the communication session [149, p. 9]. The widely deployed Trusted Platform Module (TPM) [125] is designed for the storing of sensitive information, such as private keys, on the tamper-proof hardware residing on the end-host. This way, applications can request the hardware for signatures, but a compromised software cannot steal the private key.

Server-side Authentication

TLS [63], and its predecessor, SSL [82], are the de-facto way for securing and authenticating TCP transactions, especially on the web. While TLS also supports client-side authentication, only the server is typically authenticated by the client, and some of the methods described in the previous section are used for authenticating clients. TLS authentication is based on certificates signed by the Certificate Authorities (CAs) in the Public Key Infrastructure (PKI) hierarchy. Consequently, TLS meets the challenge of avoiding server-side impersonation attacks, albeit not offering any remedy to non-persistent or heterogeneous addressing. Koponen et al [132] have implemented client-side mobility extensions for OpenSSL, an open-source implementation of SSL/TLS, but the extensions were not officially adopted in the implementation nor standardized.

TLS runs on top of TCP. The TLS handshake, which is followed by the TCP handshake, requires two round trips in the basic case, or one round trip when a connection is resumed, i.e., when the client has cached information during a previous session[24]. TLS requires a different port when a service supports both insecure and TLS-secured communications in order to avoid man-in-the middle attacks [63, p. 34]. The application has to be modified to use the TLS-specific APIs instead of Sockets API, and

[24]Google has proposed some extensions for the Chrome browser to reduce the connection set-up latency associated with SSL/TLS [147, 146]

in this way the application is always aware when the TLS-based security is being utilized. The APIs are implementation specific and a number of implementations exist, including open-source libraries OpenSSL and GnuTLS.

dTLS [194] offers protection for UDP-based communications. In essence, it is an adaptation of TLS to the limitations of UDP that does not guarantee packet delivery nor ordering. Therefore, its security and addressing characteristics are the same as in TLS. While the datagram-oriented nature of UDP does not prevent applications from sustaining address independent data flows, security is still a concern. Thus, secure extensions to support mobility in dTLS have been defined [205].

FQDN-based names can be considered as a means to authenticate services, but basic DNS offers little protection, particularly against man-in-the-middle attacks. To fill in this gap, DNSSEC [15] cryptographically authenticates DNS responses, albeit it does not really assure anything about the "identity" of the service (which is usually accomplished with TLS). With support for certificates, DNSSEC has the potential to be used as a PKI when it is more widely deployed. It is also worth noting that hosts may also dynamically update their own records in the DNS [235] in addition to using look-up functionality, and extensions for securing the updates exist as well [239].

Protection of Communication Content

A malign middlebox can breach the confidentiality of the data by reading the content of unencrypted datagrams or modify individual datagrams, whereas a malign end-host can try to forge the originator of the data or replay recorded datagrams. TLS and Datagram Transport Layer Security (dTLS), as already described in the previous section, can protect against this at the application layer. Alternatively, SSH [251] or IPsec [121] can also be used.

In SSH, a server creates a private-public key pair for itself and is authenticated using its public key. A client authenticates itself to the server using a username-password combination or with a user-specific public key. Instead of showing entire public keys, for the convenience of the user SSH prompting has employed a hash calculated over the public key as a fingerprint. SSH secures communications between the client and server using symmetric keys that are created using a Diffie-Hellman (DH) key exchange. The SSH implementations support at least terminal sessions

but typically also VPN-like tunneling of any kind of traffic that has to be set up manually by the user.

SSH requires a client-side and server-side application to be installed but does not require any additional infrastructure, to which its wild success can be largely been attributed. Albeit it has optional DNS records [203] for storing fingerprints, the records are not commonly used and therefore SSH offers weaker security based on Leap of Faith (LoF) [19, p. 5]. The weakness is that it is prone to man-in-the-middle during the first connection attempt, during which the client learns and caches the public key corresponding to the host name and IP address of the server. When no prior key exists or the key has changed for the server, the client-side software prompts and warns the user. This is a crucial aspect, not only because the user can verify the key of the server but most importantly the prompting protects against further attacks. The middleman has to be on the path every time to avoid being exposed, and thus the approach inflicts an *asymmetric cost* [19, p. 5] on the middleman.

Besides asymmetric cost, LoF relies on *temporal* and *spatial separation* [19, p. 3,5]. Temporal separation guarantees only that the server remains invariant but does not guarantee that it was the correct server the first time. Spatial separation offers an assurance that the host is on a specific communication path. In the case of SSH, this means that the public key of the host is coupled with its host name (or IP address). If this binding changes, the SSH connection fails. Therefore, it can asserted that security based on spatial separation as employed by SSH does not really support persistent addressing. Client-side mobility extensions for OpenSSH, an open-source implementation of SSH, have been implemented to recreate new TCP sessions with the server [132]. Unfortunately, the extensions have not been adopted in the implementation nor standardized. As an alternative approach, Winstein et al [245] have redesigned SSH from scratch to sustain mobility for terminal sessions and to support better interactivity with the user on top of UDP.

IPsec offers network-level protection for data flows. Compared to SSH, applications are not typically aware of when the communication was secured. IPsec is based on symmetric key cryptography based on unidirectional keys that are denoted as Security Associations (SAs). As setting up manual keys can be cumbersome for users, this task is usually automatized using a key-management protocol that negotiates the keys dynamically as when an application sends traffic matching a certain Security

Policy (SP).

IKEv2 [69] is typically used as a gateway based VPN that does not require any changes to the server side. The protocol has two phases, where the first phase is a DH key exchange that sets up symmetric keys to secure a control plane. The second phase uses the control plane to set up symmetric keys for the data plane, which is a IPsec-based tunnel between the client and the gateway. Basic IKEv2 does not support end-host mobility or multihoming, but Mobile Internet Key Exchange (MobIKE) [126, 68] extensions fill in this gap for the client side.

The basic version of IKEv2 is based on "strong" authentication that essentially requires a separately deployed PKI. To avoid the management overhead and scalability concerns involved in PKI, so-called Better Than Nothing Security (BTNS) [223, 242] has been standardized. BTNS offers two methods of operation with their own trade-offs. Stand-Alone BTNS offers the lowest level of protection to subversion by a man-in-the-middle attacker. This method merely guarantees that the other entity does not change during communications and is based on anonymous encryption [19, p. 4]. As such it is mostly suitable to be used with public services in the absence of a stronger authentication at the network layer. Channel-bound BTNS avoids middleman attacks but assumes that the application layer supports strong authentication, thus making strong authentication unnecessary at the network layer. To communicate the success or failure of this authentication to IPsec, a separate API between the application and IPsec is needed [223, pp. 6-8] to "bind" the security mechanisms together, which is also referred to as *channel binding*. Further, the same API can be used for fortifying BTNS-based security to introduce transport-layer specific SAs. This process of *connection latching* [241] ensures *fate sharing* so that IPsec associations are purged when the corresponding transport-layer connection terminates in order to avoid certain abuse [186, pp. 342-343]. As the application can prompt the user and cache information on unauthenticated credentials on behalf of BTNS, this method can achieve LoF security similar to SSH [223, p. 23],[186, p. 347].

Prevention of Unwanted Traffic

Unsolicited traffic is a nuisance at many levels of the networking stack. At the application layer, e-mail spam[25] is a nuisance that directly involves the end-users. Traffic flooding using Denial of Service (DoS) or DDoS [161]

[25]Countermeasures for VoIP spam [54] are outside of the scope of this dissertation

does not directly involve the end-users but is visible to them as a degradation of QoS. In this section, we take a brief glance at a few solutions to email spam and DoS prevention.

A recent survey [54] describes a number of spam prevention techniques. To mention a few examples, black listing assumes that email relays are benign by default, and email relays end up on the blocking list when they are reported for sending spam. White listing works exactly in the opposite way and assumes everyone is untrustworthy by default. It is common to employ machine learning techniques, such as Bayesian email filtering, either at the email clients or by the email service provider in the case of web-based email. In greylisting [137], a server receiving email will request the originator to retry after a while, and this is effective as spam relays do not usually retry.

The spam problem has been analyzed also from an economic perspective by others. Goodman et al [85] model costs for spam prevention with human-interactive proofs[26] and computational puzzles. The authors prefer the latter method and show that it is not necessary to persist in a proof or a puzzle forever for each email message sent, but it merely suffices to apply it only in the beginning for every Nth message. Finally, Levchenko et al [119] show that the payment infrastructure is the bottleneck of the spam value chain and argue that spam could be tackled most effectively by political means, i.e, enforcing a payment tier.

Packet Level Authentication (PLA) [56, 145] mitigates DoS attacks with public keys and certificates. The protocol introduces a new shim layer between transport and network layers at end-hosts. When a host sends a packet, the shim layer signs the packet and attaches a certificate[27]. This way, PLA-capable routers (or recipients) can authenticate and authorize the packet, based on TTP that certifies the public keys of the end-hosts. For instance, this facilitates source address verification, and DoS attacks can be prevented by revoking of the certificate. As such, PLA does not change the addressing model of the Internet in any way but remains compatible with other approaches that, for instance, implement end-host mobility [144, p. 29].

[26] Also known as the Completely Automated Public Turing test to tell Computers and Humans Aparts (CAPTCHAs) or Reverse Turing Tests

[27] Alternatively, the certificate can be omitted from subsequent packets if the routers cache it

2.4 Deployment Considerations

RFC 5218 [221] characterizes a number of the properties of successful protocols. One critical factor is that the early adopters should get the benefits of using the protocol. As in Mobile IPv4 or IPv6 (MobileIP) and IKEv2, one way to meet this goal is to employ gateways or proxies that terminate the client-side connections so that for users there is no dependency on server-side deployment. On the other hand, a complete avoidance of additional mandatory infrastructure can also be a recipe for success as has been the case for SSH.

It is should also be noted that IETF always mandates security considerations from the standardized protocols. This is required even when the target scenario would be trusted – wildly successful protocols can easily become reused beyond their original purpose [34, p.5].

While a full economic analysis is outside of the scope of this manuscript, it is difficult to escape it entirely in deployment considerations. Regarding routing scalability, Jen et al [112, p. 4] argue that the cost of deployment is better aligned to the benefits with core-edge separation because it is the transit networks that are facing the scalability problem. However, a deployment at the edges of at least two sites is required, even with a core-edge separation to gain some benefits of the adoption. Then, the main difference with core-edge elimination is that it has to be deployed for all the end-hosts of the two sites[28]. Thus, adoption of an elimination approach can be considered slower and, as suspected by Jen et al, may arrive too late as the routing tables may exceed a critical threshold. In the context of this dissertation, elimination approaches will nevertheless be accepted as a viable technical solution for site renumbering despite this claim.

Another protocol design consideration is incremental deployment without a flag day. This usually requires backward compatibility on end-hosts, even with IPv6 stacks that are already considered legacy [160, p. 28].

For realistic deployment of a protocol, the constraints as posed by middleboxes should be considered in the protocol design. For instance, the Internet is still ossified by IPv4 [11, p. 206] even though IPv6 is slowly making some progress[29]. NATs and firewalls pass only TCP and UDP

[28] Deploying of separation-based approaches might become easier if software-defined networking, as promoted by OpenFlow [156] for example, becomes more popular

[29] Around the globe, the world IPv6 day was organized for the second time on 6th of June 2012 to permanently enable IPv6 in the products and services of major

traffic by default [220, p. 16], and in some cases only HTTP traffic on top of TCP [191]. By default, NATs filter incoming connections, which requires NAT penetration procedures in the case of P2P applications. Some firewalls also drop ICMP messages, and a majority of firewalls drop IPv4 options, but TCP options are not usually filtered. Designing protocols that pass IP address literals by default is a doomed idea due to ubiquitous NATs.

A common misnomer about DNS is that deploying new records to DNS is difficult because it requires modifications to DNS software. On the contrary, modern DNS software is actually quite flexible. For instance, the most popular DNS service implementation, *Bind*, supports non-native DNS record types specified in a (hexadecimal) binary format.

2.5 Host Identity Protocol

As most of the collection of articles for this dissertation are based on HIP, it will be introduced in more detail than the other protocols in this section. HIP [174, 169, 90] is an approach based on the paradigm of the identity-locator split, and the conventional use of HIP classifies it to the core-edge elimination category. It introduces a cryptographic namespace to identify end-hosts, and the namespace is managed by a new shim layer between transport and network layers.

The HIP working group has been standardizing the protocol in the IETF and is in the process of moving the experimental RFCs [164, 165, 116, 142, 141, 171, 170, 129] to the standards track[30] at the time of writing. In a nutshell, the most important updates are related to improved security to facilitate dynamic negotiation of the employed cryptographic algorithms and to introduce Elliptic Curve Cryptography (ECC) extensions [190]. It should be noted that multicast mechanisms for HIP are not within the scope of this manuscript but described by others [208, 257].

2.5.1 Persistent Identifiers

HIP achieves persistent identifiers by introducing a new namespace for the transport and application layers that is decoupled from the network layer addresses. The identities are managed by a new logical layer be-

ISPs, home networking equipment manufacturers, and web companies

[30] `http://datatracker.ietf.org/wg/hip/charter/`

tween the transport and network layers[31] that manages the bindings between the identifiers and locators as illustrated in Figure 2.1.

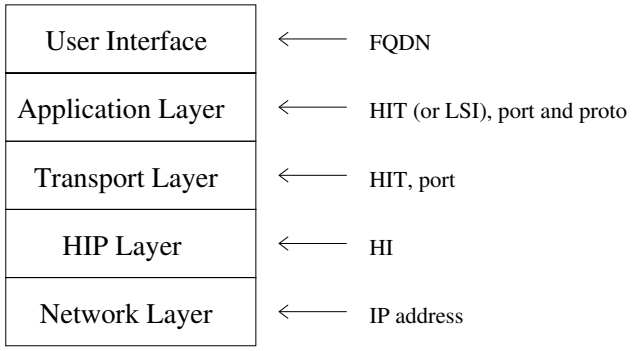

Figure 2.1. HIP decouples the identifiers of the upper layers from the locators used at the network layer

The new namespace is based on public-key cryptography. According to HIP terminology [164, p. 5], an abstract identity is referred to as a Host Identity, whereas a Host Identifier (HI) refers to the concrete representation format of the corresponding identity, that is, the public key of a host. The end-host is responsible for creating the public key and the corresponding private key for itself. This way, a HI is self-certifying and statistically unique.

A HIP-capable host creates two other compressed representations of the HI as public keys are of variable length and, thus, unsuitable to be used in fixed-length headers at the HIP control plane and incompatible with legacy IPv4 or IPv6 applications. The format for the control plane and IPv6 applications is the same: the host calculates a hash over the HI to fit it into an IPv6 address and sets a special 28-bit ORCHID prefix [172] for the generated IPv6 address called the Host Identity Tag (HIT). For IPv4 applications, the host locally assigns an IPv4 address, called a Local-Scope Identifier (LSI), that acts as an alias for the HI[32]. This way, HIP can support legacy IPv4 and IPv6 applications without any modifications to them.

