

TKK Dissertations 140
Espoo 2008

A DATA-ORIENTED NETWORK ARCHITECTURE

Doctoral Dissertation

Teemu Koponen



**Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering**

TKK Dissertations 140
Espoo 2008

A DATA-ORIENTED NETWORK ARCHITECTURE

Doctoral Dissertation

Teemu Koponen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Information and Natural Sciences for public examination and debate in Auditorium T1 at Helsinki University of Technology (Espoo, Finland) on the 2nd of October, 2008, at 12 noon.

**Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering**

**Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan laitos**

Distribution:

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering
P.O. Box 5400
FI - 02015 TKK
FINLAND
URL: <http://cse.tkk.fi/>
Tel. +358-9-4511

© 2008 Teemu Koponen

ISBN 978-951-22-9559-3
ISBN 978-951-22-9560-9 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: <http://lib.tkk.fi/Diss/2008/isbn9789512295609/>

TKK-DISS-2510

Picaset Oy
Helsinki 2008



ABSTRACT OF DOCTORAL DISSERTATION		HELSINKI UNIVERSITY OF TECHNOLOGY P. O. BOX 1000, FI-02015 TKK http://www.tkk.fi					
Author		Teemu Koponen					
Name of the dissertation A Data-Oriented Network Architecture							
Manuscript submitted		09.06.2008		Manuscript revised		12.09.2008	
Date of the defence				02.10.2008			
<input type="checkbox"/> Monograph		<input checked="" type="checkbox"/> Article dissertation (summary + original articles)					
Faculty		Information and Natural Sciences					
Department		Computer Science and Engineering					
Field of research		Networking					
Opponent(s)		Professor Jon Crowcroft					
Supervisor		Professor Antti Ylä-Jääski					
Instructor(s)		Dr. Pekka Nikander and Professor Scott Shenker					
Abstract				<p>In the 25 years since becoming commercially available, the Internet has grown into a global communication infrastructure connecting a significant part of mankind and become an important part of modern society. Its impressive growth has been fostered by innovative applications, many of which were completely unforeseen by the Internet's inventors. While fully acknowledging ingenuity and creativity of application designers, it is equally impressive how little the core architecture of the Internet has evolved during this time. However, the ever evolving applications and growing importance of the Internet have resulted in increasing discordance between the Internet's current use and its original design. In this thesis, we focus on four sources of discomfort caused by this divergence.</p> <p>First, the Internet was developed around host-to-host applications, such as telnet and ftp, but the vast majority of its current usage is service access and data retrieval. Second, while the freedom to connect from any host to any other host was a major factor behind the success of the Internet, it provides little protection for connected hosts today. As a result, distributed denial of service attacks against Internet services have become a common nuisance, and are difficult to resolve within the current architecture. Third, Internet connectivity is becoming nearly ubiquitous and reaches increasingly often mobile devices. Moreover, connectivity is expected to extend its reach to even most extreme places. Hence, applications' view to network has changed radically; it's commonplace that they are offered intermittent connectivity at best and required to be smart enough to use heterogeneous network technologies. Finally, modern networks deploy so-called middleboxes both to improve performance and provide protection.</p> <p>In this thesis, we design a clean-slate network architecture that is a better fit with the current use of the Internet. We present a name resolution system based on name-based routing. It matches with the service access and data retrieval oriented usage of the Internet, and takes the network imposed middleboxes properly into account. We then propose modest addressing-related changes to the network layer as a remedy for the denial of service attacks. Finally, we take steps towards a data-oriented communications API that provides better decoupling for applications from the network stack than the original Sockets API does. The improved decoupling both simplifies applications and allows them to be unaffected by evolving network technologies: in this architecture, coping with intermittent connectivity and heterogenous network technologies is a burden of the network stack.</p>			
Keywords		Internet architecture, naming, name resolution, addressing, communication abstractions					
ISBN (printed)		978-951-22-9559-3		ISSN (printed)		1795-2239	
ISBN (pdf)		978-951-22-9560-9		ISSN (pdf)		1795-4584	
Language		English		Number of pages		56 p. + app. 44 p.	
Publisher				Department of Computer Science and Engineering			
Print distribution				Department of Computer Science and Engineering			
<input checked="" type="checkbox"/> The dissertation can be read at http://lib.tkk.fi/Diss/2008/isbn9789512295609/							



VÄITÖSKIRJAN TIIVISTELMÄ		TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.tkk.fi	
Tekijä Teemu Koponen			
Väitöskirjan nimi A Data-Oriented Network Architecture			
Käsikirjoituksen päivämäärä 09.06.2008		Korjatun käsikirjoituksen päivämäärä 12.09.2008	
Väitöstilaisuuden ajankohta 02.10.2008			
<input type="checkbox"/> Monografia		<input checked="" type="checkbox"/> Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)	
Tiedekunta Informaatio- ja luonnontieteet			
Laitos Tietotekniikan laitos			
Tutkimusala Tietoliikenne			
Vastaväittäjä(t) Professori Jon Crowcroft			
Työn valvoja Professori Antti Ylä-Jääski			
Työn ohjaaja(t) TkT Pekka Nikander ja Professori Scott Shenker			
<p>Tiivistelmä</p> <p>Viimeisen kahdenkymmenenviiden vuoden aikana Internet on kasvanut globaaliksi tietoverkoksi, joka yhdistää ihmisiä ja josta on tullut tärkeä osa modernia yhteiskuntaa. Internetin kasvun taustalla on monia innovatiivisia sovelluksia, jotka ovat muuttaneet ihmisten arkipäivää merkittävästi. Internetin kasvu on ollut vaikuttavaa, mutta yhtä vaikuttavaa on se, miten vähän sen tekninen perusta on muuttunut samana aikana. Internetin kasvaminen ja sovellusten kehittyminen vievät Internetiä yhä kauemmas tehtävästä, johon se alunperin suunniteltiin. Tämä aiheuttaa yhä enemmän teknisiä ristiriitoja.</p> <p>Työssä keskitytään neljän ongelman ratkaisemiseen:</p> <p>1.) Internet kehitettiin puhtaasti tietokoneiden väliseen liikenteeseen, mutta nykyisin valtaosa Internetin käytöstä on tiedon latausta ja palveluiden käyttöä. Tarvittavat tekniset ratkaisut ovat monimutkaisia. 2.) Internet perustuu avoimuuteen ja mikä tahansa verkkoon liitetty tietokone voi muodostaa yhteyden mihin tahansa toiseen tietokoneeseen. Tämä periaate johtaa tietoturvaongelmiin ja hajautetuista palvelunestohyökkäyksistä on tullut vaikea pulma Internet-palveluille. 3.) Internet-yhteyksien määrä on kasvanut ja verkkoon on kytketty yhä useampia monipuolisin yhteystekniikoin varustettuja mobiililaitteita. Samalla tietoverkkojen oletetaan ulottuvan kaikkialle. Tämän vuoksi sovellusten on oltava verkkoa käyttäessään yhä älykkäämpiä. 4.) Verkkoon liitettyjen tietokoneiden suojaamiseksi sekä sovellusten tehostamiseksi verkkoihin asennetaan niin sanottuja middleboxeja. Näiden laitteiden toiminta kuitenkin perustuu Internetin suunnitteluperiaatteiden vastaisiin tekniikoihin, jotka monimutkaistavat verkkoja ja sovellusprotokollia.</p> <p>Väitöskirjassa suunnitellaan uusi verkkoarkkitehtuuri, joka vastaa paremmin Internetin muuttuneita käyttötarpeita. Työ koostuu kolmesta osa-alueesta. Ensimmäisessä osassa laaditaan reititykseen pohjautuva nimipalvelu, jonka ansiosta tiedon latauksesta, palveluiden käytöstä ja middleboxien asentamisesta tulee verkolle luontevaa. Toisessa osassa suunnitellaan verkkokerrokselle osoitteistusmuutoksia, joiden avulla palvelunestohyökkäykset voidaan torjua. Kolmannessa osassa esitellään uusi sovellusrajapinta, joka yksinkertaistaa sovelluksia ja mahdollistaa niiden käyttämisen yhä kehittyvien verkkoteknologioiden kanssa.</p>			
Asiasanat Internet-arkkitehtuuri, nimeäminen, nimenselvitys, osoitteistus, kommunikaatioabstraktiot			
ISBN (painettu) 978-951-22-9559-3		ISSN (painettu) 1795-2239	
ISBN (pdf) 978-951-22-9560-9		ISSN (pdf) 1795-4584	
Kieli Englanti		Sivumäärä 56 s. + liit. 44 s.	
Julkaisija Tietotekniikan laitos			
Painetun väitöskirjan jakelu Tietotekniikan laitos			
<input checked="" type="checkbox"/> Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/2008/isbn9789512295609/			

Abstract

In the 25 years since becoming commercially available, the Internet has grown into a global communication infrastructure connecting a significant part of mankind and has become an important part of modern society. Its impressive growth has been fostered by innovative applications, many of which were completely unforeseen by the Internet's inventors. While fully acknowledging ingenuity and creativity of application designers, it is equally impressive how little the core architecture of the Internet has evolved during this time. However, the ever evolving applications and growing importance of the Internet have resulted in increasing discordance between the Internet's current use and its original design. In this thesis, we focus on four sources of discomfort caused by this divergence.