A HI and also the corresponding locator can be stored in the DNS [171] or in any other suitable directory, such as DHT [10, 233]. However, a practical limitation with the hierarchical DNS is that flat identifiers as em-

[31] In RFC 5533 [175, p. 9] terms, HIP is located between the IP endpoint sublayer and the IP routing sub-layer

[32] The LSIs are assigned from the private address blocks, but implementers have also been experimenting with the unassigned 1.0.0.0/8 prefix. See also [186] for security advice on LSI implementation

ployed by HIP cannot be reverse looked up from the DNS unless some organization takes responsibility for the entire HIT prefix in the future [189]. As a reverse look-up is not guaranteed, a legacy application that has cached a HIT may not be able to connect to it, especially when the HIT belongs to a host located in a different domain. The caching issue can arise from the use of address literals in configuration files, or when the HIT is passed from one host to another in an application-layer protocol as a referral.

Applications assume stable addresses [220, p. 11] since the Sockets API does not expose the TTL values from the DNS to applications. In the absence of a deployed solution for this, the identifiers as introduced by HIP can be used to better meet this expectation, even in unmodified legacy applications.

In HIP terminology, the host that first contacts the other (by delivering a datagram) is called the *initiator* and the contacted host is called the *responder*. In other words, typically the initiator is the client-side host and the responder is the server-side host. The roles are used by the state machine of a HIP implementation during the setup of the control plane, which is called the *base exchange*. It is a key exchange procedure that authenticates the initiator and responder to each other using their public keys. The exchange consists of four messages during which the hosts also create symmetric keys to protect the control plane with Hash-based message authentication codes (HMACs) [229]. The keys can be also used to protect the data plane, and IPsec [116] is typically used as the data-plane protocol, albeit HIP can also accommodate others [49, 226]. Both the control and data plane are terminated using a closing procedure consisting of two messages.

The base exchange also includes a computational puzzle [24, 165] that the initiator must solve. The responder chooses the difficulty of the puzzle which allows the responder to delay new incoming initiators according to local policies, for instance, when the responder is under heavy load. The puzzle can offer some resiliency against DoS attacks because the design of the puzzle mechanism allows the responder to remain stateless until the very end of the base exchange. HIP puzzles have also been researched under steady-state DDoS attacks [33], and multiple adversary models with varying puzzle difficulties [225].

Figure 2.2 shows in more detail how HIP works in practice when IPsec is employed. In this example, the client (initiator) is equipped with a

HIP-capable DNS proxy and the server (responder) has published its HI records in the DNS. In step 1, the client-side legacy application first looks up the IPv4 (A) and/or IPv6 (AAAA) records of a server. The DNS request is processed by a locally installed DNS proxy that intercepts the request. In addition to the records requested by the application, the DNS proxy also requests HI records from the DNS in steps 2 and 3. To remain compatible with legacy servers, the proxy returns the records as they were returned from DNS when no HI records were found, but here we assume a HI record was found. The proxy caches the records with the HIP control plane module (in steps 4 and 5) and overwrites the response to the application. To be more precise, the DNS proxy translates the HI into an LSI or a HIT depending on whether the application requested A or AAAA records, and returns it to the application in step 6. Then the application delivers data to the identifiers in step 7, but this will be intercepted by the IPsec module that blocks the data and requests the HIP module to complete the base exchange in step 8. Upon completion after steps 9-12, the HIP module sets up the symmetric keys for IPsec as negotiated during the exchange in step 13, which unblocks the application data flow. In step 14, the client-side IPsec module encapsulates and protects the datagram. Finally, the server-side IPsec module receives the packet, verifies and decapsulates it, and delivers it to the server-side application in step 15.

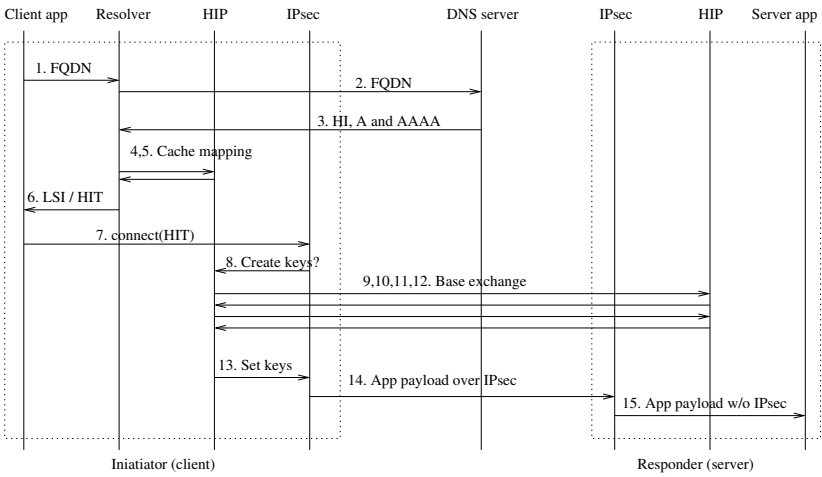

Figure 2.2. An illustration of how HIP works in practice

End-Host Mobility and Multihoming

As the identifiers in HIP are not routable, the HIP layer translates them into routable addresses, or locators as they are called in the HIP literature. The translation occurs within a new shim layer located between the transport and network layers. As the application and transport layers are bound to the persistent identifiers, the HIP can dynamically manage the mappings to the network-layer locators as shown in Figure 2.3. Thus, the HIP layer can manage both end-host mobility and multihoming [170] in a seamless way. Multihoming is typically employed for fault-tolerance purposes, albeit load balancing [187, 91] has been researched as well.

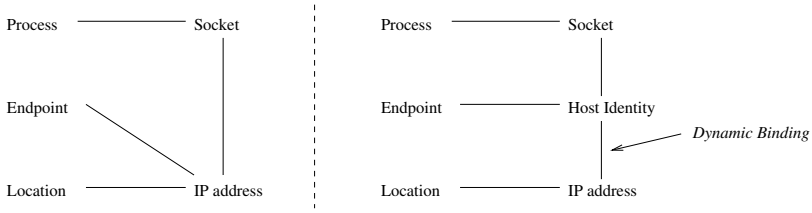

Figure 2.3. An illustration of the bindings in HIP [174]

At the application layer, legacy applications assume one address per interface [220, p. 12] despite multihomed devices being commonplace. For such applications, HIP can be utilized to mask the multiple local addresses behind a single, surrogate identifier[33].

At the lower layers, HIP-based handovers require three messages and are all protected using an HMAC and the public-key signature of the originating node. First, the mobile node announces the corresponding node of its new set of locators. Then, each of the corresponding node reply by sending a message with a nonce (i.e, a random number) to the mobile node. Finally, the mobile node completes the handover by echoing the nonce back to the corresponding node. This way, a corresponding nodes can avoid replay attacks by verifying that the mobile node has the address it claims to have. This verification procedure is commonly called the *return routability* test or check, which is also implicitly present already in the base exchange as well.

HIP based handovers have some limitations that are solved with extensions. Unless certain TCP extensions [65, 204] are used to optimize it, TCP is problematic with disconnectivity periods longer than a few minutes because its time-out mechanisms are independent of HIP. In

[33]Naturally, this assumption holds only if the underlying host offers only a single identifier for the applications

deployment scenarios where supporting *double jump* is mandatory, two communicating hosts can lose contact with each other and a *rendezvous server* [141] may be used as a contact point as it always has a fixed IP address. The rendezvous server relays only the first control message, and the communicating end-hosts communicate directly with each other after this.

Site Renumbering

During a site renumbering, the multihoming capabilities of HIP can facilitate a more seamless transition. HIP can be a suitable solution for site renumbering because it provides persistent identifiers for the hosts within a site. During business mergers, the identifiers forgotten in various software configurations will be valid even if the domain name of a site changes during the merger.

The site can also change its ISP, which also results in a site renumbering. With a publicly addressable site, the identifiers remain again persistent while the underlying locators change. A site employing a private address realm for its locators in addition to HIP can have the benefit of an aggregatable address space without limiting its connectivity. This is due to the fact that HIP provides NAT traversal capability for its persistent identifiers.

When a site changes its ISP, the HIs can have a long TTL in the DNS records. For publicly reachable locators, DNS caching issues can be avoided by two alternative means. Either the TTL should be short, or the locators of rendezvous servers can be employed as they can have a long TTL value in the DNS. To avoid readdressing of the rendezvous server itself during the change of the ISP, such servers should have locators external to the site.

Internet Transparency

As HIs and HITs are statistically unique, they can be used to distinguish and identify end-hosts in overlapping private address realms. Thus, HIP can be used to restore end-to-end connectivity in the Internet. However, IPv4-based NATs introduce two additional challenges. Firstly, they typically block all protocols other than TCP, UDP and ICMP. This issue is trivially resolved by encapsulating HIP and IPsec traffic in UDP. Secondly, NATs have introduced asymmetric reachability because they can block all new incoming data flows, including the HIP base exchange even though it would be UDP encapsulated. As rendezvous servers have public

addresses, they could be used to penetrate NATs but unfortunately this fails, especially when both of the communicating end-hosts are located in different private address realms. For such scenarios, either Teredo can be combined with HIP [234, p. 114], or the HIP-specific extensions [129] can be utilized. Besides NAT penetration procedures, the latter alternative also includes *relay server* extensions that guarantee the relaying of the control, data plane or both, in scenarios involving NATs resistant to peer-to-peer communications [213, 89]. The relay server is also a complete replacement for rendezvous servers in environments involving NATs.

2.5.2 Heterogeneous Addressing

The LSIs and HITs facilitate IPv6 interoperability at the application layer as one end-point of a communication can use an LSI and the other end-point a HIT [250, p. 3],[98, p. 15]. IPv6 interoperability is also supported at the network layer because the mobility and multihoming mechanism support cross-family handovers [117, 234]. Thus, it can be stated that HIP supports heterogeneous addressing, especially for end-hosts.

Due to the introduction of IPv6, access control lists double in network equipment such as in firewalls. This can be avoided in HIP-aware middleboxes, including firewalls and also rendezvous and relay servers, by enforcing homogenized identifier types. For instance, a HIT can be used to identify a host independently of whether the network-level connectivity is based on IPv4 or IPv6. Thus, the management of these middleboxes may be simplified when resorting to a single type of identifier such as HITs, again accommodating heterogeneous addressing.

2.5.3 Secure Addressing

The base exchange authenticates two hosts to each other with their public keys and implicitly tests for return routability. The public keys of the two hosts are present also in the application layer in a compressed format as HITs. This way, application layer security can be bound to lower layer security to support implicit channel bindings [99, p. 12]. In other words, the data flow of the application shares faith with the underlying public key based authentication: the data will either be delivered to the destination host possessing the private key or the data delivery will fail.

As HIP does not necessitate modifications to legacy applications, it improves their connection security in general as the data plane is typically

protected by IPsec. Some applications or services implement access control based on IP addresses. For this class of applications, the user or application can gain more confidence on the security level by populating the access control list with HITs (or LSIs) instead of routable IP addresses. However, other legacy applications may not be improved in this implicit way, and the user may not be aware when HIP-based security takes place [186, p. 353], thus requiring the application to be modified to become aware of HIP. This way, the application can involve the user in the decision to approve communications to verify the other party is out-of-band [186, p. 356].

A native API for HIP [128] makes the channel bindings more explicit to modified applications that can explicitly request the DNS resolver to return HITs and for the applications to manually configure HIT-to-IP mappings. The specification also defines an explicit way to use the HIP *opportunistic mode*, which facilitates LoF-based security for HIP where the initiator triggers the base exchange without prior knowledge of the identifier of the responder, and the initiator learns the identifier during the base exchange. As the use of the opportunistic mode implies a weaker security level, the API requires explicit consent from the application to use it.

The opportunistic mode does not meet the requirements for the persistent identifiers as the way the base exchange is triggered is solely based on the topologically-dependent address of the responder. Thus, this fails to achieve the goal of Internet transparency for persistent addressing and is problematic when the responder is mobile – a rendezvous server could be utilized to remedy the situation, but this would effectively render the opportunistic mode into a HIP-level anycast. Nevertheless, the mode can be useful, for instance, with publicly-reachable services with stable IP addresses when the extra interaction with DNS is to be avoided.

When HIP records are stored in DNS, the DNS responses can be forged by a man-in-the-middle attacker in the absence of DNSSEC. Thus, the DNS can be the weakest link in HIP-based security even when the opportunistic mode is not employed. As another weakness, advances in cryptanalysis may also lead to the discovery of problems with certain of the algorithms used by HIP. This means that HIs generated with the compromised algorithms will need to be regenerated and replaced with new HIs, meaning that even persistent identifiers as employed by HIP have a

limited life span[34], and may eventually have to be revoked [253].

Deployment Considerations

In the context of site renumbering, one benefit of protocols based on core-edge separation is that the number of nodes having to be upgraded is constrained as the protocol needs to be deployed only to edge routers. In contrast, core-edge elimination approaches require more upgrades as the number of end-hosts greatly surpasses the number of routers. Thus, researchers have proposed a number of proxy based deployment models for HIP [199, 109, 254, 100, 158, 173]. However, these various proposals will not be detailed here as the focus in this dissertation is on the standardized, elimination-based approach for HIP.

To operate HIP in the elimination approach, software has to deployed both at the client and server-side hosts [98]. The most essential component is the management software for the control plane[35]. Assuming the deployment scenario involves protection of application traffic, the data plane needs to be managed somehow – typically, HIP implementations employ an optimized mode of IPsec called the BEET mode [115], which is supported natively only by the Linux networking stack for the time being[36].

To support look up of HIP-based identifiers from DNS especially at the client side, either the DNS libraries should be modified to support HIP, or alternatively a local DNS proxy can be installed on the end-host as described earlier in the example in Section 2.5.1. Correspondingly, the identifiers should be stored in DNS at the service side. For the locators, a DNS server or rendezvous server can be used, with the trade-off between DNS caching issues and the cost deploying new infrastructure. However, a relay server is better than a rendezvous server in IPv4-based networks because the relay mechanism supports NAT traversal properly.

Routers, switches, NATs, existing firewalls, VPNs and the most popular DNS server software do not require changes to accommodate HIP when it is encapsulated in UDP [129]. However, penetration of aggressive firewalls can require non-standardized tricks such as employing IP-over-HTTPS [4] in HIP implementations.

[34] HIPv2 is more agile than HIPv1 in negotiating the used algorithms

[35] Regarding code complexity, the size of a HIP implementation can be half of the size of the corresponding MobileIP implementation with IKE-based security [249, p. 162]

[36] Separate support BSD exists, as well as a userspace IPsec implementation for Windows

HIP itself has been researched extensively and explored in the context of specific deployment scenarios, including in environments involving constrained devices [124, 168, 231], SIP [135, 48] and cellular [97] networks, and also in cloud networking [130]. Its principles have been reused in a number of other network research architectures [81, 78, 29, 94]. For commercial purposes, HIP has been adopted in a few known cases. At the Boeing airplane factory in Seattle, HIP secured connectivity for mobile robots [178]. HIP has also been used to implement a layer-two VPN [100] in a product called Tofino.

2.6 Summary and Comparison

The earlier sections in this chapter described individual protocols under certain categories despite some of the protocols fit multiple categories. For easier comparison, this section gives an overview of the challenges and solutions in consolidated naming.

In general, the results are presented in tables where a tick labels a protocol that fulfills the property unconditionally, whereas parentheses indicate when the property is fulfilled conditionally or when the property is optional. The qualities of the protocols are obtained from the literature references, but their arrangement as a taxonomy as described in this dissertation is novel.

To keep this summary short, we omit the protocols that do not support all the four properties for persistent identifiers: end-host mobility, multihoming, site renumbering and Internet transparency[37]. Here we exemplify some of the missing properties, albeit the list of missing properties is not complete. LISP, i3, Evolution and GSE are excluded because they not improve Internet transparency. Transparency is also missing from M-FARA, SCTP, MPTCP, TLS, dTLS and SSH in addition to the support for renumbering. DoA does not specify mobility and multihoming support as it focuses on interactions with the middleboxes. Similarly, PCP, uPnP, ICE[38] and Teredo are not tailored for mobility, multihoming, and site renumbering purposes. In addition, SHIM6, NBS, 4+4, and Plutarch do not completely survive site renumbering as they do not provide static identifiers for applications that cache or hard code the identifiers into

[37] After all, persistence of identifiers was also the most thoroughly analyzed of the three qualities

[38] At the time of this writing, mobility for ICE [244] has been proposed but not yet officially adopted in the IETF

their configurations. NAT64 is merely a single-purpose tool for IPv6 transition and DNSSEC a tool to secure DNS look ups. Finally, PLA is an efficient tool for combating DoS attacks and source address verification. However, in order to support all four properties of persistent identifiers, either the core concepts of PLA should be integrated into other architectures [143], or PLA should be combined with some other protocols similarly as in Back to My Mac (BTMM).

The exclusion leaves us with seven protocols with persistent identifiers, Mobile IP, ILNPv6, LIN6, BTMM, NUTSS, MobIKEv2 and HIP that will be compared in further detail in the remainder of this section. It should be noted that Mobile IP and MobIKEv2 are included in the comparison even though their support for persistent identifiers may depend on the particular deployment scenario. For instance, complete support for Internet transparency may require support for the protocol at the two communicating end-hosts, and also the use of publicly reachable surrogate addresses. We also assume here that the middleboxes for the two protocols are unaffected by network renumbering.

Table 2.1 shows how the seven protocols accommodate heterogeneous addressing. Legacy application support is divided into three capabilities. The first two consist of rudimentary support for IPv4 or IPv6 addresses. The third one refers to more advanced IPv4-IPv6 interoperability, that is, whether the protocol will support an IPv4 application in communicating with an IPv6 application and vice versa. Next, the protocol may be associated with an API for protocol aware or so-called "native" applications that offer relief for heterogeneous addressing simply by introducing a new homogeneous identifier. Finally, network-layer compatibility in IPv4 or IPv6 networks is displayed in the last two columns[39].

Networking	Legacy application support			Native	Network layer	
protocol	IPv4	IPv6	v4-v6 interop.	new id	IPv4	IPv6
Mobile IP	✓	✓			✓	✓
ILNPv6	(✓)	✓		✓	(✓)	✓
LIN6		✓				✓
BTMM		(✓)			✓	
NUTSS	✓	✓		✓	✓	✓
MobIKEv2	✓	✓			✓	✓
HIP	✓	✓	✓	(✓)	✓	✓

Table 2.1. Capabilities of facilitating heterogeneous addressing in the protocols

Table 2.2 summarizes benign mechanisms to support security. The first

[39]Compatibility of BTMM in IPv6-only networks was not documented [59]. Hence, we tried it in July 2012 and it failed to work for no apparent reason

column names the protocol in question. The following column refers to a protocol where the client authenticates itself to the server, and correspondingly the next column refers to the server authenticating to the client. Here, authentication refers to authentication based on pre-shared secrets (including passwords), public keys or certificates. The following column under the label confidentiality refers to data-plane encryption. The last column signifies that the protocol is designed to be resilient against DoS attacks. Again, as a baseline, it should be noted that unmodified TCP/IP does not have any of the properties.

Protocol	Client auth	Server auth	Confidentiality	DoS prot.
Mobile IP	✓	✓	(✓)	
ILNPv6	(✓)	(✓)	(✓)	
LIN6			✓	
BTMM	✓		✓	
NUTSS	(✓)	(✓)	(✓)	(✓)
MobIKEv2	✓	✓	✓	(✓)
HIP	✓	✓	✓	✓

Table 2.2. Secure-related properties of the seven protocols

Table 2.3 summarizes the different namespace properties of the seven protocols when they are used in the context of the existing IP-based Internet. After the protocol column, the next column describes whether the protocol is semantically based on a separate, disjoint namespace from the IPv4 or IPv6 address spaces or is overlapping from the standpoint of the application layer. The following column describes whether the address space is structured i.e, based on aggregatable or hierarchical identifiers. The last column signifies whether the identifiers are assigned centrally or in a distributed fashion. For instance, modified Extended Unique Identifier (EUI)-64 [103, p. 8] addresses are based on centrally assigned MAC addresses, and also URIs [36] require central assignment, whereas the ORCHID [172] and ULA [104] types of identifiers are self-assigned and, hence, statistically unique. As a base line, unmodified IPv4/IPv6 addresses are typically structured and centrally assigned, and they are also overlapping because a routable IP address couples the roles of an identifier and locator. In general, it should be noted that unstructured namespaces are subject to referral issues when the underlying name resolution infrastructure supports only structured look-ups.

Table 2.4 summarizes the design of the seven protocols in general. After the protocol column, the next column describes the deployment model: symmetric middlebox deployment at both ends (core-edge elimination),

Protocol	Disjoint	Structured	Assignment
Mobile IP		✓	No special assignment
ILNPv6	✓	✓	EUI-64
LIN6	✓	✓	EUI-64
BTMM	✓	✓	ULA
NUTSS	✓	✓	URI
MobIKEv2		✓	No special assignment
HIP	✓		ORCHID

Table 2.3. Technical characteristics of the identifiers in the different protocols

symmetric end-host deployment (core-edge separation) or asymmetric deployment (gateway at one end). The following column specifies whether the protocol stores state information on the identity-locator bindings in datagrams (tunneling) or as an extra state at hosts (translation). After this, the next column denotes a protocol that only requires changes at either the client or server side, but not both, typically implying an intermediate proxy or gateway. Finally, the last column indicates whether the protocol depends on new infrastructure, such as protocol-specific proxies or name look-up servers.

Protocol	Design type	Data plane	1-side	Infra
Mobile IP	Gateway	Tunneling	✓	✓
LIN6	Elimination	Translation		✓
ILNPv6	Elimination	Tunneling		(✓)
BTMM	Elimination	Tunneling		✓
NUTSS	Elimination	Translation		✓
MobIKEv2	Gateway	Tunneling	✓	✓
HIP	Elimination	Tunneling	(✓)	(✓)

Table 2.4. Summary of some of the technical design choices related to deployment

It is worth noting a few dependencies in Table 2.4. The last two columns depend on each other, i.e., a proxy-based deployment requires infrastructure, albeit the reverse is not necessarily true. Also, core-edge separation results in incomplete Internet transparency as the end-hosts cannot address routers directly[40].

The seven protocols have their own trade-offs when deployment is considered. For example, NUTSS is heavily reliant on infrastructure although able to support off-the-path services by introducing this dependency. Similarly, LIN6 introduces its own infrastructure for identity-locator mappings even though its generalized identifiers survive site renumbering events well. In contrast, ILNPv6 reuses DNS to store the mappings and also to deal with the double jump, albeit this requires DNS servers

[40]Thus, all protocols are based on elimination as non-transparent protocols were excluded from this summary

to be upgraded to support secure dynamic DNS updates. As a drawback, ILNPv6[41] splits an IPv6 address into identifier and locator formats in a way that exposes the possibly stale locator portion to the application, a problematic issue with legacy applications which cache addresses during site renumbering. Hence, legacy applications should be ported to use the FQDN-based API for ILNPv6. BTMM is a very complete protocol, albeit still a vendor-specific technology. It inherits a weakness from its use of uPnP as it does not work with multiple cascading NATs. Finally, HIP is similar to BTMM except that it is based on a single, unified protocol rather than a collection. In contrast to BTMM, it introduces secure identifiers (HITs) that can be used for authentication in access-control lists of applications and middleboxes. The trade-off here is that its flat, self-assigned HITs introduce referral issues that are problematic for reverse resolution, i.e, when mapping a HIT back to a FQDN or routable IP address. While the existing IPv4 is also tainted by referral issues due to issue of private address realms and missing records to achieve reverse resolution, HIP can remain architecturally clean if an organization takes on the responsibility of managing the entire IPv6 prefix assigned for HIP.

To conclude, this chapter has introduced a taxonomy for consolidated namespace and described a number of solutions that fit one or more of the categories. Based on the solutions that have been found for persistent addressing, the capabilities to facilitate heterogeneous addressing in Table 2.2 and the improved security characteristics in Table 2.2, HIP appears to be a decent match for a consolidated namespace. An initial mention has already been made of the contributions of the individual publications in this section, and the following chapter will explain these contributions in more detail.

[41] ILNPv4 employs IPv4 options and ICMPv4, which are blocked by many firewalls

3. A Consolidated Namespace for Network Applications, Developers, Administrators and Users

In this chapter, we summarize the contributions of the individual publications related to consolidated name spaces at a high level. First, we do a reality check to understand the real state of network applications and verify some of the challenges in Section 3.1. Next, we analyze how well HIP meets the challenges of a consolidated namespace in Section 3.2 – especially from the viewpoint of a developer – based on the contributions of the publications. Then, we view the impact of HIP on end-users in Section 3.3 and continue with a deployment perspective in Section 3.4, especially concerning network administrators. Finally, we summarize the contributions in Section 3.5 and suggest items for further work in Section 3.6.

3.1 Revisiting the Challenges for Network Applications

Publication I characterizes the network applications and frameworks in Ubuntu Linux. The goal of this investigation was to understand how open-source network applications utilized the low-level Sockets and POSIX APIs directly or indirectly, and how applications employ security. As the number of C-based software packages using the low-level APIs was relatively high (710), they were analyzed only statistically, and four example application frameworks were inspected manually. We investigated challenges related to IPv6, the use of UDP, TLS/SSL-based security, and the use of a number of extensions for the Sockets API. In this section, we highlight some key issues related to persistent, heterogeneous and secure addressing in the low-level networking APIs based on our findings. It is worth noting that revisiting all aspects of consolidated addressing is impossible within a single publication and, thus, we admit to have merely scratched the surface.

As IPv4 address space has been nearly exhausted, IPv6 has become more important. This means that network applications will have to be modified to support IPv6 addresses. In the investigation, the number of applications supporting both IPv4 and IPv6 was 26.9%. To support IPv6 addressing for DNS resolution and network I/O, both client and server-based applications have to be modified. Such use of heterogeneous addressing increases the complexity of all applications. IPv6-mapped IPv4 addresses do not really solve the issue of heterogeneity because they do not work with all sockets API functions and are actually considered harmful when they leak to the network[159, pp. 1-4].

To avoid complexity with the Sockets API, certain applications employ network application frameworks to hide the low-level networking details. Therefore, we also investigated the use of low-level networking in four frameworks: java.net for java-based applications, Twisted for Python-based software, and Boost and ACE for C++ applications. We discovered an issue with non-persistent use of addresses in all of the frameworks, which was a problem tainting many of the non-framework applications as well. To be more precise, the problem occurred only in the context of UDP with hosts equipped with multiple addresses. The UDP multihoming problem typically occurs when a server-based application receives a UDP-based message from a client and responds back without specifying the source address explicitly. Ignoring the source address may result in the underlying networking stack choosing a wrong source address at the server and the client dropping the message as it appears to originate from an entirely different server. As many of the commodity devices of today, ranging from hand-held devices to rack servers, are multihoming capable, the impact of the problem should not be ignored. It is worth mentioning that similar problems are also being addressed in the Multiple Interfaces (MIF) working group at the IETF but with a broader scope [39, 238].

While the UDP multihoming issue can be directly solved by fixing the issue in the application, this problem could also be worked around with multihoming extensions that introduce only a single identifier to the application[1]. Some of these extensions also support TCP and allow address changes throughout the lifetime of transport-layer sessions – a vanilla TCP, on the other hand, cannot survive an address change during com-

[1]This applies also to the weak end system model mentioned in the previous section

munications because it is tightly bound to the IP addresses. UDP is more tolerant of such failures due to its disconnected nature but would still require additional application-specific logic to facilitate decoupling of the data flows from the addresses in a secure way.

For instance, SCTP, MPTCP, SHIM6, HIP, MobileIP or MobIKE can be used to support persistent connectivity for multihoming. However, SCTP requires some changes in the application and has been adopted only by few applications in Ubuntu Linux. With the exception of MobIKE[2], the remaining three protocols have not yet been adopted in vanilla Ubuntu, and all four would have been difficult to trace due to their transparency at the application layer.

The most popular TLS/SSL implementation was OpenSSL, which was used in 10.9% of the applications. Here, we highlight two security-related aspects with its use. Firstly, the initialization procedures were neglected in many of the applications. For example, the Pseudo-Random Number Generator (PRNG) was seeded properly only in 58.4% of the C-based applications utilizing OpenSSL and in two out of four frameworks. Secondly, the setting of the security-related options was popular (53.3%). For instance, 20.1% of the applications explicitly allowed downgrade from TLSv1 to an older version of the protocol, SSLv3. While supporting such backward compatibility during transition periods is beneficial for the end-users, this raises the question of why the applications are exposed to such protocol details at all. In the ideal case, such complexity should be automatically and transparently handled inside the library implementing the security. This lesson applies also to other security protocols with explicit APIs, including HIP [127].

3.2 HIP as a Consolidated Namespace for Network Applications

TLS/SSL had been embraced by the developers, perhaps partly due to its visibility to the applications, and it was only natural to try to repeat its success with HIP. In Publication II, we designed and implemented a native API for HIP-aware applications. The API implementation required changes both in the system resolver and in the Linux kernel. We integrated the APIs with an example proof-of-concept application and extended the host-specific security model of HIP to allow user or application-

[2]Strongswan-ikev2 package had 3582 installations (with rank of 16804) in a Ubuntu popularity contest in January 2012

specific identities; similar trends appeared later in the Unmanaged Internet Architecture (UIA) [79, 78] architecture.

The native API for HIP meets the criteria for consolidated addressing quite well. For persistence, the API uses the location-independent identities of HIP. From the standpoint of security, HIs are secure by their nature. While the legacy APIs for HIP accommodate heterogeneous addressing with LSIs and HITs, the native API homogenizes the identifiers for HIP-aware applications using so called *end-point identifiers*. Such an identifier hides the different forms of HIs – LSIs, HITs and also application-specified HIs (i.e., public keys) from the application. The end-point identifier resembles a file or socket descriptor, and thus serves as an indirect, local reference to the corresponding HI. In the native API, applications manage mappings from the descriptors to HIs either directly or indirectly using the DNS resolver. The application interaction with the native APIs and DNS is illustrated in Figure 3.1.