First, the Internet was developed around host-to-host applications, such as *telnet* and *ftp*, but the vast majority of its current usage is service access and data retrieval. Second, while the freedom to connect from any host to any other host was a major factor behind the success of the Internet, it provides little protection for connected hosts today. As a result, distributed denial of service attacks against Internet services have become a common nuisance, and are difficult to resolve within the current architecture. Third, Internet connectivity is becoming nearly ubiquitous and reaches increasingly often mobile devices. Moreover, connectivity is expected to extend its reach to even most extreme places. Hence, applications' view to network has changed radically; it's commonplace that they are offered intermittent connectivity at best and required to be smart enough to use heterogeneous network technologies. Finally, modern networks deploy so-called middleboxes both to improve performance and provide protection. However, when doing so, the middleboxes have to impose themselves between the communication end-points, which is against the design principles of the original Internet and a source of complications both for the management of networks and design of application protocols.

In this thesis, we design a clean-slate network architecture that is a better fit with the current use of the Internet. We present a name resolution system based on name-based routing. It matches with the service access and data retrieval oriented usage of the Internet, and takes the network imposed middleboxes properly into account. We then propose modest addressing-related changes to the network layer as a remedy for the denial of service attacks. Finally, we take steps towards a data-oriented communications API that provides better decoupling for applications from the network stack than the original Sockets API does. The improved decoupling both simplifies applications and allows them to be unaffected by evolving network technologies: in this architecture, coping with intermittent connectivity and heterogenous network technologies is a burden of the network stack.

To Aino, for her everlasting love.

Preface

This thesis is a result of my long-time interest in designing and building systems. Over the last twenty years, this interest of mine has been gradually evolving towards bigger systems, starting with elementary programming, on to distributed systems, and then to large-scale distributed systems. Looking backward, it seems inevitable this journey would eventually take me to the study of the largest distributed system, the Internet.

While the seeds for this thesis have certainly been laid a long time ago, the initial steps towards it were taken a three and half years ago, when I asked Martti Mäntylä for an opportunity to take time off from industry to concentrate on research. He promptly arranged the financial support together with my advisor to be, Antti Ylä-Jääski, and provided me the necessary time to gradually depart the industry. Without their generous support, this thesis would not have been possible. Financial support from Emil Aaltonen, Elisa, Nokia, TeliaSonera, and Jenny and Antti Wihuri foundations is also gratefully acknowledged. Once I was funded and began looking for a research topic, it was Pekka Nikander who originally encouraged me to take Internet architecture and its cornerstones, naming and addressing, as a research topic. Therefore, it is him I thank for pointing me towards the eventual direction of my research.

The research itself is very much result of the truly wonderful collaboration with Scott Shenker. Working together with the other bright minds of the DONA team – Ion Stoica, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, and Kye Hyun Kim – has also been a privilege; I'm still amazed how much progress we made during those hectic months of the summer and fall of 2006. In a similar manner, the depth of knowledge of the entire AIP team – together with their firm belief in flat identifiers – has been an unforgettable experience. In the API work, it was the solid DTN expertise of Mike Demmer and Kevin Fall that permitted us to take the next steps with the blurry API vision we had after finishing with DONA. I also thank Pasi Eronen and Mikko Särelä, with whom I have been fortunate to work in many occasions over the years.

Finally, I have a brief message for my pre-examiners, Jussi Kangasharju and Petri Mähönen, and my opponent, Jon Crowcroft: I know you all have heard till boredom about flat identifiers, but one more time!

Palo Alto, June 2008

Contents

1	Introduction	11
1.1	Problem	14
1.2	Design Principles	15
1.3	Contributions	16
1.3.1	Name Resolution	17
1.3.2	Network Layer	17
1.3.3	Application Programming Interfaces (APIs)	18
1.4	Author's Contributions	19
1.5	Structure of the Thesis	19
2	Background	21
2.1	Addressing	21
2.2	Naming	24
2.3	Issues	26
2.3.1	Data-orientation	26
2.3.2	Challenged Internet	28
2.3.3	Controlled Access	29
2.3.4	Delivery Models	31
2.4	Context	35
2.4.1	Architecture and Protocol Design Principles	35
2.4.2	Cryptography	36
2.4.3	Routing	38
2.4.4	Scalability of Routing	39
2.4.5	Securing Routing	41
2.4.6	Communication Abstractions	42
2.5	Summary	44
3	Conclusions	45
	References	47
	Publication I	57
	Publication II	71
	Publication III	79
	Publication IV	93

Original Publications

This thesis consists of an introduction followed by four published articles:

1. Teemu Koppinen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. *A Data-Oriented (and Beyond) Network Architecture*, in Proc. of ACM SIGCOMM '07, Kyoto, Japan, August 2007.
2. David Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koppinen, Daekyeong Moon, and Scott Shenker. *Holding the Internet Accountable*, in Proc. of the 6th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-VI), Atlanta, GA, November 2007.
3. Teemu Koppinen, Pasi Eronen, and Mikko Särelä. *Resilient Connections for SSH and TLS*, in Proc. of USENIX '06 Annual Technical Conference, Boston, MA, May–June 2006.
4. Michael Demmer, Kevin Fall, Teemu Koppinen, and Scott Shenker. *Towards a Modern Communications API*, in Proc. of the 6th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-VI), Atlanta, GA, November 2007.

Chapter 1

Introduction

In the 25 years since becoming commercially available, the Internet has grown into a global communication infrastructure connecting a significant part of mankind and become an important part of modern society. Its impressive growth has been fostered by innovative applications, many of which were completely unforeseen by the Internet's inventors. While fully acknowledging ingenuity and creativity of application designers, it is equally impressive how little the core architecture of the Internet has evolved during this time.

The foundation of the Internet is the Internet Protocol (IP). This protocol provides a strikingly simple service for the Internet-connected hosts: it transmits blocks of data, datagrams, from one host to another over interconnected networks [107]. Moreover, it doesn't even guarantee if, or when, the datagrams will be delivered; the provided delivery service is "best-effort" in nature, in that the Internet tries its best to deliver each packet, but makes not guarantees. Instead, it is the responsibility of hosts and their transport protocol implementations to split application data into appropriate blocks and retransmit any lost datagrams as necessary. This simplicity helped the Internet to interconnect a wide variety of networks.

The original Internet addressing scheme was another important aspect of its design. Every host on Internet had a unique IP address, which is numeric and fixed to 32 bits in size. The address consisted of two parts: a network component (identifying the network the host was on) and host component (identifying the host uniquely on that network). Every transmitted datagram contained both a source IP address and destination IP address; *i.e.*, by inspecting a packet one could determine the sending host and the destination host. For the network, this numeric, fixed size, hierarchical IP address format was a key factor in the feasibility of Internet: the format made the packets easy to process and routers, the packet forwarding network elements, only needed to inspect the network component of the address and could remain ignorant about host component. In other words, it was sufficient for routers to exchange information about available routes to different networks.

While this addressing structure was well-suited for routers, identifying a remote host with its numeric IP address was much less convenient for users. A welcomed improvement to the user-experience was the establishment of a directory mapping host names to their corresponding IP addresses. Users on their terminals could then identify hosts using easy-to-remember names and it became the responsibility of the hosts to translate these names into IP addresses before initiating any communication with a remote host. The first implementation of the name directory was implemented in a centralized fashion, and its contents were merely replicated to every host on the Internet, but as the number of connected hosts increased, a more scalable directory implementation quickly became

necessary.

The Domain Name System (DNS) [93] introduced a distributed name directory for Internet. Hosts would now use DNS name resolvers to resolve a host name into an IP address. As with the design of IP itself, scalability was a major concern in the design of DNS, so naming was organized in a hierarchical manner. However, organizational structures – and not the network structure – typically defined the structure of hierarchy. This rendered the names even easier to use since the embedded organizational structure in the names made names easy to remember and understand.

DNS has been a tremendous success and, in many ways, it solved the right problem, in exactly right way, at right time. However, it is worthwhile to note that DNS was auxiliary to the architecture, not an intrinsic part of it. Before DNS (and its predecessor, a centralized name directory) was deployed, the structure of the basic network protocols (IP and TCP) were already set: in essence, transport protocols had been built directly on top of IP and the DNS name resolution became merely an extra step to execute if the plain IP addresses were not provided by the application. Hence, even though a crucial part of the Internet today, the role of the naming in the Internet architecture is more result of happenstance than of thoughtful architecture.

These two fundamental aspects of networking, *addressing* and *naming*, are the very foundations of Internet and together they form the core of the Internet architecture. It is astonishing how little they have changed since their inception and well they have weathered the Internet's rapid evolution. Indeed, the list of changes to them is very minimal over the years; only very pressing issues have caused the Internet engineering community to change them. Below, we describe some of these changes and their causes.

The Internet originally had a single backbone connecting all attached networks: this eventually became the academic backbone, NSFNET, run by NSF. However, as the Internet grew, and its commercial usage grew even faster, in 1995 NSF finally decided it was no longer appropriate for it to fund and maintain the Internet backbone, which had outgrown from its original academic role. In the name of competition, the single backbone network was replaced by multiple commercial backbones. The original Internet routing protocols, which exchanged the vital routing information between the routers, didn't fit the new decentralized backbone structure of Internet NSF wanted to establish. Hence, the Border Gateway Protocol (BGP) was developed and deployed. Unlike the original routing protocols, it allowed the networks to implement autonomous interdomain routing policies, which essentially were rules how to route network traffic in a way that was most economical for the owner of the network.

At the same time, the growth of Internet was causing increasing concerns about the scalability of Internet; assignable IP addresses were running out and routing table growth was accelerating at a worrying rate. To address these concerns, Classless Inter-Domain Routing (CIDR) [58] was successfully deployed. The original IP address format had led to inefficient address allocation and did not support hierarchical address aggregation, which would have limited the routing table growth. CIDR replaced the fixed number of classes with variable sized address classes, enabling both better address aggregation and more

efficient address allocation practices.

While the deployment of CIDR and its extensions to BGP were well controlled and designed, address exhaustion introduced a less controlled change to Internet: Network Address Translation (NAT). Before NAT, every connected host on Internet had a unique IP address. NAT capable devices changed this. NAT capable routers allowed a network of hosts to share a single IP address; the NAT router would translate private host addresses to a public IP address (and vice versa) on the fly for the packets it forwarded. Even though they broke a fundamental design principle of Internet, as NATs modified packets between the source and destination of the packets, NATs solved an important real-world problem and quickly became commonplace.

These changes — BGP, CIDR, and NATs — are the most fundamental architectural changes the Internet has seen since its inception. Each of these arose due to a change in the Internet's usage: moving from monolithic research network to composite commercial network, from small routing tables to rapidly growing routing tables, and from plentiful addresses to scarce addresses. However, there is now another set of changes in the Internet's usage, and these might also require architectural accommodation. We now discuss four of these changes and their implications.

The Internet architecture was developed around host-to-host applications, such as *telnet* and *ftp*, and the vast majority of its current usage is service access and data retrieval. While less host-oriented applications have existed from the very beginning of the Internet, the World Wide Web was the first popular application to move away from the prevailing host-orientation. The World Wide Web simplified the use of Internet and promptly increasing amount of services, packed with content, were created. Peer-to-peer applications lowered the technical barrier for data retrieval further and transformed the sharing of digital content. However, without underestimating the significance of these applications, the evolution of applications has been universal, and indeed, in some sense, all networking nowadays revolves around content. Some have even argued the content-orientation to be the next era of networking [72].

The Internet has its roots in academia. Openness was considered a critical success factor – after all, this was aligned with the academic roots of Internet – and every connected host received equal treatment. Unfortunately, the same openness that helped to boost the innovation of Internet applications now offers little or no protection for connected organizations and individuals; it is no coincidence that criminals have found ample opportunities on Internet. Today, the Internet is a critical infrastructure and the global economy is increasingly dependent on it. For malicious users it is easy to exploit the many security flaws of major operating systems and their applications to accumulate large botnets, which can launch devastating Denial of Service attacks.

Internet connectivity has become more and more ubiquitous. This has resulted in a variety of Internet connectivity options; both wireless and wired. Unfortunately, a consequence of the improved connectivity has been the varying quality of end-to-end connectivity. Some hosts are well connected (when attached to wired Internet), and others (e.g., mobile hosts) have very intermittent connectivity – and indeed, disruptions

in connectivity are increasingly familiar nuisance for majority of Internet users. In some more extreme networks, one can't expect end-to-end connectivity to ever exist.

Finally, the original Internet was a purely transparent carrier of packets, whereas today such a perfect transparency is rarely the case. Network devices inspecting and modifying packets, or even responding to packets not destined to them are commonplace. The practice unquestionably challenges the early design principles of the Internet, but such middlebox devices are of definite value to users. The middleboxes both improve the performance of popular applications and provide protection for the network and its connected hosts. For instance, it was the World Wide Web which made HTTP caches popular, and these caches were designed to benefit the users. In general, if an application becomes popular, it is only natural that a part of its functionality gets moved into the network to improve the application performance. In a similar manner, while deep packet inspecting firewalls violate the design principles of the Internet, their role in providing practical securing methods for network administrators is hard to challenge.

1.1 Problem

The Internet does its job amazingly well. However, the radical departure from its original usage is a source of discomfort for network administrators, application developers, and for network engineers working on improving the Internet. Therefore, in this thesis we ask: *what would the network stack look like if we were to design it around the contemporary requirements for internetworking?*

Considering the grand-scale of the question, we limit our scope and focus on the design of three essential features of any internetworking-capable communications stack: *a) addressing in the network layer, b) naming, and c) programming abstractions provided for applications.*

- The network layer provides the connectivity over interconnected networks. As discussed above, the packet-switching nature of the original Internet design has proven its value as an interconnection layer, but yet, in the current Internet, the network layer fails to protect its connected hosts. Therefore, we seek to understand how the network layer, and addressing in particular, should change in order to implement the packet forwarding in a more secure manner.
- Whereas the design of the network layer is essential for the feasibility of the network itself, eventually it's the naming that provides applications with a usable abstraction: the naming system translates application friendly names into network addresses.¹ Therefore, the challenge is to come up with a naming system that best meets the needs of modern applications.
- Finally, a network stack must provide applications with an interface to the communication service it implements. The communications Application Programming Interface (API) provides the critical programming abstractions application

¹Transport protocols are certainly essential in providing reliable connectivity on top of the packet forwarding service of a network layer.

developers need to interact with the stack. In the API design, the challenge is in various trade-offs: the API should be convenient for the developers to use, generic enough not to change drastically over the time as technologies evolve, and yet it should be feasible to implement.

We note that the test of any architecture is whether it accommodates both current and expected (and unexpected) usage. Therefore, we note our success in solving the above challenges is not measured in the quantity of novel ideas we can establish for any of the sub-problems, but how well the resulting network architecture fits together with its current and foreseeable usage.

Moreover, we note our focus leaves out important features for any network stack:

- Protocol layers below the network layer are not in our scope since our focus is on the internetworking and not local area networking.
- Transport, session, and application protocols are not addressed here.
- Management aspects of networks.

However, at the same breath, we point that much of the research conducted on the above omitted issues is orthogonal to the design of the network layer, naming, and programming abstractions. Therefore, we expect our efforts to be largely complementary with the efforts focusing on the issues outside of our scope.

1.2 Design Principles

Three high-level design principles have guided us throughout the design process:

Principle #1: *Naming and addressing should support the use of Internet.*

As discussed above, the current role of naming in Internet architecture is not the result of careful architectural crafting; instead, it was an add-on to the core architecture when the most of its critical design (IP and transport protocols) was already set. Moreover, the design of naming was constrained by scalability concerns. In a similar manner, in the design of Internet addressing, scalability was a priority that set the direction of the design process. Here, we set our priorities differently: we aim to design the addressing and naming for the use of Internet. While obviously scalability is a concern even today, we challenge ourselves to seek solutions that are on the edge of being feasible and then count on improving hardware to ensure continued feasibility.

As a result of this principle, in our design names and addresses are flat. In other words, the names and addresses don't contain any structure nor semantics. However, they do have a cryptographic meaning as the names embed a cryptographic hash of a public key. This simple, and strikingly elegant, mechanism of constructing names and addresses will be repeating throughout the thesis, and we'll see how on it becomes the foundation for a network architecture meeting today's needs.

Principle #2: *Favor commodity hardware components.*

This principle largely follows from first principle. History has shown that the commodity hardware components develop at a very favorable rate; they get cheaper, faster and more featureful. While the Internet architecture should not depend on any particular hardware realization, it would be best if it could be implemented with commodity hardware. The closer we are to using commodity hardware components in the design space, the better guarantee there is of steady and favorable development of the necessary components. In many ways, the history of Internet routing is an contradicting example and demonstrate what we want to avoid: CIDR introduced the tough requirement of longest prefix matching, which at least in the beginning, made custom made memory chips, Ternary Content-Addressable Memories (TCAM), a necessity since the longest prefix matching was challenging to implement at high constant rates required by the Internet core.

However, we emphasize the principle is not to suggest there wouldn't be a place for special hardware. On the contrary, positioning functionality to the hardware components is often the most desirable point in the entire design space. For example, trusted hardware would provide a valuable security solution: once the implementation of a functionality is provided in hardware solely, circumventing the functionality becomes much more challenging compared to exploiting of a software implementation.

Principle #3: *Don't build everything into the network architecture, but assume that complementary infrastructures will arise to fill in the necessary functionality.*

We have strived for elegance and simplicity in the design in order to establish a solid foundation for applications built on the resulting network architecture.² Hence, we do not seek an architecture that implements every need, but instead, we assume the omitted aspects can be built on top of the foundations we establish – and yet, that the resulting architecture offers an attractive platform for such missing future aspects.

This principle hits to the essence of a common counter-argument against the use of flat identifiers. While they have attractive properties for naming and addressing (as we'll show), their direct exposure to users is next to impossible; 128 bits (or more) long identifiers are hardly user-friendly. Indeed, they are even less user-friendly than the original 32 bits long IP addresses of Internet, which were already a reason to establish human-friendly naming on top of them. However, we emphasize that by setting our priorities differently, if nothing else, we can demonstrate how different approach to naming and addressing can have simplifying implications to the problems difficult to resolve on the Internet today.