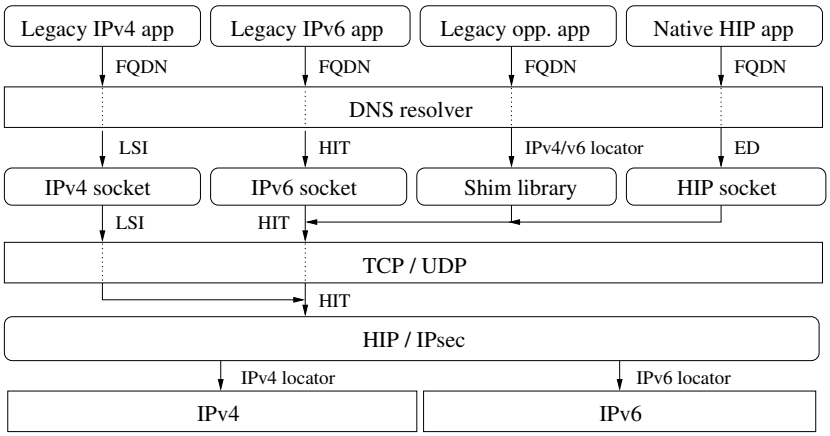

Figure 3.1. An illustration of how HIP was logically used in an end-system with legacy, opportunistic and native APIs. It should be noted that the resolver interaction is unnecessary after the socket is bound to the selected identifiers

While the descriptors certainly introduced some amount of complexity, they were a useful utility in organizing variable-sized HIs into Sockets API structures with a fixed maximum length. Unfortunately, the somewhat "unorthodox" concept of the end-point identifier did not make it to the final RFC [128] due to a lack of consensus in the IETF. Instead of the syntactically homogeneous format for the identifiers, the RFC version specifies two different formats for different purposes. The first format is used when specific HITs are discovered from the DNS. The second format is used to denote unspecified HITs using "wildcards". For instance, the

wildcards are used by server-side applications that accept incoming data flows from any clients.

The wildcards are applicable to client-side applications as well. A HIP-aware client-side application uses a wildcard when utilizing HIP in a LoF fashion. According to the RFC [128], the application specifies the HIT as a wildcard and sets the corresponding locator using a socket option the locator. This way, the application can request the user to confirm the HIT of the server. In fact, later research verifies that a separate monitoring API (or user confirmation in general) should be available when employing the LoF model with HIP [186, p. 353].

While applications ported to use the native HIP API can directly interact with the user to inform about the security being employed, this still leaves a vast number of legacy applications without such support. In Publication VI, we implemented a graphical firewall for end-hosts to manage and approve HIP-based data flows directly from the end-user. While the prototype was far from complete, the experiment was a success in the sense that HIP could be used with legacy applications while improving visibility of HIP-based security to the user.

In Publication VI, we also conducted usability tests with the graphical interface for HIP to understand how users perceive the use of HIP with varying levels of visibility in the context of the web. In general, the users perceived the interface reasonably well considering the maturity of the prototype. The experiment included usability tests with HITs that were visible to the application and also based on the transparent LoF implementation. In retrospect, it became apparent that the implemented graphical interface improved security, especially for the opportunistic mode implementation for legacy applications as described in Publication V, because a subsequent security analysis by Pham et al [186, p. 352] asserted that user involvement in LoF-based security is critical in HIP. Further, the authors state that the credentials (i.e., the binding from the FQDN to HI in this case) of the server should be stored permanently and its deletion should be left only for expert users. Our user interface met the former criteria but not the latter as the deletion did not require special privileges in the implementation. Figure 3.2 illustrates LoF-based security using screenshots of the graphical user interface and the browser[3]. In this case, the HIP plug-in for the browser does not dis-

[3]It should be noted that the prompt appears chronologically speaking before the browser receives the web page, but these two consecutive events are joined here for illustration purposes

play a lock or highlight the address bar because the implemented LoF library masks the use of HIP-based identifiers from the application layer. However, the user still receives a security notification from the graphical user interface.

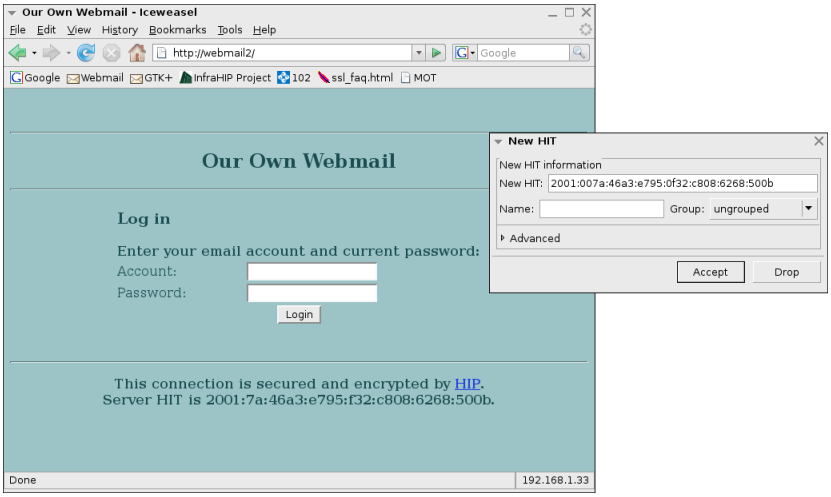

Figure 3.2. An illustration of LoF-based connectivity using a HIP-aware web service, a browser and the graphical user interface

To protect application-layer services with HIs in an address-family agnostic and secure way, Publication IV experimented with a middlebox-based firewall that can filter HIP control and the data plane. Basically, the firewall tracked persistent HIs instead of the ephemeral IP addresses of mobile end-hosts, thus supporting access control for services using HIs.

Compared to traditional firewalls, a limitation of the middlebox-based firewall for HIP is that it provides somewhat coarse grained access control. It cannot distinguish traffic between different services that are being hosted by a single server because the traffic can be encrypted. As discussed in Publication IV, this issue could be mitigated by employing the service-specific identifiers. That is, by assigning a unique identifier for each service, the firewall can control access to the services at a more fine-grained granularity even for encrypted data flows.

Similarly as in the native API, the HIP-based firewall can be made up of entirely homogeneous identifiers that hide the details of the underlying network topology and the IP version. Also, the HITs could be used to define more fine-grained policies at the network layer than with VPNs in order to protect against attacks originating from inside the company; it may therefore be easier to exclude individual hosts as the HIP firewall

policies are based on lists of individual end-hosts rather than network prefixes.

Publication III further continued the exploration of mitigation of unwanted traffic but focusing on server-to-server communication. The use case here is mitigation of unwanted application-layer traffic in the form of email spam which continues to thrive, especially considering that sending spam is cheap because spammers maximize their profits by maximizing spam rates [119, p. 4]. In our cross-layer solution, a HIP-aware spam filter installed on a public Simple Mail Transfer Protocol (SMTP) server authenticates and monitors email traffic based on HIP-based identifiers. Upon receiving spam from another server, the filter throttles the sender by terminating the corresponding HIP control and data plane. Thus, the sender of the spam has to establish a new base exchange with the filtering server, but this time experiencing larger puzzle. This way, the server can slow down the spam rate by introducing a computational cost to the sender. It should be noted that the use of computational puzzles has also been proposed by others to prevent outgoing spam from an economic perspective [85]. Our work differs from the other research on HIP puzzles [33, 225] because we focus on cross-layer mitigation of spam, and we also consider attackers that change their identity.

The main findings of spam mitigation with HIP-based identifiers and puzzles can be summarized with two key points. Firstly, ill-behaving hosts can try to circumvent the system merely by changing their identity. Upon being throttled by a puzzle, the host sending spam has strong incentives to change its identity if the cost of generating a new identity is smaller than solving the issued puzzle. Secondly, the puzzles also incur a cost to well-behaving hosts because such hosts may first have to earn a good reputation before receiving smaller puzzles. Also, it should be considered that sometimes spam filters can classify good email as spam. Taking the two findings into account, the main strategic decision for a server issuing the puzzles is to decide how much well-behaving hosts can be throttled with puzzles. In other words, increasing the time ill-behaving hosts spend in solving puzzles respectively results in an increase to the time well-behaving hosts use for puzzles.

3.3 Impact of HIP on End-users

The usability experiments in Publication VI investigated how end-users perceive security indicators consisting of visual and textual cues. The indicators were implemented for the HIP-based management and prompting user interface, a HIP aware web browser, and test website as illustrated earlier in Figure 3.2. The indicators (or their absence) in these software modules visualized communications based on plain IP, normal HIP, opportunistic HIP (as implemented in the leap of faith prototype in Publication V), SSL over IP, and SSL over normal HIP.

The usability tests revealed that the management interface for HIP connectivity was rather unpolished. Unsurprisingly, we observed that users did not even want to see long and lengthy HITs but preferred human-friendly names. For a relatively new protocol such as HIP, users preferred familiar security indicators for web browsers, including lock symbols and colored address bars.

It appeared to be irrelevant for the users to see the difference in how the security was implemented, whether it was normal HIP, TLS/SSL or both. They also did not perceive any noticeable difference between normal and opportunistic HIP, even though the latter effectively disabled the security indicators in the browser because the opportunistic mode implementation hid the presence of identifiers from the application as shown earlier in Figure 3.1. However, this phenomena could be explained by the presence of the user interface that prompted for the new HIP based connections.

We witnessed the recurring fact that security can easily go unnoticed by the users. This applied to the absence and presence of security indicators; users did not report the absence of indicators when connecting to an unsafe site, nor did they report the presence of the indicators when they were not prompted as the HIT of the server was already cached by the user interface for HIP. Nevertheless, the users ranked security levels correctly; connections intended to be secure were ranked clearly more secure than insecure ones.

Besides describing the native API for HIP, Publication II introduced user-specific identifiers. The idea was that users could transfer the identity from one device to another, and the HIP software module would then import the identity. Thus, a HI became a portable user identity.

It should be noted that the approach had its limitations; on-the-fly session migration was not supported because it would have required trans-

ferring the transport-layer state as well[4]. As another limitation, the user-specific identifiers could be imported from any media, including USB memories or disks, but importing of user identity to a host involves at least two security risks as mentioned in the publication. On a multi-user machine, the same identity could be used by other users, depending on the security granularity of HIP implementation. On a compromised host, the private key part of the identity could be replicated by the intruder. While the multi-user issue could be solved with fine-grained access control mechanisms, the other issue is more difficult to mitigate, especially when the intruder has administrative privileges. The escalation of the situation could be contained by employing a smart card that would securely store the private key and sign data upon request. At least this would prevent compromising of the private key.

3.4 Deployment Aspects

Any new protocol, whether it be research or industry originated, is subject to the scrutiny of a realistic deployment scenario. Based on the statistics on the Sockets API in Publication I, we discuss the deployment of HIP and other related protocols from the viewpoint of their APIs. While the findings lack longitudinal analysis, the adoption of certain API trends in the latest Ubuntu Long-Term Support (Ubuntu) (LTS) were apparent.

The deployment of IPv6 at the application space was relatively small as only a quarter of the applications supported it. Partially, this could be explained by the fact that the analysis also included less popular applications with perhaps inactive code maintenance. However, it can be also speculated that the lack of IPv6 support may originate from application developers that endorse homogeneous APIs by clinging onto IPv4. As explained earlier in Section 3.1, management of IPv6 introduces extra complexity in managing the network connections of applications.

OpenSSL was used by 10.9% percent of the software even through it requires pervasive changes in the networking logic of the application. Other protocols requiring more modest changes were not so popular; SCTP and Datagram Congestion Control Protocol (DCCP) were used only by a few applications. While the kernel and Sockets API changes for these two protocols are present in modern Linux systems, it appears that library-based

[4]It should be noted that session delegation with HIP has been further analyzed by others [102]

solutions such as OpenSSL are more welcomed by application developers. As the native API for HIP is also based on the Sockets API, it could be argued that the native API will not be adopted rapidly either.

Roughly two out of six applications were setting socket options. Based on their popularity, it could argued that API extensions based only on socket options might have better chances of adoption due to their familiarity with developers. Examples of protocols employing such extensions are MPTCP [202, pp. 8-12], shared multihoming extensions for SHIM6, and HIP [127]. In contrast, native API for HIP [128] may experience slower adoption because the extensions are tightly bound to the new and still unpopular DNS resolver.

According to the coarse-grained estimates summarized in Section 3.1, it appeared that roughly two thirds of the applications were selecting IPv4 source addresses explicitly. Assuming IPv6 will be more widely deployed, one could assume a similar ratio for the adoption of IPv6 source address selection. Proper source address selection is important also in the context of HIP-based firewalls; ignoring the selection may result in the networking stack choosing a source HIT that will be filtered by the firewall in the middle. If not the firewall, the server-side application may be the source of a failed connectivity; the reported UDP multihoming problem applies equally to HIP-based connectivity if the server is configured to have multiple HITs.

RFC5887 [52, p. 34] lists a number of potential sources of IPv4 address literals and refers to another study [211] that lists potential address dependencies in 34 out of 257 RFC protocol specifications. Based on an empirical study of an IPv6-only testbed, Arkko et al [17, p. 14] reported that some network software used IPv4 address literals directly instead of relying on DNS-based resolution. In contrast, we did not observe any such addresses in the statistical analysis of the Sockets API in Publication II. However, this may have occurred because our investigation was limited to a static source code analysis (instead of a run-time one or traffic analysis) and we also excluded configuration files.

As a specific use case study of referrals, we experimented empirically with FTP in the context of the native API for HIP in Publication II. This particular case did not appear problematic in practice because FTP typically passes addresses as “callbacks” between a client and a server, meaning that addresses are mirrored between a pair of hosts. This is not a problem unless the FTP server redirects the session to another server

based on the IP address. In such a case, a HIP-aware application and application-layer protocol would be required to also pass the IP address to the other host.

While RFC 5218 [221] does not consider adoption issues specifically in the context of APIs, the native API for HIP was designed to maximize familiarity for developers; the API included a simple resolver option to make it return HITs exclusively. While this provided a minimal way to enforce the use of HIP-based security by a client-side application, we also defined some additional functionality for more advanced use. It is also worth mentioning that the extensions for user-specific identifiers in the API were also designed to be compatible with the standardized HIP to make their adoption more seamless. Yet, the native API may have a rocky adoption road in front of it because even its minimal use depends on the unpopular, modern DNS resolver.

In general, it would be easier to deploy HIP along with a new system such as P2P-SIP [113, p. 77],[48] that is already HIP aware [50]. However, Publication III proposed a use case for the existing email services, but limited the deployment of HIP to the server side. More specifically, HIP was deployed only on SMTP servers because the number of client-side hosts clearly outnumber the SMTP servers. Instead of end-to-end use, HIP was used here in a point-to-point fashion due to the nature of the SMTP service. In order for early adopters to obtain the benefits, the publication advised introducing extra delays for HIP-incapable servers as an adoption incentive. This could facilitate incremental deployment as the deployment of HIP would not require a flag day.

The DNS-related changes for HIP are backwards compatible as reported in Publication III. HIP has new records in the DNS that do not interfere with HIP-incapable hosts, and the format can be used without any changes in the most popular DNS server software, bind [98, p. 19]. Also, the look-up of DNS records for HIP can be transparently handled at the SMTP servers using a locally installed DNS proxy.

Similarly as in the SMTP use case, the middlebox-based firewall for HIP in Publication IV was aimed at specific groups of people in order for early adopters to obtain the benefits. To be more specific, the HIP-based firewall can be used internally, e.g., within a single company in a similar fashion to a VPN. The benefits of the firewall can be useful for the entire company but especially for the network administrators. When a company changes its ISP or merges with another, the HIP firewall can support

network renumbering because it tracks persistent identifiers rather than ephemeral addresses[5]. The hard-coded HIP-based identifiers residing in various application and service configuration files are immune to changes in the site topology and, thus, avoid unnecessary down time. Management of the firewall can be more simple for the administrators as separate rules for IPv4 and IPv6 are not needed, thus perhaps reducing the number of configuration errors.

The LoF prototype in Publication V offered the most conservative way of using HIP, which did not expose LSIs and HITs to the application layer at all. Instead, the applications used regular routable IP addresses. Consequently, no support from the DNS infrastructure was needed in order to deploy HIP, and also referral issues were not a major concern. While routable IPv4 and IPv6 addresses are not as secure as HITs in access control lists, it should be considered that the model was targeted for legacy applications unaware of HIP. As SSH was successful with its LoF model, we experimented to determine if HIP would be flexible enough to also support this using its opportunistic mode.

The LoF for HIP was implemented using a shim library that intercepted Sockets API calls between the application and the networking stack. The library was based on the opportunistic key exchange of HIP, and could be enabled at system, user or application granularity depending on the local policies. The library translated the IP addresses of the application to HITs, which were further processed by the IPsec module and translated back to routable IP addresses as illustrated earlier in Figure 3.1. Despite the additional translation between IP addresses to HITs, the library added negligible overhead to the throughput. As a configurable feature of the library, it also implemented a fallback mode[6] that bypassed the HIP-based processing after a timeout when connecting to a HIP-incapable host.

With the implementation model used for LoF, a legacy application witnesses only the initial locators, while the identifiers remain invisible to the application as they are masked by the shim library. This is effectively a design compromise because the LoF effectively gains ease of deployment at the expense of employing non-persistent identifiers. To be more precise, the persistence is lost only for the initial contact, since the opportunistic

[5]To support external access from other sites without HIP support, e.g., a web proxy is needed [130]

[6]The fall back as a generic mechanism was later also endorsed by others [13, p. 2]

mode operates without prior knowledge of the HI of the responder. The lack of persistent identifiers further compromises Internet transparency for NATted environments.

From the standpoint of security, the DNS resolution is the weakest link in the use of HIP until DNSSEC is widely deployed [59, p. 11]. Thus, during the transition to DNSSEC, the LoF model as presented in Publication V is a viable option because even the non-opportunistic HIP is susceptible to man-in-the-middle attacks without DNSSEC. Further, while introducing DNS records is fairly trivial and does not cause any conflicts with HIP-incapable end-hosts, the LoF model avoids HIP-specific records altogether. However, the presence or absence of DNS records describes clearly when a remote host supports HIP. Otherwise, a host blindly connecting to every host with the opportunistic mode has to resort to waiting and can only detect remote support for HIP after a possibly long timeout, which can be frustrating to the user. In fact, the fall back on non-HIP communications in the implemented LoF approach was based on such timeouts. As briefly discussed in the publication, we suggested a solution for the timeout issue in order to reduce the fallback delay. As IP options are typically filtered by firewalls, our TCP-specific solution for this was to exploit TCP options to detect HIP-capable hosts [38, pp. 24-26].

3.5 Summary and Lessons Learned

This section summarizes the contributions of the publications in supporting a consolidated namespace with HIP. The resulting architecture is viewed from the viewpoint of end-users, network application developers and network administrators as illustrated in Figure 3.3.

Publication I revisited the challenges of consolidated naming: non-persistent, heterogeneous, insecure addressing. The statistical analysis was limited to certain aspects of application-layer solutions as network-layer solutions are difficult to trace due to their transparency[7].

As an example of non-persistent addressing, we discovered a programming bug in the way developers implement UDP-based communications. The problem typically occurs on multihoming hosts and affects many of the UDP-based applications in Ubuntu Linux, including four network application frameworks where the problem was verified manually[8].

[7] Chapter 2 revisited the challenges and solutions through literature references
[8] It should be noted that HIP was not integrated into the frameworks

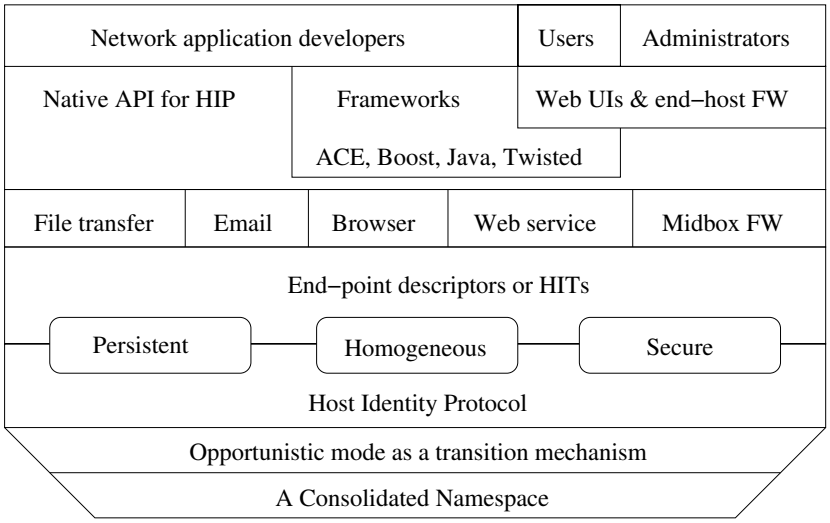

Figure 3.3. A visualization of the HIP-based solution for the challenges of a consolidated namespace

Regarding heterogeneous addressing, we observed that a fourth of the applications in Ubuntu support both IPv4 and IPv6. This dual use of the addresses complicates the networking logic of applications and can also have an impact on the user experience as the related latency issues have to be solved redundantly by each developer in the absence of a de-facto solution.

As IP addresses offer little security per se, the developers typically use SSL/TLS. In Ubuntu, roughly every tenth application utilized it through the OpenSSL library. However, almost three out of five applications using the library were not initializing it in a secure way. We also observed that a large number of the applications had explicitly configured a variety of security details for the library, leaving us puzzled over whether some of the details should not be exposed to the applications.

We proposed HIP as a unified solution for consolidated addressing and improved it to better fulfill this proposition. To support persistent identifiers for services, Publication III studied access control for HIP-based identifiers to mitigate against email spam with the aim of motivating end-hosts for long-lived identifiers. Publication IV presented a middlebox-based firewall to control access to HIP-based services for mobile HIP clients. To support heterogeneous identifiers, the firewall could again be used to homogenize IPv4 and IPv6 access control lists under a single list based on uniform HIs. The homogenization was also employed in the native API for HIP in Publication II that unified addressing for end-host applications us-

ing end-point identifiers. For secure addressing, the native API and also the graphical end-host firewall in Publication VI improved LoF-based security for HIP as later confirmed by others. It should be mentioned that the security of the identifiers played an important part in experimentation with the end host and middlebox-based firewall, and email spam.

In Publication III, we proposed a deployment model for HIP where it was used internally between SMTP servers. In the same publication, we also corrected a deployment-related misunderstanding according to which the HIP records would require change to the DNS server implementations. In Publication V, we showed how HIP can actually be used in the opportunistic mode without any additional DNS records. From the standpoint of security, the opportunistic mode is based on the weaker LoF-based security. Nevertheless, LoF can be considered secure enough until such a time as DNSSEC fortifies the weakest link in HIP, that is, the look-up of the HIP records from the DNS. However, we identified two shortcomings of the opportunistic mode for HIP. Firstly, the time-out mechanism to fall back on non-HIP communications when encountering HIP-incapable hosts can be optimized. Secondly, the opportunistic mode regresses to non-persistent addressing in order to facilitate easier deployment.

To recap the contributions from the viewpoint of the target groups, we tested usability of HIP in Publication VI, where the end-users used a web browser to connect to a website. The presence of HIP was visualized using various cues, including the lock symbol and also using a graphical prompt that operated at the system level. Despite the prototype being rather unpolished, the users understood when security was employed. The experiment also confirmed that HIP-based security should be visualized using traditional security indicators.

For developers, the native API for HIP, as presented in Publication II, allows gaining more control of HIP and supports application or user-specific identifiers. For network administrators, the middlebox-based firewall of Publication IV can simplify management issues as the persistent identifiers of HIP support mobile clients and survive network renumbering while unifying separate access control rules for IPv4 and IPv6 into single ones.

3.6 Future Directions

Publication I uncovered statistically a number of issues in networking software for Linux. In general, it would be useful to automatize them using a static analysis tool such as Coverity [37].

Publication II proposed native API for HIP. Parallel to this work, IPsec-policy APIs for applications [248] were developed, and the two approaches can perhaps be integrated. As another future work item in the native API itself are the user-specific identifiers that were excluded in the final standardized version [128] because the security issues were not thoroughly analyzed. Namely, handling of user-specific identifiers requires strict access control measures on multi-user devices. Related to the security of the identities, storing the user identity on a smart card or a TPM chip was proposed but not implemented. In practice, this would require delegation of privileges to the device from the card or chip and some kind of a revocation mechanism. As a straw-man proposal, a cryptographically accelerated smart card could be used to sign the data plane with public keys [49] instead of IPsec, but only while the card is connected to the device. As an alternative, an on-board TPM chip might be requested to sign a HIP-specific certificate [96] for a certain predefined time period. This way, the potential abuse of a portable long-term identifier would be limited spatially or temporally when using the identity in a compromised device, for instance, in a public Internet kiosk.

Publication III tackled spam by disconnecting TCP connections with misbehaving SMTP hosts and by introducing large computational puzzle values when they reconnect. In contrast, greylisting does not require the deployment of HIP and, for instance, can mitigate spam by disconnecting connections from previously unknown IP addresses. However, the assumption in greylisting is that spammers do not reconnect, which may not be necessarily valid in the future as spammers get smarter. Thus, an additional mechanism as proposed by the publication may become necessary in the future.

While Publication III acted as a starting point for HIP deployment considerations in the context of SMTP, we later also subsequently experimented with a HIP deployment internally in a cloud network [130]. In another publication [218], we also analyzed the deployment barriers of HIP from a techno-economical perspective based on interviews with IETF experts. The two most significant barriers were the low demand for the func-

tionality (in general) and the substitute technologies that have been earlier in the market. A potential direction forward is to focus on application-specific scenarios. As another approach, it could be easier to deploy HIP as a userspace library above the transport layer [86]. As a benefit, HIP-aware applications could then be developed independently of the deployment hurdles associated with the native HIP API.

Publication V used the opportunistic base exchange in HIP to achieve LoF for legacy applications. However, we were dissatisfied with the shim placement just above the sockets layer in contrast to others on this point [89, p. 10], and explored using another implementation model where the shim layer captures datagrams at the network layer instead of socket calls [75, pp. 34-36]. Independently of the implementation model, a user confirmation or a monitoring API is required [186, p. 353].

A relatively unexplored aspect of the opportunistic mode is that it could be used for implementing HIP-specific anycast with the help of rendezvous servers (or relay servers). This can be described with the example of an initiator that triggers an opportunistic base exchange through a rendezvous server to the responder. In this case, the rendezvous server can choose the responder from the list of registered hosts, thus facilitating HIP-specific anycast [9]. Alternatively, the same functionality could be achieved through the use of DNS-based load balancing schemes where the DNS server changes the order of the returned IP addresses for each look-up, the difference between the two being that the described HIP-specific anycast is independent of DNS. Again, it should be emphasized here that user confirmation or a monitoring API such as introduced by the native API is critical from the standpoint of security.

As outlined in Publication IV, the middlebox-based firewall for HIP could be deployed in WLAN access points to support passwordless authentication for end-users. However, the site administrators would need still to approve the users, and users need to possibly to configure a certificate for the site. As the associated configuration complexity hinder the usability of the system, the publication suggests adopting of the EasyVPN [35] approach, which bootstraps IPsec-based VPNs using TLS and web services.

In general, the number of distributed experiments with HIP has been small. For instance, the middlebox-based firewall and its experimentation in Publication IV could be extended. For production purposes, the

[9] Currently, opportunistic base exchange is always terminated by the rendezvous server [141]. A flag or a special HIT encoding should be added to denote anycast

prototype of the middlebox-based firewall could be extended to support de-centralized access control to support fault tolerance and asymmetric routes. For routing, also the impact of off-the-path routing based on the optional HIP relays could also be measured. As HITs are not aggregatable, bloom filters could be used to reduce the overhead of large access control lists [45]. In addition, a performance comparison with another similar scheme such as MobIKE would be useful.

Publication VI prototyped a graphical user interface for HIP. The design was rather immature as the results of the usability tests indicated. Nevertheless, we believe this end-user firewall for HIP could be integrated seamlessly with existing end-user firewall products, such as those offered by F-Secure, Symantec and other anti-virus companies. However, this remains future work.

4. Conclusions

While users and services are identified using DNS-based names, such names are an extension of the TCP/IP architecture, where IP addresses are omnipresent. IP addresses are used either directly or indirectly to develop network applications, and they are employed at the transport and network layers. While the design choice resulted in a relatively simple addressing architecture, the choice can be considered as a source of inflexibility because the computers of today are able to run sophisticated and complex software. Consequently, it has become economically challenging to overhaul this addressing model since the IP addresses are metaphorically speaking the glue that holds the Internet together. In this dissertation, we have attempted to improve three sources of inflexibility in the TCP/IP architecture in a backward-compatible way: non-persistent, heterogeneous and insecure addressing. A solution that meets the different aspects of these three challenges is referred to as a consolidated namespace in the context of this work.

Non-persistent addressing stems from the nature of IP addresses that are dependent on the local topology. This causes problems for mobile and multihoming devices when changing their network attachment point. The change of the local IP address unnecessarily terminates TCP streams as they are still bound to the previous address, which is not only a problem for on-going flows of data but also for new data flows. For instance, overlapping private address realms as introduced by NAT technology are tainted by non-persistent addressing because the devices cannot be uniquely identified based on their IP address alone. Thus, a mobile device may simply contact a wrong host when transitioning between two networks. As another example, the issue may manifest itself during a company merger or acquisition, or when the site changes its provider. In this case, the entire network prefix of the site may change, leaving a number of stale

IP addresses forgotten in various configuration files, only to be discovered by network administrators during site renumbering.