1.3 Contributions

The contributions of this thesis are architectural. Many of the central ideas have existed in the research community for a long time, and we have merely borrowed them. Therefore,

²Indeed, in many ways, we have merely followed C.A.R. Hoare's now famous statement about the relationship of simplicity and reliability: *"The price of reliability is the pursuit of the utmost simplicity."*

it's the synthesis of these ideas into a coherent network architecture that we claim to be our contribution. In the following sections, the main contributions are introduced in more detail.

1.3.1 Name Resolution

The main contribution of this thesis is the design of a clean-slate name resolution system that helps the Internet meet the requirements it faces today. We call the anycast-based name resolution infrastructure the *Data-Oriented (and Beyond) Network Architecture (DONA)*. Its design is discussed in detail in Publication I.

As discussed above, modern usage of the Internet revolves largely around content and services, which is very different from the original host-oriented use of the Internet, for which the addressing and naming of the Internet was originally designed. It is our contribution to demonstrate how to design a naming and name resolution system to provide the necessary support for this data-oriented usage. When it comes to naming, there are three issues an Internet user cares about:

- **Persistence.** Users expect the names of the content and services to remain unchanged over time and over changes in the location of hosting.
- **Availability.** Users expect the named content and services to be available regardless of service and network failures.
- **Authenticity.** Users expect the content to be authentic; in essence, users don't care about the where the content comes from, but that it's authentic (i.e., was generated by the source they expected generated the data).

Moreover, the modern Internet has stakeholders between the end-points that the original design didn't consider since they were not present at that time. In addition to supporting the data-oriented use of Internet, our contribution is to show how to design a name system to take these various new "middle" stakeholders into account.

Finally, we note that a naming system can take pressure off of the network layer below; once the name system provides an anycast primitive, it reduces the need for the network layer to support anycast.

1.3.2 Network Layer

While one could use DONA to replace many of the current network-layer routing functions (and instead have the network layer use source-routes), the resulting design is too extreme. Such a design would make the host responsible for reacting in changing network connectivity: if the path between two end-points changes, the end-points must re-establish their connectivity through the naming system.

While tempting to overload DONA with routing functionality, the design presented here still uses the network layer for basic routing. However, we do not deal with routing in this thesis, but instead focus on the main unsolved problem at the network layer: Denial of Service attacks.

The critical feature IP lacks today is *accountability*; the hosts on Internet are not enforced to be responsible for their actions. At high-level, our observation follows very much the principles of modern societies; while preventing crimes is important, it's even more important to make stakeholders responsible for their crimes. It is our contribution to design a network layer that resolves the security problems of current IP by establishing a firm concept of accountability at the network layer.

We call the resulting design *Accountable Internet Protocol (AIP)* and discuss its design in detail in Publication II. In short, we show that the required changes to IP are in the structure of addresses. Moreover, we show that the required changes are rather modest in nature; we merely replace fixed size longest prefix matching addresses with stacks of flat identifiers having cryptographic meaning. Indeed, in many ways, we take a step backwards in the history of Internet to the era before CIDR. Once the addressing establishes such a strong notion of identity, the network can efficiently provide hosts a trustable mechanism to shut-off unwanted traffic as well as prevent address forgeries.

1.3.3 Application Programming Interfaces (APIs)

Today, an application's main interface to the TCP/IP stack is the *Sockets API*. The sole goal behind this design was to provide an interface to the main transport protocols of that time, TCP and UDP. It was a response to a pressing need: applications needed an interface, and hence, considering long term implications was beyond the scope.

As a result, the Sockets API is inherently tightly coupled to the underlying communications stack and to the properties of its protocols. Unfortunately, this means that protocol changes ripple into applications. The deployment efforts of IPv6 are a sad example of this: simple quadrupling the address length required years for applications to adapt. (Although, admittedly, the deployment rate itself didn't enforce developers to act faster.)

Tight coupling between applications and the communication protocols has another effect. In the presence of intermittent connectivity (e.g., due to popularizing wireless connectivity and device power management) modern applications have to deal with connection problems not prevalent in the wired Internet: connectivity comes and goes, different interfaces are active at different times, and so on. Moreover, it's challenging for legacy applications to support the emerging communications networks where the end-to-end connectivity between hosts is a rare event.

It is our first contribution to exemplify the practical implications of intermittent connectivity conditions for applications and then design the necessary protocol extensions for two major secure session protocols, Secure Shell (SSH) and Transport Layer Security (TLS) protocol. We'll discuss their design and challenges in Publication III.

It's our second contribution to design and take steps towards a modern communications API that enables better decoupling of the communications stack and applications. This decoupling establishes the very features challenging to provide with the Sockets API:

- **Evolution.** Once applications and the stack become decoupled, they may evolve independently. In other words, ideally, with such an API, the applications will

remain unchanged even new network technologies and protocols are deployed below the API.

- **Heterogeneous connectivity.** It seems inevitable that there will be more and more connectivity technologies available for devices. In addition to supporting unexpected future network technologies better, the decoupling can free the applications from the problem of using communications resources efficiently. If the stack is smart enough, the application remains unaware of the various connectivity options in use.

The result of our API design is a communications API based on publish/subscribe primitives. While the research community has considered the publish/subscribe primitives as an attractive programming abstraction for a long time, it's our contribution to propose an initial concrete design using these primitives as the basis of a generic communications API. We discuss the details of the resulting API in Publication IV.

1.4 Author's Contributions

The author's role and contributions were as follows:

- Publication I (12 p.): The author was the architect, team leader, and prototype developer in the research project resulting in *Data-Oriented (and Beyond) Network Architecture (DONA)*.
- Publication II (7 p.): The author was a member of the *Accountable Internet Protocol (AIP)* architecture design team, and was especially responsible for the mechanisms involved in areas of scalability and traffic engineering.
- Publication III (12 p.): The author was the main developer and codesigner of the secure session protocol extensions.
- Publication IV (7 p.): The author shared the architect's role together with another student (Mike Demmer), who was responsible for the Disruption-Tolerant Networking aspects of the API design – while the author was responsible for the design for the other half, the wired Internet.

1.5 Structure of the Thesis

The thesis has the following structure. In the following chapter, we'll go through the essential background of our contributions. The chapter is followed by a conclusion, which after the actual original publications will follow, one publication per a chapter.

Chapter 2

Background

This thesis is about designing a clean-slate network stack for internetworking. Covering the related research here in its full depth and variety would require a book of its own, even with our already limited scope. To reduce it to a manageable length, our review of the necessary background will be limited to a few seminal papers from the past and some recent works that illustrate the issues we address here. These should prepare a reader new to the field of Internet architecture with a high-level overview and a guide to where to start diving in to the literature for more details.

The structure of this chapter is as follows. We begin with the focal point of this thesis: *addressing* and *naming*. Then we discuss the issues the Internet faces today that motivate the changes proposed herein: the trend towards *data-orientation*, communication in *challenged networks*, the need to *control access*, and the difficult history of supporting *delivery models* beyond the traditional unicast model. We conclude this chapter by discussing several issues that are necessary context for the material at hand: the *design principles* of the Internet, *essential cryptographic mechanisms*, *routing*, and basic *communications abstractions*.

2.1 Addressing

In [125], Shoch suggested the following terminology for the addressing, naming, and routing:

- A *name* identifies what you want,
- An *address* identifies where it is, and
- A *route* identifies a way to get there.

These definitions make the concepts of addressing and naming very concrete. In this subsection we focus on addressing then continue to naming in 2.2; we come back to discuss routing in Section 2.4 after considering motivational issues in 2.3.

At a very high-level, there are only two types of addresses for the networks to use:

- Flat addresses that are free from any network topological information. In other words, they merely *identify* the communication end-point, and don't directly provide location information. As a result, a network using flat addresses must know the location of every end-point or broadcast the packet in some fashion.

Perhaps the most widely-spread example of such addresses today are the Ethernet Local Area Networks (LAN) [90]¹.

- Addresses that embed topological information. This form of address determines the exact network location of the communication end-point, and is therefore very useful in making forwarding decisions in routers. Using a hierarchical structure for these topological addresses is the foundation for scalable routing on networks [78]: at the top of the hierarchy of networks, i.e., in the core of the network, the routers can remain unaware of the networks in the bottom of hierarchy and only inspect part of destination address that defines the next hop at their level of hierarchy.

From the network point of view, the topologically-relevant hierarchical addresses are the most convenient form of addressing, since they scale well and easily provide routers with the information they need to forward. Indeed, since the original Internet addressing [107], the addressing has been hierarchical. While in the beginning the level of hierarchy was limited to two and the addresses consisted of a network and host part, the introduction of CIDR [59] generalized the hierarchy to improve scaling.

However, IP addresses have also a role beyond determining the location of an end-point on the Internet; they also serve as end-point identifiers. Hence, current IP addresses serve as identifiers to identify the host and as locators to define the host location in the Internet. Moreover, the host network stack also uses them as transport connection end-point identifiers when the transport layer above the network layer binds its connections to the IP addresses. For exactly this reason, supporting host mobility at the network layer is especially complicated since the end-point identifier (i.e., the address) inevitably changes when a host roams from a network to another.

The site connectivity requirements of the modern Internet demonstrate the contradiction of these dual roles. Today sites increasingly prefer multihoming to improve connection resiliency, but at the same time they prefer to be reachable at a single address (block) and therefore they need to get a provider independent address (block). As a result, hierarchical addressing is not an option since provider-independent addressing does not reflect underlying topology. This desire for provider independent addresses is enforced by administration practicalities as well; IP addresses are often manually configured in network devices and applications (e.g., to avoid dependencies to DNS), site system administrators prefer to have addresses that don't change even provider would change. We'll return to the issue later in Section 2.4.