Heterogeneous addressing as introduced by IPv6 involves additional complexity for application developers and network administrators since they have to deal with two heterogeneous namespaces instead of a single one. Insecure addressing is the basis for the TCP/IP model; IP addresses are insecure by their nature and additional measures are needed to secure network applications.

A number of backward-compatible solutions that meet the various architectural requirements for a consolidated namespace do exist but fulfill the requirements only partially. From the surveyed evolutionary solutions, seven prominent solutions meet the described properties for persistent addressing and are compared in more detail for their other technical aspects. Mobile IP and MobIKE are successfully commercialized protocols, but do not facilitate IPv4-IPv6 interoperability at the application layer. NUTSS architecture is distinguished from the others because it facilitates off-path services albeit requiring additional infrastructure to complete this task. While LIN6 solves renumbering in a relatively simple way by dividing an IPv6 address into identifier and locator portions, it also requires extra infrastructure. In contrast, ILNPv6 is not ideal for renumbering purposes but mostly avoids the deployment costs incurred from new infrastructure. BTMM is a comprehensive solution for a consolidated namespace but unsuitable with multiple cascading NATs, and is essentially a vendor-specific, proprietary technology. The last protocol, HIP is chosen for the empirical experimentation as it fairs well in the comparison. Similarly to BTMM, HIP offers a complete solution but is based on open standards and open-source implementations. Another difference is that HIP is based on unstructured identifiers which have some ramifications for the structured DNS. However, the impact of this so-called referral issue still remains a moot issue. As a trade-off, the merits are clear as HIP offers a topologically-independent namespace based on cryptographically secured, self-certifying identifiers that are compatible with legacy applications.

In the collection of articles, we have empirically experimented with HIP and improved it to better meet the criteria of a consolidated namespace as set out by this dissertation. The results are examined from the viewpoint of three different target groups of people: end-users, network application developers, and network administrators.

First, we focused on the development aspects by trying to understand the general state of networking applications. To be more precise, we analyzed how network applications and frameworks use the low-level networking APIs in Linux. Related to heterogeneous addressing, we observed that a fourth of the IPv4-based applications also supported IPv6. Related to persistent addressing, all of the four manually examined frameworks had a bug related to UDP that occurred during initial connectivity with multihoming end-hosts. We also observed that every tenth application employed TLS/SSL-based security using the OpenSSL library. Of these, almost three out of five were not initializing the library correctly from the standpoint of security. Consequently, our analysis of network applications corroborates the challenges for consolidated addressing, at least to some extent. To make the investigation more concrete, we experimented with one of the best candidates for consolidated naming, HIP, and explored how some of its shortcomings can be improved.

Traditionally, HIP has been transparent to applications, whereas TLS/SSL offers explicit APIs for applications in order to secure their communications. The visibility of TLS/SSL to applications may have contributed to its success and in an attempt to repeat this with HIP, we designed and implemented a native API for HIP that gives more control for the developers of HIP-aware applications. The API also extends HIP to support application or user-specific identifiers instead of merely host specific ones. To better meet the requirements for a consolidated namespace, the API also unifies the heterogeneous two HIP identifier types used in legacy IPv4 and IPv6 applications into a single one. As later discovered by others, the separate API, such as the one we developed, is required to improve security when so-called leap-of-faith security is employed [186, p. 353].

As an example case study of HIP-aware applications, we exploited computational puzzles, a feature of HIP, to mitigate against unwanted traffic in the case of email spam. A modified spam filter throttled senders of spam by disconnecting sessions and offered more time-consuming puzzles for them. The access control was based on the persistent identifiers of HIP that could be circumvented with temporary identifiers. To avoid penalties with puzzles, the proposed strategy was to reward benign hosts with long-lived identifiers with less time-consuming puzzles. We proposed to adopt the mechanism only between email relays to avoid deployment hurdles at the client side.

As another case study, we developed a HIP-aware firewall that exploited

the secure identifiers of HIP to control access to HIP-based services. The novelty of the approach was that the firewall supported mobile client devices by tracking and authenticating them based on their persistent identifiers instead of their ephemeral IP addresses. Changes in the addresses of the servers were also supported as the firewall may also support site renumbering because identifiers forgotten in various configuration files can continue to work. As another relief for network administrators, a single identifier-based rule for access control replaced the two separate rules traditionally required for IPv4 and IPv6, thus supporting the goal of heterogeneous addressing.

In an intermediate step to making the transition towards HIP easier, we explored the use of HIP as a transparent mechanism that controversially does not meet the goals for consolidated naming at the application layer. Our implementation of the leap-of-faith security, known otherwise as the opportunistic mode for HIP, acts as a shim library between the application and the sockets layer that transparently translates IP addresses of the legacy application to HIP-based identifiers for the transport layer. As a trade-off, the opportunistic mode is subject to man-in-the-middle attacks because the client learns the identity of the server during communications instead of obtaining it from pre-shared information or look-up from a directory. Until DNSSEC is deployed more widely, we believe that operating HIP in this fashion offers reasonable security because DNS can be considered the weakest link for storing the identifiers for HIP. In addition, the implemented approach avoids the referral issue at the application layer.

We conducted usability tests by using a group test of persons to understand how end-users perceive different ways of using HIP in a web-based environment. The use of HIP was illustrated using traditional security indicators, such as the lock symbol in the browser. The transparent use of HIP was illustrated using a new graphical system level prompt that essentially acted as an end-host firewall to allow the user to accept or deny HIP-based communications. The users clearly noticed when security was employed despite the difference between the LoF and normal HIP-based security not being obvious. While we observed room for improvement in the prototype, others have identified confirmation of the opportunistic mode from the user as critical from the standpoint of security [186, p. 352].

The research contributions of this dissertation also have real-world impact. The general investigation into the low-level networking APIs and

network application frameworks revealed a number of bugs that can be fixed to improve the software quality in various Linux distributions. Some of the problems, especially the ones associated with the Java-based framework, may also affect the Linux-based Android operating system that has dominated the mobile handset business lately. Besides the general contributions, the HIP-specific contributions have had an impact on the IETF standardization. The experimentation with the UDP-based multihoming, opportunistic mode, end-host and middlebox firewalls are explicitly cited by two experimental standards [99, 98]. We improved the specification for the native API according to the feedback from the IETF community, and it was published as an experimental standard [128]. The work further inspired joint activities with the SHIM6 working group and produced another API-related standard [127].

For future directions, we have already taken a few steps that we hope others will pursue further. For instance, we sketched but did not implement HIP-based anycast which should be combined with the security of native API for HIP. To understand the deployment dimension better, we applied HIP in a cloud deployment scenario in another recent publication [130] but more work is needed to automatize the use of HIP and integrate it properly with the cloud management infrastructure. Then, we also conducted a techno-economic survey [218] and, based on the findings, we explored HIP as a stand-alone library at the application layer [86] to avoid some of the deployment hurdles.

Due the efforts of various researchers, HIP has become an extensively researched topic and its principles have also been adapted to other research architectures [81, 78, 29, 94]. Besides the Tofino security product and production-quality deployment at Boeing, the time is ripe for the industry to adopt HIP as it moves from the experimental to the standards track in the IETF. In contrast to the ambitious research goals in the beginning, the development and deployment should now be focused on application-specific scenarios, and optimized to the underlying environment to maximize the user experience.

Bibliography

[1] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 6: Medium Access Control (MAC) Security Enhancements, July 2004.

[2] OpenID Authentication 2.0 - Final, Dec. 2007.

[3] Port-based Network Access Control, IEEE Std 802.1X-2010, Feb. 2010.

[4] IP over HTTPS (IP-HTTPS) Tunneling Protocol, Oct. 2012.

[5] J. Abley, B. Black, and V. Gill. Goals for IPv6 Site-Multihoming Architectures. RFC 3582 (Informational), Aug. 2003.

[6] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill. IPv4 Multihoming Practices and Limitations. RFC 4116 (Informational), July 2005.

[7] B. Aboba, D. Simon, and P. Eronen. Extensible Authentication Protocol (EAP) Key Management Framework. RFC 5247 (Proposed Standard), Aug. 2008.

[8] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '04, pages 20–24, New York, NY, USA, 2004. ACM.

[9] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proceedings of the seventeenth ACM symposium on Operating systems principles*, SOSP '99, pages 186–201, New York, NY, USA, 1999. ACM.

[10] J. Ahrenholz. Host Identity Protocol Distributed Hash Table Interface. RFC 6537 (Experimental), Feb. 2012.

[11] S. Akhshabi and C. Dovrolis. The Evolution of Layered Protocol Stacks Leads to an Hourglass-shaped Architecture. *SIGCOMM Comput. Commun. Rev.*, 41(4):206–217, Aug. 2011.

[12] H. T. Alvestrand. Overview: Real Time Protocols for Brower-based Applications, Mar. 2012. Internet draft, work in progress.

[13] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: An Architecture for an Evolvable and Trustworthy Internet. In *Proceedings of the*

10th ACM Workshop on Hot Topics in Networks, HotNets-X, pages 2:1–2:6, New York, NY, USA, 2011. ACM.

[14] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). *SIGCOMM Comput. Commun. Rev.*, 38(4):339–350, Aug. 2008.

[15] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), Mar. 2005. Updated by RFC 6014.

[16] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971 (Proposed Standard), Mar. 2005. Updated by RFCs 6494, 6495.

[17] J. Arkko and A. Keranen. Experiences from an IPv6-Only Network. RFC 6586 (Informational), Apr. 2012.

[18] J. Arkko, V. Lehtovirta, and P. Eronen. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). RFC 5448 (Informational), May 2009.

[19] J. Arkko and P. Nikander. *Weak Authentication: How to Authenticate Unknown Principals without Trusted Parties*, pages 5–19. Springer, 2002.

[20] R. Atkinson and S. Bhatti. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740 (Experimental), Nov. 2012.

[21] R. J. Atkinson, S. Bhatti, and S. Hailes. ILNP: Mobility, Multi-homing, Localised Addressing and Security through Naming. *Telecommunication Systems*, 42(3-4):273–291, 2009.

[22] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), Jan. 2007.

[23] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard), Mar. 2005. Updated by RFCs 4581, 4982.

[24] T. Aura, P. Nikander, and J. Leiwo. DOS-resistant Authentication with Client Puzzles. In *8th International Workshop on Security Protocols*, pages 170–177. Springer, Apr. 2001. `http://www.tml.hut.fi/~pnr/publications/cam2000-aura.ps`.

[25] J. Baek, J. Newmarch, R. Safavi-naini, and W. Susilo. A Survey of Identity-Based Cryptography. In *Proc. of Australian Unix Users Group Annual Conference*, pages 95–102, 2004.

[26] M. Bagnulo, P. Matthews, and I. van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146 (Proposed Standard), Apr. 2011.

[27] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147 (Proposed Standard), Apr. 2011.

[28] F. Baker, E. Lear, and R. Droms. Procedures for Renumbering an IPv6 Network without a Flag Day. RFC 4192 (Informational), Sept. 2005.

[29] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 343–352, New York, NY, USA, 2004. ACM.

[30] H. Ballani, P. Francis, T. Cao, and J. Wang. Making Routers Last Longer with ViAggre. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 453–466, Berkeley, CA, USA, 2009. USENIX Association.

[31] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7):1655–1695, 2007.

[32] T. Bates and Y. Rekhter. Scalable Support for Multi-homed Multi-provider Connectivity. RFC 2260 (Informational), Jan. 1998.

[33] J. Beal and T. Shepard. Deamplification of DoS Attacks via Puzzles, Oct. 2004.

[34] S. Bellovin, J. Schiller, and C. Kaufman. Security Mechanisms for the Internet. RFC 3631 (Informational), Dec. 2003.

[35] M. C. Benvenuto and A. D. Keromytis. EasyVPN: IPsec Remote Access Made Easy. In *Proceedings of the 17th USENIX conference on System administration*, pages 87–94, Berkeley, CA, USA, 2003. USENIX Association.

[36] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396 (Draft Standard), Aug. 1998. Obsoleted by RFC 3986, updated by RFC 2732.

[37] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World. *Commun. ACM*, 53(2):66–75, Feb. 2010.

[38] B. Bishaj. Backwards Compatibility Experimentation with Host Identity Protocol and Legacy Software and Networks, June 2008.

[39] M. Blanchet and P. Seite. Multiple Interfaces and Provisioning Domains Problem Statement. RFC 6418 (Informational), Nov. 2011.

[40] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro. Realm Specific IP: Framework. RFC 3102 (Experimental), Oct. 2001.

[41] C. Boulton, J. Rosenberg, G. Camarillo, and F. Audet. NAT Traversal Practices for Client-Server SIP. RFC 6314 (Informational), July 2011.

[42] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), Oct. 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633.

[43] S. Bradner, A. Mankin, and J. I. Schiller. A Framework for Purpose-Built Keys (PBK).

[44] M. Buddhikot, A. Hari, K. Singh, and S. Miller. MobileNAT: a New Technique for Mobility Across Heterogeneous Address Spaces. *Mob. Netw. Appl.*, 10(3):289–302, June 2005.

[45] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. Rofl: routing on flat labels. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 363–374, New York, NY, USA, 2006. ACM.

[46] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588 (Proposed Standard), Sept. 2003. Updated by RFCs 5729, 5719, 6408.

[47] T. Callahan, M. Allman, and V. Paxson. A longitudinal view of HTTP traffic. In *Proceedings of the 11th international conference on Passive and active measurement*, PAM'10, pages 222–231, Berlin, Heidelberg, 2010. Springer-Verlag.

[48] G. Camarillo, I. Mas, and P. Nikander. A framework to combine the session initiation protocol and the host identity protocol. In *WCNC*, pages 3051–3056. IEEE, 2008.

[49] G. Camarillo and J. Melen. Host Identity Protocol (HIP) Immediate Carriage and Conveyance of Upper-Layer Protocol Signaling (HICCUPS). RFC 6078 (Experimental), Jan. 2011.

[50] G. Camarillo, P. Nikander, J. Hautakorpi, A. Keranen, and A. Johnston. HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment (BONE). RFC 6079 (Experimental), Jan. 2011.

[51] B. Carpenter. Internet Transparency. RFC 2775 (Informational), Feb. 2000.

[52] B. Carpenter, R. Atkinson, and H. Flinck. Renumbering Still Needs Work. RFC 5887 (Informational), May 2010.

[53] B. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. RFC 3234 (Informational), Feb. 2002.

[54] G. Caruana and M. Li. A Survey of Emerging Approaches to Spam Filtering. *ACM Comput. Surv.*, 44(2):9:1–9:27, Mar. 2008.

[55] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod Routing Architecture. RFC 1992 (Informational), Aug. 1996.

[56] H. K. Catharina Candolin, Janne Lundberg. Packet Level Authentication in Military Networks. In *Proceedings of the 6th Australian Information Warfare & IT Security Conference*, Geelong, Australia, Nov. 2005.

[57] G. Chen, K. Minami, and D. Kotz. Naming and Discovery in Mobile Systems. In P. Bellavista and A. Corradi, editors, *The Handbook of Mobile Middleware*, chapter 16, pages 387–407. 2006.

[58] D. R. Cheriton and M. Gritter. TRIAD: a Scalable Deployable NAT-based Internet Architecture, Jan. 2000.

[59] S. Cheshire, Z. Zhu, R. Wakikawa, and L. Zhang. Understanding Apple's Back to My Mac (BTMM) Service. RFC 6281 (Informational), June 2011.

[60] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the Addressing Architecture. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, FDNA '03, pages 313–321, New York, NY, USA, 2003. ACM.

[61] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: an Argument for Network Pluralism. *SIGCOMM Comput. Commun. Rev.*, 33(4):258–266, Aug. 2003.

[62] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963 (Proposed Standard), Jan. 2005.

[63] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[64] A. L. Dul. Global IP Network Mobility using Border Gateway Protocol (BGP), Mar. 2006.

[65] L. Eggert and F. Gont. TCP User Timeout Option. RFC 5482 (Proposed Standard), Mar. 2009.

[66] T. Ernst. Network Mobility Support Goals and Requirements. RFC 4886 (Informational), July 2007.

[67] T. Ernst and H.-Y. Lach. Network Mobility Support Terminology. RFC 4885 (Informational), July 2007.

[68] P. Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555 (Proposed Standard), June 2006.

[69] P. Eronen, H. Tschofenig, and Y. Sheffer. An Extension for EAP-Only Authentication in IKEv2. RFC 5998 (Proposed Standard), Sept. 2010.

[70] K. Fall. A Delay-tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[71] P. Faltstrom and G. Huston. A Survey of Internet Identities, Dec. 2004. An expired Internet draft.

[72] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP), Feb. 2012. Work in progress.

[73] D. Farinacci, D. Lewis, D. Meyer, and C. White. LISP Mobile Node, 2011 Oct. Work in progress.

[74] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFCs 2817, 5785, 6266, 6585.

[75] T. Finez. Efficient Leap of Faith Security with Host Identity Protocol, Dec. 2008.

[76] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. RFC 6182 (Informational), Mar. 2011.

[77] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer Communication Across Network Address Translators. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 13–13, Berkeley, CA, USA, 2005. USENIX Association.

[78] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. Persistent Personal Names for Globally Connected Mobile Devices. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, Seattle, Washington, November 2006.

[79] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. User-relative names for globally connected personal devices. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS06)*, Santa Barbara, CA, February 2006.

[80] U. Forum. *"Internet Gateway Device (IGD) V 2.0"*, Apr. 2012.

[81] P. Francis and R. Gummadi. Ipnl: A nat-extended internet architecture. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 69–80, New York, NY, USA, 2001. ACM.

[82] A. Freier, P. Karlton, and P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Historic), Aug. 2011.

[83] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis. LISP Alternative Topology (LISP+ALT), 2011 Dec. Work in progress.

[84] M. Goff. *Network Distributed Computing: Fitscapes and Fallacies*. Prentice Hall Professional Technical Reference, 2003.

[85] J. T. Goodman and R. Rounthwaite. Stopping Outgoing Spam. In *Proceedings of the 5th ACM conference on Electronic commerce*, EC '04, pages 30–39, New York, NY, USA, 2004. ACM.

[86] X. Gu. Host Identity Protocol Version 2.5, June 2012.

[87] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382 (Best Current Practice), Oct. 2008.

[88] S. Guha and P. Francis. Characterization and Measurement of TCP Traversal through NATs and Firewalls. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 18–18, Berkeley, CA, USA, 2005. USENIX Association.

[89] S. Guha and P. Francis. An End-middle-end Approach to Connection Establishment. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 193–204, New York, NY, USA, 2007. ACM.

[90] A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley Publishing, 2008.

[91] A. Gurtov and T. Polishchuk. Secure Multipath Transport for Legacy Internet Applications. In *Broadband Communications, Networks, and Systems, 2009. BROADNETS 2009. Sixth International Conference on*, pages 1 –8, sept. 2009.

[92] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608 (Proposed Standard), June 1999. Updated by RFC 3224.

[93] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Informational), Apr. 2010.

[94] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Seshan, and P. Steenkiste. XIA: Efficient support for evolvable internetworking. In *Proc. 9th USENIX NSDI*, San Jose, CA, Apr. 2012.

[95] T. Heer. Direct End-to-Middle Authentication in Cooperative Networks, Dec. 2011.

[96] T. Heer and S. Varjonen. Host Identity Protocol Certificates. RFC 6253 (Experimental), May 2011.

[97] S. Heikkinen. Applicability of Host Identities in Securing Network Attachment and Ensuring Service Accountability, Nov. 2011.

[98] T. Henderson and A. Gurtov. The Host Identity Protocol (HIP) Experiment Report. RFC 6538 (Informational), Mar. 2012.

[99] T. Henderson, P. Nikander, and M. Komu. Using the Host Identity Protocol with Legacy Applications. RFC 5338 (Experimental), Sept. 2008.

[100] T. Henderson, S. C. Venema, and D. Mattes. HIP-based Virtual Private LAN Service (HIPLS), Mar. 2012.

[101] T. R. Henderson. Host Mobility for IP Networks: A Comparison. *IEEE Network*, 17(6):18–26, Nov. 2003.

[102] S. Herborn, A. Huber, R. Boreli, and A. Seneviratne. Secure host identity delegation for mobility. In *COMSWARE*. IEEE, 2007.

[103] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006. Updated by RFCs 5952, 6052.

[104] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193 (Proposed Standard), Oct. 2005.

[105] C. Huitema. Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs). RFC 4380 (Proposed Standard), Feb. 2006. Updated by RFCs 5991, 6081.

[106] C. Huitema and B. Carpenter. Deprecating Site Local Addresses. RFC 3879 (Proposed Standard), Sept. 2004.

[107] G. Huston. Architectural Approaches to Multi-homing for IPv6. RFC 4177 (Informational), Sept. 2005.

[108] G. Huston. Transitioning Protocols, Mar. 2011.

[109] G. Iapichino and C. Bonnet. Host Identity Protocol and Proxy Mobile IPv6: a Secure Global and Localized Mobility Management Scheme for Multihomed Mobile Nodes. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 578–583, Piscataway, NJ, USA, 2009. IEEE Press.

[110] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[111] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. *SIGCOMM Comput. Commun. Rev.*, 34(4):145–158, Aug. 2004.

[112] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang, and L. Zhang. Towards a New Internet Routing Architecture: Arguments for Separating Edges from Transit Core. *HotNets-VII*, October 2008.

[113] C. Jennings, B. B. Lowekamp, E. Rescorla, S. A. Baset, and H. Schulzrinne. *REsource LOcation And Discovery (RELOAD) Base Protocol*. Internet Engineering Task Force, Oct. 2011. Internet draft, work in progress.

[114] C. Jennings, B. B. Lowekamp, E. Rescorla, S. A. Baset, and H. Schulzrinne. REsource LOcation And Discovery (RELOAD) Base Protocol, Mar. 2012. Internet draft, work in progress.

[115] P. Jokela, R. Moskowitz, and J. Melen. Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP), July 2012. Internet draft, work in progress.

[116] P. Jokela, R. Moskowitz, and P. Nikander. Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). RFC 5202 (Experimental), Apr. 2008.

[117] P. Jokela, P. Nikander, J. Melen, J. Ylitalo, and J. Wall. Host Identity Protocol: Achieving IPv4 - IPv6 handovers without tunneling. In *in Proceedings of Evolute workshop 2003: "Beyond 3G Evolution of Systems and Services"*, Nov. 2003.

[118] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: Line Speed Publish/subscribe Inter-networking. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 195–206, New York, NY, USA, 2009. ACM.

[119] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamalytics: an Empirical Analysis of Spam Marketing Conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 3–14, New York, NY, USA, 2008. ACM.

[120] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), Dec. 2005. Obsoleted by RFC 5996, updated by RFC 5282.

[121] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), Dec. 2005. Updated by RFC 6040.

[122] A. Keränen, G. Camarillo, and J. Mäenpää. *Host Identity Protocol-Based Overlay Networking Environment (HIP BONE) Instance Specification for REsource LOcation And Discovery (RELOAD)*. Internet Engineering Task Force, Apr. 2011. Internet draft, work in progress.

[123] V. Khare, D. Jen, X. Zhao, Y. Liu, D. Massey, L. Wang, B. Zhang, and L. Zhang. Evolution Towards Global Routing Scalability. *IEEE Journal on Selected Areas in Communications*, 28(8):1363–1375, 2010.

[124] A. Khurri. Evaluating IP Security on Lightweight Hardware, Jan. 2011. ISBN 978-952-60-4004-2.

[125] S. L. Kinney. *Trusted Platform Module Basics: Using TPM in Embedded Systems (Embedded Technology)*. Newnes, 2006.

[126] T. Kivinen and H. Tschofenig. Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol. RFC 4621 (Informational), Aug. 2006.

[127] M. Komu, M. Bagnulo, K. Slavov, and S. Sugimoto. Sockets Application Program Interface (API) for Multihoming Shim. RFC 6316 (Informational), July 2011.

[128] M. Komu and T. Henderson. Basic Socket Interface Extensions for the Host Identity Protocol (HIP). RFC 6317 (Experimental), July 2011.

[129] M. Komu, T. Henderson, H. Tschofenig, J. Melen, and A. Keranen. Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators. RFC 5770 (Experimental), Apr. 2010.

[130] M. Komu, M. Sethi, R. Mallavarapu, H. Oirola, R. Khan, and S. Tarkoma. Secure Networking for Virtual Machines in the Cloud. In *The 2012 International Workshop on Power and QoS Aware Computing (PQoSCom'12), held in conjunction with IEEE Cluster'12*. IEEE, sep 2012.

[131] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-oriented (and beyond) Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, Aug. 2007.

[132] T. Koponen, P. Eronen, and M. Särelä. Resilient connections for SSH and TLS. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, ATEC '06, pages 30–30, Berkeley, CA, USA, 2006. USENIX Association.

[133] T. Koponen, J. Lindqvist, N. Karlsson, E. Vehmersalo, M. Komu, M. Kousa, D. Korzun, and A. Gurtov. Overview and Comparison Criteria for the Host Identity Protocol and Related Technologies, Nov. 2005.

[134] J. Korhonen. IP Mobility in Wireless Operator Networks, Nov. 2008. 978-952-10-5014-5.

[135] J. Koskela and S. Tarkoma. Simple Peer-to-Peer SIP Privacy. In *Security and Privacy in Mobile Information and Communication Systems*, volume 17 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 226–237. Springer Berlin Heidelberg, 2009.

[136] D. V. Krioukov and K. C. Claffy. Toward Compact Interdomain Routing. *CoRR*, abs/cs/0508021, 2005.

[137] M. Kucherawy and D. Crocker. Email Greylisting: An Applicability Statement for SMTP. RFC 6647 (Proposed Standard), June 2012.

[138] M. Kulkarni, A. Patel, and K. Leung. Mobile IPv4 Dynamic Home Agent (HA) Assignment. RFC 4433 (Proposed Standard), Mar. 2006.

[139] M. Kunishi, M. Ishiyama, K. Uehara, H. Esaki, and F. Teraoka. LIN6: A New Approach to Mobility Support in IPv6. In *in Proc. of the Third International Symposium on Wireless Personal Multimedia Communications*, nov 2000.

[140] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), Aug. 2007.

[141] J. Laganier and L. Eggert. Host Identity Protocol (HIP) Rendezvous Extension. RFC 5204 (Experimental), Apr. 2008.

[142] J. Laganier, T. Koponen, and L. Eggert. Host Identity Protocol (HIP) Registration Extension. RFC 5203 (Experimental), Apr. 2008.

[143] D. Lagutin. Packet Level Authentication (PLA) Extensions for Host Identity Protocol (HIP), July 2010.

[144] D. Lagutin. Securing the Internet with Digital Signatures, Dec. 2010.

[145] D. Lagutin and H. Kari. Controlling Incoming Connections Using Certificates and Distributed Hash Tables. In Y. Koucheryavy, J. Harju, and A. Sayenko, editors, *NEW2AN*, volume 4712 of *Lecture Notes in Computer Science*, pages 455–467. Springer, 2007.

[146] A. Langley. Transport Layer Security (TLS) Snap Start, June 2010. Expired Internet draft.

[147] A. Langley, N. Modadugu, and B. Moeller. Transport Layer Security (TLS) False Start, June 2010. Expired Internet draft.

[148] E. Lear. NERD: A Not-so-novel EID to RLOC Database, Apr. 2012. Work in progress, expires in October 2012.

[149] E. Lear and R. Droms. What's In A Name: Thoughts from the NSRG. Internet-draft, IETF Secretariat, Fremont, CA, USA, Sept. 2003.

[150] K. Leung, G. Dommety, V. Narayanan, and A. Petrescu. Network Mobility (NEMO) Extensions for Mobile IPv4. RFC 5177 (Proposed Standard), Apr. 2008. Updated by RFC 6626.

[151] T. Li. Design Goals for Scalable Internet Routing. RFC 6227 (Informational), May 2011.

[152] T. Li. Recommendation for a Routing Architecture. RFC 6115 (Informational), Feb. 2011.

[153] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard), Apr. 2010.

[154] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 90–102, New York, NY, USA, 2009. ACM.

[155] J. Manner and M. Kojo. Mobility Related Terminology. RFC 3753 (Informational), June 2004.

[156] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[157] A. Medina, M. Allman, and S. Floyd. Measuring interactions between transport protocols and middleboxes. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 336–341, New York, NY, USA, 2004. ACM.

[158] J. Melen, J. Ylitalo, and P. Salmela. Host Identity Protocol-based Mobile Proxy, Aug. 2009. An expired Internet draft.

[159] C. Metz and J. ichiro itojun Hagino. IPv4-Mapped Addresses on the Wire Considered Harmful, Oct. 2003. Work in progress, expired in Oct, 2003.

[160] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984 (Informational), Sept. 2007.

[161] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, Apr. 2004.

[162] A. Mishra, M. Shin, and W. Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *SIGCOMM Comput. Commun. Rev.*, 33(2):93–102, Apr. 2003.

[163] G. Montenegro. Reverse Tunneling for Mobile IP, revised. RFC 3024 (Proposed Standard), Jan. 2001.

[164] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423 (Informational), May 2006.

[165] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), Apr. 2008. Updated by RFC 6253.

[166] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021, 5896, 6111, 6112, 6113, 6649.

[167] C. Ng, P. Thubert, M. Watari, and F. Zhao. Network Mobility Route Optimization Problem Statement. RFC 4888 (Informational), July 2007.

[168] P. Nie, J. Vähä-Herttua, T. Aura, and A. Gurtov. Performance Analysis of HIP Diet Exchange for WSN Security Establishment. In *Proceedings of the 7th ACM symposium on QoS and security for wireless and mobile networks*, Q2SWinet '11, pages 51–56, New York, NY, USA, 2011. ACM.