The dual role of the IP addresses has proven to be problematic and *identifier-locator split* has been proposed by many to separate the identifier and locator roles of the IP addresses in a clean manner.

The first concrete identifier-locator split proposal was 8+8/GSE [100], which emerged in the early stages of IPv6 [36] development in the Internet Engineering Task Force (IETF). IPv6 extended the 32-bit address size to 128 bits, and hence, the scalability of IPv6 routing

¹Ethernet addresses certainly contain a prefix defining the vendor of the network device, i.e., they do have an internal structure.

was seen potentially challenging compared to IPv4 routing. 8+8/GSE proposed to have a stable end-point identifier part in IP addresses, whereas the network part could be changed by routers. In this manner, the network addresses could be assigned in a strictly hierarchical fashion (thus enabling very good scaling properties for IPv6 routing), whereas hosts would remain mostly unaware of them. The edge routers of a multi-homed site would select the most appropriate network part for their hosts.

In the post-mortem of the initial IP mobility support development, IETF produced its next significant proposal for identifier-locator split: Host Identity Protocol (HIP) [94]. In HIP, an (host) identifier layer was added between the transport layer and the IP layer. In this manner, the transport layer could base its operations to stable (cryptographic) identifiers, while the network addresses could change without transport layer seeing the changes. This explicit notation of identities enables seemingly simple security solutions compared to pure IP address based mobility solutions [98].

Recent research proposals (e.g., [21]) go even further from the identifier-locator split and get entirely rid of addresses. In such a network, the routing at the network layer occurs directly on (flat) identifiers (without any topological information for the network to use), and thus, the goal of shortest-path routing is sacrificed for the benefit of the scalability of routing. (Otherwise, the routers would be required to store information about every connected host on the Internet, which clearly is not economical nor feasible technically.) Hence, in essence, the routing on flat identifiers comes with a price of increased path stretch, which is the ratio of the resulting path and the shortest available path over the network.²

While none of the above proposals has been deployed to this date, the role of the IP addresses has changed over the years. The inception of Network Address Translators (NAT) and middleboxes [23] has decreased the role of IP addresses as the communication end-point identifiers. NATs and middleboxes translate IP addresses *in the network* without hosts involvement. In a typical setting, a NAT capable router has the only public IP address of the site, and it translates between hosts' private addresses (behind the NAT) and the public IP address. While hosts can connect to hosts in the Internet, connecting to these hosts from the Internet can't be done without special arrangements. Hence, to re-establish connectivity to the hosts behind NATs, protocols above the network layer use their own identifiers with global meaning. SIP [119] with DNS names is an increasingly common protocol capable of doing so. Indeed, it has been proposed to use DNS names as end-point identifiers [53, 66] instead of IP addresses in all communication.

FARA [27] represents an abstract model for reasoning about identifier-locator split architectures. In many ways, it pushes the split to the extreme: it has no global addresses at all, but instead uses Forwarding Directives (FD) in forwarding packets. FDs are sequences of labels with only local meaning and only together they form a sequence of instructions how to reach point A from point B on network. In such a model, it's a responsibility of a higher layer to construct such sequence of FDs for the network layer. NIRA [150] is an example of a practical instantiation of this abstract model; it encodes the FDs into IPv6

²For an overlay based approach, see e.g., Internet Indirection Infrastructure [131].

source and destination addresses. Plutarch [33] takes the last remaining step away from global addressing and argues that to establish connectivity over heterogenous networks of the future Internet the network interconnections should merely translate between network specific addresses (and protocols).

2.2 Naming

As defined by Shoch in [125], names define what you want. Having such a fundamental purpose in the networking, one could think naming would be a carefully designed part of the Internet architecture. However, Internet naming is not a carefully architected design.

Originally Internet naming was organized around centrally distributed text file, *hosts.txt*, which contained mappings between host names and their corresponding IP address. As the Internet grow and became a network connecting workstations instead of servers, the scalability of this approach was about to reach its limits. It became clear that the Internet needed a distributed name directory system. However, while other distributed directory implementations were already available at that time (e.g. [122]), their scalability was considered too insufficient for the Internet. A new more scalable design was needed. However, at this point of time, much of the design of the TCP/IP suite had already been finished, and the transport protocols had already bound themselves to IP addresses.

The Domain Name System (DNS) [93] is organized as a static distributed tree. The namespace is hierarchical and partitioned into *domains*, which then may have sub-domains. The highest domain in the hierarchy is the global root domain. Top level domains (e.g., .com, .net, and the national domains) are then the sub-domains of this global root domain. Each domain, with the exception of the root domain, has a human-readable name, and therefore, to construct a fully qualified domain name it's sufficient to concatenate the domain names from the leaf to of the tree to the top domain. Since the hierarchy of domains follows organizational structures, even the fully qualified domain names remain relatively easy to remember and use.

DNS name resolution begins from the servers of the global root domain and progresses downwards in the domain hierarchy (each domain having a server) until the full name has been resolved. In other to avoid the domain servers high in the hierarchy from becoming the bottleneck for the entire system, DNS uses caching extensively. While resolving, a domain server doesn't need to be contacted if a fresh enough response from it to a domain name query is available. This combination of the hierarchical organization and extensive caching establishes the attractive scalability properties so critical in the design of DNS.

To this date, DNS remains the primary naming service of Internet. For example, the names used in World Wide Web, URI/URLs [12, 15] contain a DNS name, and so do the email addresses. And indeed, the DNS names have become a visible part of applications used by so many. However, many of the challenges DNS faces stem exactly from its very success:

- The DNS namespace has become contentious. Short, especially easy to remember, names having market value have become more popular (and thus, more valuable)

than the rest of the names. In essence, this results in a trend towards a flatter and flatter namespace, which is very much against the original design principles of DNS.

- Names have become more dynamic. The design of DNS assumed relatively static bindings between host names and their IP addresses. However, nowadays the names are extensively used in applications, which prefer the names to remain persistent over the time; after all, this maximizes the user-friendliness of applications. A practical example is the problem of changing Web URLs: since an URL embeds a DNS name (used as a service name), the binding of a DNS name and an IP address has to change when the content moves from a host to another. As a practical solution, the caching time of corresponding DNS names is often limited to be artificially low. While mostly solving the problem, this is again in contradiction with the original design principles.
- The level of DNS security doesn't match with the critical role of DNS. While DNS has a distributed implementation, the domain information it stores is often not properly distributed. It's typical to host the domain information of individual domains using only a few servers. Hence, by attacking only a few critical servers, the resolution of individual domain names can be severely impaired. Moreover, since the overall operation of DNS depends on the global root name servers, they are equally attractive targets for attacks.³ Finally, the domain information itself is not secured in DNS, i.e., a host resolving a name has today little means to verify the integrity and origin of the name resolution responses. Efforts to deploy security extensions to DNS are still ongoing [5].

Indeed, these aspects have caused the research community to actively seek alternatives to DNS. It has been argued the core Internet architecture should not embed such a non-neutral aspect as human-readable names at all but it should focus on providing persistent names [8, 143, 144]. In essence, this is asking for a depart from the hierarchical naming to flat identifiers. As a result, alternative approaches are needed for the scalability of the naming system since the name hierarchy itself can't be used anymore to establish the scalability properties. Distributed Hash Tables (DHT) have been a promising but yet tricky research direction. DHTs can provide logarithmic number of resolution hops (in relation to the number of nodes). Unfortunately, their implementation and especially management have proven to be non-trivial [117]. Instead of replacing the DNS names with flat names, it has also been proposed to re-build DNS on DHTs [112]. This would tackle especially the problems of the DNS security and the dynamic nature of the names.

Finally, while the Internet naming has been dominated by the model of translating a (hierarchical) name into an address, the increasing popularity of mobile devices has driven the research community to propose radical departures from this name translation paradigm. In dynamic networks consisting of mobile devices, the network can much

³As we will see in Section 2.3, anycast can help to secure the root servers.

better assist hosts in finding their resources and services, if the hosts express their *intent* instead of compressing the intent into a name. In [3], Adjie-Winoto et al. propose a simple language for the task. A more recent proof of the benefits of the approach is presented in [132]. In a similar manner arguing in favor of infrastructureless name resolution, it has been argued that in such networks (perhaps without any infrastructural support) the naming shouldn't be laid out according to the organizational structures but based on social networks users and their devices form. Unmanaged Internet Architecture (UIA) [51] is one concrete proposal taking steps towards this kind of vision.

2.3 Issues

In this section, we discuss the major issues the Internet faces.

2.3.1 Data-orientation

When Tim Berners Lee invented the World Wide Web in the late 1980s, he had no idea the extent of the impact it would have on the Internet, and the world. Today, the Web is the primary mechanism for Internet information retrieval and the Hypertext Transfer Protocol (HTTP) [49], the key protocol of the Web, carries a major portion of the overall Internet traffic. HTTP's original protocol design followed a strictly client-server model, but as the Web became popular additional elements were incorporated into the design.

The increasing traffic volumes of popular web sites combined with consumer availability and performance requirements led to the development and deployment of caching HTTP proxies. HTTP proxies were designed to absorb some of the load born by web servers: instead of sending content requests directly to web servers, web browsers sent their requests to a local HTTP proxy, which then served a cached copy of the content without contacting the web site at all – if it had happened to have a fresh-enough copy of the content. While in the beginning many researchers had the goal of deploying an Internet-wide hierarchy of caches, it soon became obvious the benefits of such caching hierarchies didn't justify their deployment efforts [147]. Hence, HTTP proxies are mainly deployed at the edge networks, where the benefits are clear and deployment easy.

Despite the widespread use of proxies, major web sites continued to experience increasing traffic volumes, and at the same time the published web content got richer in quality. The major web sites continued to seek for mechanisms to improve the loading times of their web pages and to reduce the load on their servers. This opened an opportunity for Content Distribution Networks (CDNs) operated by companies like Akamai.

CDNs are networks of globally distributed HTTP proxies.⁴ Typically the proxies are placed near critical traffic junctions of the Internet. To offload content requests from a web site to the CDN, the web site modifies links on its web pages to point the CDN's domain name instead of its own. In this manner, the DNS servers of the CDN network get queried when a web browser initiates the downloading of the referenced content. Based on the incoming DNS query the CDN then determines which of its HTTP proxies is the

⁴CDNs have extended their protocol support since their early days, though.

closest to the user and should respond to the content request. By then responding to the DNS query with the IP address of the HTTP proxy, the web browser gets directed to this proxy. While the design of CDNs is a set of carefully conceived and integrated tricks, rather than a principled architecture, they remain to be the primary form of assisting performance-challenged web sites. However, since their services are commercial, they also remain an unrealistic option to many non-commercial, but popular sites. As a response, the research community has produced their own, free for use, CDNs (see, e.g., [54]).

In addition to the Web, content-sharing peer-to-peer applications was another new class of applications that arose in the 1990s. The source of their success was their ability to operate without any fixed infrastructure: the hosts running the applications formed the infrastructure by themselves. Without needing any expensive support from CDNs and without being controlled by anyone, these peer-to-peer networks quickly filled with vast amounts of (often illegal) content. Suddenly, distributing even large content files had come within reach of anyone who could download and install such a peer-to-peer application. And for the more established peer-to-peer applications, such as BitTorrent [17], the installation certainly isn't difficult.

Since an increasingly large portion of Internet usage revolves around content distribution, and yet the Internet was designed to connect hosts and not to distribute content, it has been proposed by many in the research community that there is reason to reconsider the design of the Internet:

- In their seminal work, TRIAD [65], Gritter and Cheriton formulate a response to the increasing popularity of HTTP. They suggest embedding much of the HTTP's functionality into the Internet architecture itself. In TRIAD, URLs identify endpoints and routing occurs based on them; IP addresses are merely used as transient routing labels in fetching content after a name-based routing mechanism has routed the content request to the closest copy of the content.
- In a recent research proposal [140], Venkataramani and Towsley argue for a similar architectural role for BitTorrent as TRIAD did for HTTP: they argue that the robustness of the underlying *swarming* mechanisms of the BitTorrent is a reason to consider whether the Internet architecture could be based on swarming.
- In [72], Van Jacobson boldly argues the next era of networking should be content-centric networking. He claims the content domination to be already so prevalent that the packet switching era of the original Internet architecture has come to its end.

All these proposals share a reliance on a *publish/subscribe* communication model: applications fetching content subscribe to content published by other applications. While perhaps new in the context of network research for some, it's worthwhile to note the generality and power of the publish/subscribe model has been well known in systems research, where the decoupling of publishing and subscribing in both time and space has been considered an attractive complexity-reducing property in building distributed

systems [41]. The decoupling allows a subscriber both to subscribe to content before or after its publishing *and* to remain ignorant about its location. (In a similar manner, a publisher knows little about subscribers' interest in its content.)

Most advanced publish/subscribe systems implement content-based routing; subscribers define filters that match the content they are interested in instead of specifying individual publication names. While this enhances the generality and appeal of the publish/subscribe systems, implementing such content-based routing at large scale remains to be a challenging task even when done for an individual application (see, e.g., [4, 24, 105]). Indeed, the challenge of designing, implementing, and deploying a content-based publish/subscribe system at Internet-scale has been cited as a reason to reconsider the paradigm of content-based routing and to split it into more manageable subparts [109].

2.3.2 Challenged Internet

In the computer networking era before the Internet, computers were connected to smaller local area networks, and to reach remote computers beyond the local network they had to establish dial-up connections over the phone network. The communication protocols of that time, such as UUCP (Unix to Unix CoPy), were designed for such intermittent connectivity: a sending system queued the work requests, opened a dial-up connection, sent them to the remote system(s), and later polled the system(s) for results. Email could be sent even over multiple hops – as long as the sender defined its path through the system. When later integrated with the Internet mail system [70], UUCP mail reached networks beyond the fixed Internet. The resulting model was very general as it made few assumptions about the networking technologies and did not require end-to-end connectivity between the sender and receiver of messages.

Once Internet connectivity became widespread, the use of dial-up oriented protocols largely disappeared. However, the Internet is again entering era where not every host can be assumed to have a fixed Internet connection. This development is mainly due to two rather different reasons:

- Network connectivity is continuously expected to reach new places. However, it's evident that providing end-to-end connectivity is an unrealistic assumption in many extreme conditions. For example, in many interplanetary and underwater (submarine) contexts, connectivity is only available in certain periods [42].
- Network connectivity is available on an increasing variety of mobile devices, but often this network connectivity is intermittent. This dynamic environment is challenging for applications to support.

In response, the research community has produced numerous proposals. The IRTF led *Disruption-Tolerant Networking (DTN)* effort [25] has its roots in interplanetary communications. It discards the assumption of end-to-end connectivity and instead builds on the founding ideas of UUCP and Internet mail service: instead of providing a byte-oriented end-to-end connection service for applications, the applications are

expected to transmit and receive self-contained messages. The communication system then delivers the messages, by perhaps storing them for long periods of time on their way from the sender to receiver.

Haggle [132] is another important proposal focusing on the networking challenges of modern mobile devices. It provides applications with a generic content-based publish/subscribe like API that effectively decouples applications from the communications system. The API lets the applications to expose their data and related metadata, and remain ignorant much of the communications details (network interfaces, names, protocols). The few assumptions applications make about networking technology enables the Haggle implementation then to orchestrate the use of communication resources in a way it best can meet the applications' needs, whether the mobile device was connected to the fixed Internet or to an ad-hoc network of mobile devices.

2.3.3 Controlled Access

It was the openness of Internet that largely facilitated its growth. The fact that the network placed no limits on connectivity meant that as innovative applications could be deployed without obstacles. However, the very same openness has now made protecting the network from malicious hosts very difficult. While rudimentary security measures solve most of the problems (e.g., security holes in applications can be patched and end-to-end security protocols can be deployed), the openness has created a class of attacks that are hard to defend against: *Denial of Service (DoS) attacks*. In a DoS attack, a victim host is flooded with packets it didn't intend to receive. If the attacker is equipped with resources to send more packets than the victim can receive, the victim's Internet connectivity is effectively shut down: the flooded packets override any legitimate traffic on the victim's Internet connection.

The early DoS attacks took advantage of the fact that IP addresses are easy to forge. If the attacker repeated the same process on multiple well-connected hosts, it could easily generate enough traffic to block the victim's connection, *and* yet remain unaccountable for its actions. After all, the source address of the sent packets was pointing away from the attacker.

The problem looked seemingly simple to solve: it would be sufficient to ask ISPs to install packet filters on their customer facing routers. The filters would drop any incoming packets with invalid source addresses [48]. In essence, if according to the routing table a source IP address shouldn't be sent to a link, any traffic from that source IP address shouldn't be accepted from that link either. The approach is often called unicast Reverse Path Forwarding (uRPF). This was soon a recommended practice for ISPs [77], but unfortunately, the filters create another manual management task for ISPs. Keeping the filters up-to-date requires extra work, and moreover, the practice isn't feasible anywhere but near the edges of the Internet. In the core, asymmetric routes make use of uRPF based filtering impossible: in the presence of asymmetric routes, traffic from an address may be received from another link than the traffic to the address is sent to. This translates into a requirement for near ubiquitous deployment of the filters, which is a difficult

goal to reach.⁵ In practice, developing and deploying automatic mechanisms for filter management is also challenging [38, 103].

The first proposals against the source address forgery (or spoofing as it's often called) argued the need to establish *traceback* functionality in the Internet (see, e.g., [14, 121, 128]). The traceback functionality would allow the victim hosts to determine where the malicious traffic is coming from, regardless of the spoofed addresses. Most of these proposals are based on the idea of letting routers to store path information in the little-used identification header of the IPv4 header. This required modifying some part of the Internet routers, and therefore, traceback was never deployed.

As the Internet has grown, the rogue communities have created sophisticated tools to exploit security flaws of the connected hosts' operating systems and applications. In the worst case, the criminals have gained access to millions of hosts [40]. They control these hosts using sophisticated decentralized controllers to launch attacks and send distributed spam. When such *botnets* of hijacked computers is used to launch a DoS attack, the victims receive perfectly valid packets from an immense number of seemingly valid hosts. Since the attackers hide themselves behind the compromised hosts, they don't even need to spoof IP addresses anymore. If preventing source IP address spoofing was challenging, protecting from this kind of *Distributed Denial of Service (DDoS)* attacks has proven to be even more challenging.

All the numerous proposals to mitigate the effect of DDoS attacks attempt to empower the hosts to control which hosts are able to send packets for them – which is something very different from the original design principles of the Internet.

In capability based approaches such as Stateless Internet Flow Filter (SIFF) [149] and TVA [151], an Internet connected server authorizes any clients connecting to it by assigning them unforgeable, fine-grained, and relatively short-living *capabilities*. The capabilities are tickets providing non-rate-limited access to the server: if a connecting client host doesn't attach a valid capability to its packets, the network will rate-limit its packets. The privileged traffic can be prioritized over the non-privileged traffic. The rate-limiting capability checking mechanism must be implemented in the routers' data-path. Hence, the capability implementation must be light-weight enough for gigabit speeds.

Instead of requiring changes to the data-path, which is next to impossible achieve, it has been proposed routers' existing filtering mechanisms should be used [6]. Once a victim can remotely control whether packets destined to it should be filtered, the victim can install the required filters by itself without any manual help from the ISPs. The approach doesn't scale to the Internet core since even in the high-end routers number of available filters is rather limited. Therefore, the key is provide victims of DDoS attacks control of the edge routers' filters. However, with this approach, the routers must record the path packets take as only the the victim knows which upstream router to ask to install filters for it. In [124], Shaw pushes the case of remote configurable filters even further and argues the filters should be installed into traffic sending hosts. After all, most users are

⁵It has been shown the deployment of source address filtering remains to be incomplete: according to [16] around 23% of networks use either no filtering at all or their configuration is incomplete.

well-intended, but likely to be unaware their host participates to a DDoS attack. Hence, the users are willing to give such control over their outbound traffic as long as the remote hosts can only filter the traffic destined to them.

However, as argued in [67], a fundamental circle of problems remains unsolved: as long as the address spoofing remains feasible, deploying automatic filtering mechanisms remains challenging. And as long as filters are not deployed, address spoofing remains to be a useful tool for attackers and complicates the deployment of filters. Indeed the difficulty of solving the DDoS problem has led the research community to ask whether the very openness of Internet should be removed. Handley et al. propose moving towards asymmetric connectivity model as a solution: if client addresses were made non-global and hence clients were prevented from communicating directly with each other, as well as, servers were prevented from connecting clients directly, attackers' ability to obtain and control vast number of client hosts would be severely limited.

2.3.4 Delivery Models

The Internet was designed around the unicast delivery model. Therefore, it is well-suited for host-oriented communication, i.e., sending packets from a host to another. However, over the years, the evolving usage of the Internet has asked for additional delivery models. Unfortunately, these models have proven to be difficult to provide in the context of an Internet designed for unicast.

Anycast

In many networking situations, a group of servers implements a network service and clients have little preference over which of the servers they connect to, the only exception being that ideally the connected server would be the closest of the servers. If network can support server selection, the load on servers is better balanced and clients receive better service. Moreover, host configuration can remain unchanged even the server configurations and locations change, or if the host roams from one network to another. Anycast provides exactly this functionality.

In [104], Partridge et al. define IP anycast, in which the selection of the closest service is done at the datagram granularity. It becomes the responsibility of the IP routing protocols to set routing tables properly for anycast IP addresses. While ideal for discovering UDP datagram based network layer services, the IP layer anycast isn't a nice fit with TCP based services; after all, nothing guarantees all datagrams of a TCP flow get routed to the same server. Moreover, to render to an IP address as an anycast address, it needs to be individually advertised in the BGP. However, this poses a severe scalability challenge. Hence, the use of IP anycast remains very limited: the primary users of IP anycast remain to be the global DNS root servers, which are moving towards the use of IP anycast to improve both their scalability and resilience against DoS attacks [68].

However, since the anycast delivery model is so appealing for service discovery, alternative approaches to implement anycast have been proposed and even deployed in a limited scale. In [74], Katabi et al. divide anycast IP addresses into categories based on the popularity of the related service. By providing shortest paths only for the most

popular services and using less optimal default routes for less popular services, the IP anycast can remain scalable the routers don't need to waste space in their routing tables for little used services. In [9] Ballani et al. discuss their design (and give a hint of deployment plans) of an approach using proxies to enhance the scalability of the IP anycast. In their model, anycast proxies advertise multiple services as a single anycast IP address block. Hence, the routing table is not polluted with individual service entries, but it becomes a scalability problem of proxies to determine the closest server for an incoming request. If proxies are placed near the critical interconnections point of the Internet, the length of resulting paths is close to the ones provided by pure IP anycast.

Challenges in implementing and using anycast at the IP layer have driven the development of application-layer anycast mechanisms, which remain to be most widely deployed anycast mechanisms today since they both are usable with TCP-based services and avoid the interaction with the network layer routing. These proposals are typically based on modified authoritative DNS servers, which provide their DNS clients with client specific responses, i.e., the resolution result depends on the client defined properties [45] or on the IP address of the client. The latter requires the DNS server to be capable of mapping IP addresses to their geospatial locations (for a design example, see e.g., [55]). CDNs discussed in Section 2.3.1 typically use application-layer anycast to select the closest caching proxy. Finally, we note that overlay networks can also provide anycast based primitive (e.g., [131]); however, these approaches, in addition to being complicated since not being able to rely on IP anycast in establishing proximity of clients and servers, require modifications to both clients and servers.

Multicast

Multicast is a primitive with a wide range of applications in distributed systems; it provides an efficient mechanism for delivering datagrams to a group of hosts. However, while very useful, interdomain IP multicast has a difficult history. Obstacles for its wide-scale deployment have ranged from the scalability challenges to questioning its economical feasibility.

Until Deering and Cheriton presented the IP multicast service model and their extensions to distance-vector and link-state routing protocols [35], multicast was believed to have significance only in local area networks. Their ideas resulted in Distance Vector Multicast Routing Protocol (DVMRP) [142], which has its foundations in *reverse-path flooding*. In DVMRP, multicast trees are source-based: routers of a single administrative domain maintain a shortest-path delivery tree per a multicast source for every group. DVMRP capable routers run the unicast distance-vector protocol to construct to prune broadcast delivery trees into multicast trees as follows.⁶ The first packet sent by a source is flooded to every link not having a unicast route back to the source. Routers not having members for the specific unicast group then report back their preference not to receive anymore packets for the group. However, to reach potential new multicast group members,

⁶Protocol Independent Multicast Dense-Mode (PIM-DM) [2] is similar to DVMRP with minor modifications; it uses the default unicast routing protocol instead of implementing its own.

DVMRP has to occasionally re-flood packets to not interested routers. This together with the per source state in the routers and reliance to distance-vector routing protocol results in poor scalability.

In shared-tree multicast routing protocols – Core-Based Trees [11] and Protocol Independent Multicast - Sparse Mode (PIM-SM) [46] – the focus is on scalability. In them, sources share a tree rooted at a Rendezvous Point (RP) router (of an administrative domain). Routers explicitly join to a tree by sending a join request towards the unicast path of an RP, instead flooding multicast packets and receiving routers only then asking for being pruned from the tree. This fits nicely to applications where the multicast trees are sparse, i.e., the majority of routers are not a member of a group. However, shared-tree protocols are not as robust as source-based ones due to their dependence on the RPs, and the discovery of an RP is challenging to implement at interdomain level. Multicast Service Discovery Protocol (MSDP) [47] and its BGP related extensions [13] enable the discovery of RPs and group membership management at interdomain level.

The ongoing difficulties to deploy interdomain IP multicast led Holbrook and Cheriton to question the need to support multiple sources in IP multicast [69], and in the resulting Protocol Independent Multicast - SSM (a part of [46]), multicast groups are source-specific. While this limits the supported use cases, the model is much simpler. As the group is identified based on the source address, maintaining the multicast tree using Reverse Path Forwarding (RPF) [34] becomes easy. In a similar manner, the ISPs know exactly which source is responsible (and to be charged, perhaps) for the traffic, and moreover, the access control to the tree is a problem of the source. In a more recent proposal, Free-Riding Multicast [114], inspired by hardware and bandwidth development trends, the approach is taken even further: routers are relieved from the management of multicast delivery trees and the responsibility to form the tree becomes a problem of a router near the source. In the proposal, the source router receives the group membership information over BGP and then computes the multicast tree, and sends a multicasted packet to its every link leading to group members. In every packet the source router then encodes the delivery tree. Routers receiving these packets then merely decode the tree information and using it determine to which links they should be sending the packet next. In essence, the complexity of maintaining interdomain multicast trees is traded to increased bandwidth (since encoding is probabilistic, routers may forward packets to links not having members), and increased consumption of CPU resources (since routers need to decode every multicast packet they receive).

Finally, we note that the overlay networks have been successfully used to implement and deploy multicast networks. While avoiding many of the above deployment challenges, these proposals do face the same challenge the shared-tree multicast protocols face; in the end, discovering a multicast tree (i.e., locating the closest RP) translates into a requirement of implementing the anycast primitive. For an example of a such design, see e.g., [97].

Mobility

Internet architecture was not designed for mobile hosts: since IP addresses embed topological information, they change by necessity when a host roams from an IP network

to another. As a result, a mobile host can't be reached at a single IP address and transport layer connections break when migrating from a network to another. The research community has produced numerous proposals to the problem, but in central role to most of them is the concept of *indirection*: a mobile host is provided with an address that remains unchanged and network infrastructure takes of delivering packets sent to that point of contact further to the mobile host.

Teraoka et al. in [136] were among on the first ones to propose implementing indirection using a “shim layer” existing directly on top of IP. The shim layer operates on virtual IP addresses which remain unchanged regardless of the host's movement. Later the shim layer found its way into Mobile IPv4 [96]. This keeps mobile hosts reachable at a single IP address by providing them with a home agent (which the mobile host keeps informed its current location). The home agent then relays any packets received to the mobile host's home address to its current address. If both end-points support IP mobility extensions, they can avoid routing traffic via home agent in normal conditions (however, if they lose connection with each other, they would use home agent to re-synchronize themselves about each others' locations). However, since IP addresses are so easy to spoof, lacking strong notion of end-point identity, securing these route optimizations is non-trivial (see e.g., the background of Mobile IPv6 security solutions [98]).

Host Identity Protocol (HIP) [94] is a proposal responding to these practical engineering challenges. It introduces a flat, cryptographic end-point identifiers, which exist (again) at the shim-layer just above IP: the cryptographic end-point identifiers establish strong notion of end-point identity and the required security solutions are easier to design. The cryptographic identifiers introduce a new set of problems, however: DNS doesn't help in resolving a flat, host identifier to an IP address of the host nor to an address of its home agent. DHT-based overlays can provide the required functionality and the Internet Indirection Infrastructure [131] demonstrates how elegantly the problem of mobility can be solved together with placing flat identifiers on top of IP *and* capability to route on flat identifiers.

Mobility can be implemented above the network layer. Since changing IP addresses mostly affect the operations of TCP, extensions to implement support for mobility in TCP have been proposed by many (e.g., [60, 71]). However, they introduce a new problem: changing TCP implementations required changes on both hosts, unless peculiar measures are used to determine whether a remote end-point supports the necessary TCP extensions or not [153]. In [129], Snoeren pushes the solution even higher in the stack and argues the most natural place for solving the mobility problem is at the session layer. Since neither the host stack nor the network required modifications, the approach is practical. However, the simplicity comes with a performance cost: for many, solving the mobility problem lower in the stack is mostly due to the need to support relatively fast hand-overs, and the need to re-establish transport and session protocol sessions in a handover largely nullifies that goal. For the rest of the users, session based mobility provides a clean solution, since for them, mobility problem isn't about fast handovers but about intermittent connectivity due to laptop power management and relatively infrequent migrations from a network to

another. In the end, there are no large scale deployment of any IP-based mobility protocol and most of the deployed solutions are based on extending the application layer protocols to support mobility (as e.g., is done for SIP in [123]), or on updating DNS with the current IP address and assuming the addresses changes are rare (which works for e.g., a host with a dynamically assigned IP address).

Finally, we note that while we discuss host mobility above, mobility is more general concept. For instance, increasingly popular host virtualization introduces application mobility [154] and a special form of host mobility due to virtual machine migration [26].

2.4 Context

In this section, we discuss several issues that provide essential context for this thesis.

2.4.1 Architecture and Protocol Design Principles

The architecture of the Internet is based on a number of principles. In the following, we briefly recap the key principles essential for the research presented in this thesis.

Modularization is an essential software development technique: it divides a large system into more manageable subcomponents. In the TCP/IP protocol stack, protocol layering was the chosen method of modularization. Each protocol layer provides functionality built on top of the layer below, the lowest being the layer interfacing with hardware.⁷ Layers both provide an interface for the layer above as well as for their peers over the network. Specific protocols implement these layers, and to remain interoperable with other hosts, they all need to follow the protocol specifications.

The network layer, or IP layer as it's often called, has a special role in the TCP/IP protocol stack: it provides the base for interoperation over various network technologies. IP imposes few limitations on the protocol layers below *or* above. Hosts may implement various transport and applications protocols and remain interoperable as long as both peers implement the same protocols. Where as below the IP, networks may use various local area network technologies as long as the protocols are translated at the network interconnections. To emphasize the important role that such a layer plays in enabling interoperability over a wide range of heterogenous systems, IP has been called a “spanning layer” [30].

In their seminal paper [120], Saltzer, Reed, and Clark present a design principle that has greatly affected the design of the Internet: the *end-to-end argument*. While today it may seem obvious, in its time the principle was a radical departure from the classic communications system design. In the classic design, the communication system provided a feature-complete interface for attached hosts and applications – even the lowest layers of the communication systems had mechanisms nowadays present mostly at higher layers (e.g., mechanisms to improve reliability). According to the end-to-end principle, implementing such mechanisms at low layers in the stack (i.e., in the “middle”) made little sense, since the communication end-points had to implement the same functionality to guarantee the level of service for applications.

⁷Over the time, other forms of modularization has been proposed as well (see, e.g., [19,132]).

The use of the Internet has evolved greatly since the original presentation of the end-to-end argument. Indeed, Blumenthal et al. in [18] observe there's a risk the original design principles will be compromised by new requirements stemming from the modern use of the Internet. As discussed in Chapter 1, it's the combination of involvement of new stakeholders, new applications, and growing popularity of the Internet that together drive this development. However, since the conflicts in the network ultimately stem from conflicts between various stakeholders in general, it has been argued accommodating and designing for these *tussles* is a challenge the Internet will face and have to resolve [28].

The issue of *fate-sharing* is closely related to the end-to-end argument: while the end-to-end argument defines the placement of the functionality, considerations of fate-sharing often indicate which parts of the system should share the state. The principle was nicely captured by Clark: "it is acceptable to lose state information associated with an entity if, at the same time, the entity itself is lost." [29] In other words, hosts shouldn't share state with the network – since network failures can occur – whereas, sharing state with a peer is acceptable. After all, if the host is lost, their connection has little meaning anymore. The shared state can be further divided to *soft-state* and *hard-state*; the first can be re-established (by the network) without interrupting a communication session of end-points, while re-establishing the latter does require re-establishing the communication session.

The primary Internet transport protocol, TCP, provides a byte-oriented stream abstraction for applications, and hence, little of application information is exposed to the stack. In Application Level Framing (ALF) [31], applications expose more information by operating on Application Data Units (ADU). Once the application message boundaries are visible for the stack, their protocol implementations can optimize the performance more easily. For example, if a part (e.g., a packet) of an ADU can't be recovered due to some reason, it's likely that the rest of the ADU is useless for the application as well. Moreover, once the messages become self-contained, their processing order becomes less important for the protocol stack. ALF has had a strong impact on the design of modern end-to-end protocols. For example, SCTP [130] implements its principles, and in a similar manner, the message bundles of DTN [42] can be considered an instantiation its principles.

Protocol stack implementation performance became a concern once the multimedia applications began to emerge in the late 1980s. In Integrated Layer Processing (ILP) [31], the goal was to integrate the processing of an ADU at all protocol layers into a single computational loop. It would maximize the locality of data references [1], and therefore, would optimize the performance of protocol stack implementations. However, in the end, modest performance improvements didn't justify the complexity and the fragility of the required protocol implementations, even when implementations were machine generated [20]. Moreover, the performance bottleneck was often elsewhere, in the application itself or in the network.

2.4.2 Cryptography

In public-key cryptography, or asymmetric cryptography as it's also called, a key to encrypt data is public while a key to decrypt data is private. This is very different from the classic

symmetric cryptography where the keys to encrypt and decrypt data are the same: here, the security of encrypted data depends on a single key – if the key is leaked, the encrypted data becomes public. Therefore, public-key cryptography is more suitable for communications purposes; a public key can be widely distributed (for potential communication peers), while the private key doesn't need to be distributed to anyone but its owner. Public-key cryptography also facilitates digital signatures. For signatures, instead of using a private key to decrypt, it is used to sign data and the corresponding public key is used to verify the resulting signature. RSA [118] was one of the first public-key cryptosystems and remains to be widely used, whereas elliptic curve public key cryptosystems [89] are more recent, and require fewer computational resources [101].

To trust a public key belongs to an intended communication peer, public-key cryptography requires additional mechanisms. Without these mechanisms, an attacker could replace a public key with its own without being detected and the data would be encrypted for the attacker, not for its intended receiver. *Public-Key Infrastructures (PKI)* provide a mechanism to establish trust in a public key: a trusted-third party, a Certification Authority (CA), signs a digital certificate binding the communication peer's identity and public key. If a host has obtained the public key of the CA from a trusted source, a potential communication peer can hand it a certificate (with both peer's public key and identity) issued by the CA and the host can verify the CA's signature in the certificate. Hence, the host can verify the public key belongs to a host it intended to contact. While PKIs are widely deployed, their dependency to a single trust anchor (i.e., CA), is both a scalability challenge and in mismatch with many trust models in practice. Simple PKI (SPKI) [39] is a response to these challenges and can be seen as a generalization of the PKI: in SPKI, any principal can issue a certificate to *delegate* authorizations from itself to another principal, which may delegate them further.

While public-key cryptography can be used to encrypt and decrypt data, it's more efficient to use symmetric cryptography for actual transmission security and to use the public-key cryptography only to execute a *key exchange protocol*, which generates a symmetric key for the