[169] P. Nikander, A. Gurtov, and T. R. Henderson. Host identity protocol (hip): Connectivity, mobility, multi-homing, security, and privacy over ipv4 and ipv6 networks. *Commun. Surveys Tuts.*, 12(2):186–204, Apr. 2010.

[170] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-Host Mobility and Multihoming with the Host Identity Protocol. RFC 5206 (Experimental), Apr. 2008.

[171] P. Nikander and J. Laganier. Host Identity Protocol (HIP) Domain Name System (DNS) Extensions. RFC 5205 (Experimental), Apr. 2008.

[172] P. Nikander, J. Laganier, and F. Dupont. An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID). RFC 4843 (Experimental), Apr. 2007.

[173] P. Nikander and K. Slavov. Proxying Approach to SHIM6 and HIP (PASH), Feb. 2007. An expired Internet draft.

[174] P. Nikander, J. Wall, and J. Ylitalo. Integrating Security, Mobility, and Multi-Homing in a HIP Way. In *Proceedings of Network and Distributed Systems Security Symposium*, pages 87–99, San Diego, California, Feb. 2003. Internet Society. `http://www.tcm.hut.fi/~pnr/publications/NDSS03-Nikander-et-al.pdf`.

[175] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533 (Proposed Standard), June 2009.

[176] M. O'Dell. GSE - An Alternate Addressing Architecture for IPv6, Feb. 1997.

[177] L. Ong and J. Yoakum. An Introduction to the Stream Control Transmission Protocol (SCTP). RFC 3286 (Informational), May 2002.

[178] R. H. Paine. *Beyond HIP: The End to Hacking As We Know It*. BookSurge Publishing, 2009.

[179] J. Pan, S. Paul, and R. Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26 –36, july 2011.

[180] S. Paul, J. Pan, and R. Jain. A Survey of Naming Systems: Classification and Analysis of the Current Schemes Using a New Naming Reference Model, 2009.

[181] S. Paul, J. Pan, and R. Jain. Architectures for the future networks and the next generation internet: A survey. *Comput. Commun.*, 34(1):2–42, Jan. 2011.

[182] X. Pérez-Costa, M. Torrent-Moreno, and H. Hartenstein. A Performance Comparison of Mobile IPv6, Hierarchical Mobile IPv6, Fast Handovers for Mobile IPv6 and Their Combination. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(4):5–19, Oct. 2003.

[183] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), Nov. 2010.

[184] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275 (Proposed Standard), July 2011.

[185] H. Petander. A Network Mobility Management Architecture for a Heterogeneous Network Environment, Dec. 2007. ISBN 978-951-22-9098-7.

[186] V. Pham and T. Aura. Security Analysis of Leap-of-Faith Protocols. In *Seventh ICST International Conference on Security and Privacy for Communication Networks*, Sept. 2011.

[187] S. Pierrel, P. Jokela, J. Melen, and K. Slavov. *A Policy System for Simultaneous Multiaccess with Host Identity Protocol*. Munich, Germany, May 2007.

[188] V. K. Pingali, A. Falk, T. Faber, and R. Braden. Farads prototype design document, june 2003.

[189] O. Ponomarev and A. Gurtov. Embedding Host Identity Tags Data in DNS, 2009. An expired Internet draft.

[190] O. Ponomarev, A. Khurri, and A. Gurtov. Elliptic Curve Cryptography (ECC) for Host Identity Protocol (HIP). In *Proceedings of the 2010 Ninth International Conference on Networks*, ICN '10, pages 215–219, Washington, DC, USA, 2010. IEEE Computer Society.

[191] L. Popa, A. Ghodsi, and I. Stoica. HTTP as the Narrow Waist of the Future Internet. In *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets '10, pages 6:1–6:6, New York, NY, USA, 2010. ACM.

[192] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959 (Standard), Oct. 1985. Updated by RFCs 2228, 2640, 2773, 3659, 5797.

[193] R. Raghavendra, E. M. Belding, K. Papagiannaki, and K. C. Almeroth. Understanding Handoffs in Large IEEE 802.11 Wireless Networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 333–338, New York, NY, USA, 2007. ACM.

[194] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), Apr. 2006. Obsoleted by RFC 6347, updated by RFC 5746.

[195] J. Rexford and C. Dovrolis. Future Internet Architecture: Clean-slate versus Evolutionary Research. *Commun. ACM*, 53(9):36–40, Sept. 2010.

[196] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865 (Draft Standard), June 2000. Updated by RFCs 2868, 3575, 5080.

[197] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard), Apr. 2010. Updated by RFC 6336.

[198] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard), Oct. 2008.

[199] P. Salmela and J. Melén. Host Pdentity Protocol Proxy. In J. Filipe and L. Vasiu, editors, *ICETE*, pages 222–230. INSTICC Press, 2005.

[200] J. Saltzer. Naming and Binding of Objects. In *Operating Systems, Lecture notes in Computer Science, Vol. 60*. Springer-Verlag, 1978.

[201] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, Nov. 1984.

[202] M. Scharf and A. Ford. MPTCP Application Interface Considerations, Oct. 2012. Work in progress.

[203] J. Schlyter and W. Griffin. Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. RFC 4255 (Proposed Standard), Jan. 2006.

[204] S. Schütz, L. Eggert, S. Schmid, and M. Brunner. Protocol enhancements for intermittently connected hosts. *SIGCOMM Comput. Commun. Rev.*, 35(3):5–18, July 2005.

[205] R. Seggelmann, M. Tüxen, and E. P. Rathgeb. DTLS Mobility. In *Proceedings of the 13th international conference on Distributed Computing and Networking*, ICDCN'12, pages 443–457, Berlin, Heidelberg, 2012. Springer-Verlag.

[206] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer. Session Initiation Protocol (SIP) Session Mobility. RFC 5631 (Informational), Oct. 2009.

[207] Z. Shelby and C. Bormann. *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010.

[208] C. Shields and J. J. Garcia-Luna-Aceves. The HIP Protocol for Hierarchical Multicast Routing. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, PODC '98, pages 257–266, New York, NY, USA, 1998. ACM.

[209] J. Shoch. Inter-Network Naming, Addressing, and Routing. In *IEEE Proc. COMPCON*, pages 72–79. IEEE, 1978.

[210] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek. Reconsidering Internet Mobility. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HOTOS '01, pages 41–, Washington, DC, USA, 2001. IEEE Computer Society.

[211] R. Sofia, P. Nesser, and II. Survey of IPv4 Addresses in Currently Deployed IETF Application Area Standards Track and Experimental Documents. RFC 3795 (Informational), June 2004.

[212] H. Soliman. Mobile IPv6 Support for Dual Stack Hosts and Routers. RFC 5555 (Proposed Standard), June 2009.

[213] P. Srisuresh, B. Ford, and D. Kegel. State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs). RFC 5128 (Informational), Mar. 2008.

[214] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), Sept. 2007. Updated by RFCs 6096, 6335.

[215] R. Stewart, M. Tuexen, K. Poon, P. Lei, and V. Yasevich. Sockets API Extensions for the Stream Control Transmission Protocol (SCTP). RFC 6458 (Informational), Dec. 2011.

[216] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. RFC 5061 (Proposed Standard), Sept. 2007.

[217] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. *IEEE/ACM Trans. Netw.*, 12(2):205–218, Apr. 2004.

[218] A. K. Tapio Levä, Miika Komu and S. Luukkainen. Adoption Barriers of Network-layer Protocols: the Case of Host Identity Protocol. In *The International Journal of Computer and Telecommunications Networking*. Elsevier, Nov. 2012. Unpublished manuscript, accepted to Elsevier COMNET journal.

[219] F. Templin. The Internet Routing Overlay Network (IRON). RFC 6179 (Experimental), Mar. 2011.

[220] D. Thaler. Evolution of the IP Model. RFC 6250 (Informational), May 2011.

[221] D. Thaler and B. Aboba. What Makes For a Successful Protocol? RFC 5218 (Informational), July 2008.

[222] N. D. Tom Scavo. Shibboleth Architecture, Technical Overview, 2005 June.

[223] J. Touch, D. Black, and Y. Wang. Problem and Applicability Statement for Better-Than-Nothing Security (BTNS). RFC 5387 (Informational), Nov. 2008.

[224] J. Touch and R. Perlman. Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement. RFC 5556 (Informational), May 2009.

[225] S. Tritilanunt, C. Boyd, E. Foo, and J. M. G. Nieto. Examining the DoS Resistance of HIP. In *OTM Workshops (1)*, volume 4277 of *Lecture Notes in Computer Science*, pages 616–625. Springer, 2006.

[226] H. Tschofenig, M. Shanmugam, and F. Muenz. *Using SRTP transport format with HIP*. IETF, Aug. 2006. expired Internet draft.

[227] G. Tsirtsis, V. Park, and H. Soliman. Dual-Stack Mobile IPv4. RFC 5454 (Proposed Standard), Mar. 2009.

[228] Z. Turányi, A. Valkó, and A. T. Campbell. 4+4: an Architecture for Evolving the Internet Address Space Back Toward Transparency. *SIGCOMM Comput. Commun. Rev.*, 33(5):43–54, Oct. 2003.

[229] S. Turner and L. Chen. Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms. RFC 6151 (Informational), Mar. 2011.

[230] J. Ubillos, M. Xu, Z. Ming, and C. Vogt. Name-Based Sockets Architecture, Sept. 2010. Experimental and expired Internet draft, work in progress.

[231] P. Urien, D. Nyami, S. Elrharbi, H. Chabanne, T. Icart, C. Pépin, M. Bouet, D. D. O. Cunha, V. Guyot, G. Pujolle, E. Gressier-Soudan, and J.-F. Susini. HIP Tags Privacy Architecture. In *Proceedings of the 2008 Third International Conference on Systems and Networks Communications*, ICSNC '08, pages 179–184, Washington, DC, USA, 2008. IEEE Computer Society.

[232] S. Varjonen. Secure Connectivity With Persistent Identities, Nov. 2012.

[233] S. Varjonen, T. Heer, K. Rimey, and A. Gurtov. Secure Resolution of End-Host Identifiers for Mobile Clients. In *IEEE GLOBECOM 2011 - Next Generation Networking Symposium (GC'11 - NGN), Awarded the NGN Best Paper Award*, Piscataway, NJ, USA, 12 2011. IEEE.

[234] S. Varjonen, M. Komu, and A. Gurtov. Secure and Efficient IPv4/IPv6 Handovers using Host-based Identifier-locator Split. In *SoftCOM'09: Proceedings of the 17th international conference on Software, Telecommunications and Computer Networks*, pages 111–115, Piscataway, NJ, USA, 2009. IEEE Press.

[235] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), Apr. 1997. Updated by RFCs 3007, 4035, 4033, 4034.

[236] C. Vogt. Six/one Router: a Scalable and Backwards Compatible Solution for Provider-independent Addressing. In *Proceedings of the 3rd international workshop on Mobility in the evolving internet architecture*, MobiArch '08, pages 13–18, New York, NY, USA, 2008. ACM.

[237] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, Berkeley, CA, USA, 2004. USENIX Association.

[238] M. Wasserman and P. Seite. Current Practices for Multiple-Interface Hosts. RFC 6419 (Informational), Nov. 2011.

[239] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. RFC 3007 (Proposed Standard), Nov. 2000.

[240] R. Whittle. Ivip (Internet Vastly Improved Plumbing) Architecture, 2010 Mar.

[241] N. Williams. IPsec Channels: Connection Latching. RFC 5660 (Proposed Standard), Oct. 2009.

[242] N. Williams and M. Richardson. Better-Than-Nothing Security: An Unauthenticated Mode of IPsec. RFC 5386 (Proposed Standard), Nov. 2008.

[243] D. Wing, S. Cheshire, M. Boucadair, R. Penno, and P. Selkirk. Port Control Protocol, Mar. 2012.

[244] D. Wing, P. Patil, and T. Reddy. Mobility with ICE (MICE), July 2012. Internet draft, work in progress.

[245] K. Winstein and H. Balakrishnan. Mosh: An Interactive Remote Shell for Mobile Clients. In *USENIX Annual Technical Conference*, Boston, MA, June 2012.

[246] J. Wu, J. Bi, X. Li, G. Ren, K. Xu, and M. Williams. A Source Address Validation Architecture (SAVA) Testbed and Deployment Experience. RFC 5210 (Experimental), June 2008.

[247] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, Aug. 2008.

[248] H. Yin and H. Wang. Building an Application-aware IPsec Policy System. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, pages 21–21, Berkeley, CA, USA, 2005. USENIX Association.

[249] J. Ylitalo. Secure Mobility at Multiple Granularity Levels over Heterogeneous Datacom Networks, Nov. 2008. ISBN 978-951-22-9530-2.

[250] J. Ylitalo and P. Nikander. "A New Name Space for End-Points: Implementing Secure Mobility and Multi-homing across the Two Versions of IP". In *in Proc. of the Fifth European Wireless Conference*, Mobile and Wireless Systems beyond 3G, pages pp. 435–441. SCI UPC (Eds: Olga Casals, Jorge Carcia-Vidal, Jose M Barcelo, and Llorenc Cerda), Feb. 2004.

[251] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard), Jan. 2006.

[252] V. C. Zandy and B. P. Miller. Reliable network connections. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, MobiCom '02, pages 95–106, New York, NY, USA, 2002. ACM.

[253] D. Zhang, D. Kuptsov, and S. Shen. Host Identifier Revocation in HIP, Mar. 2012. Internet draft, work in progress.

[254] D. Zhang, X. Xu, J. Yao, and Z. Cao. Investigation in HIP Proxies, Oct. 2011. Work in progress, Internet draft.

[255] L. Zhang. An Overview of Multihoming and Open Issues in GSE, Sept. 2006.

[256] Z. Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. *Communications Surveys & Tutorials, IEEE*, 8(1):24–37, Mar. 2006.

[257] X. Zhu, Z. Ding, and X. Wang. A Multicast Routing Algorithm Applied to HIP-Multicast Model. In *Proceedings of the 2011 International Conference on Network Computing and Information Security - Volume 01*, NCIS '11, pages 169–174, Washington, DC, USA, 2011. IEEE Computer Society.

Errata

Publication III

Equation 1 is missing the explanation for C_0; it refers to the computation time of first received puzzle.
The p_Z variable is missing a coefficient in equations 10 and 13-15 in section 4.5. In the text of the publication, this should be fixed by replacing p_Z with p'_Z that equals to $\frac{\alpha}{1-\alpha} \cdot p_Z$. Fortunately, the example plot in Figure 4 remains valid because p'_Z equals to one with the chosen value (0.5) for α.
Further in the paper, two new variables are introduced even though old ones could be reused: $\lambda_L = \lambda_C$ and $\lambda_S = \lambda_Z$.

Publication IV

The journal article contains a few minor typos (“the the”, “as as”)

Publication V

Reference number 32 does not exist in the bibliography, and should be replaced with reference number 31 in the text.

Publication VI

The unabbreviated “EV” acronym stands for Extended Validation

Aalto-DD 166/2012

ISBN 978-952-60-4904-5
ISBN 978-952-60-4905-2 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science and Engineering
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS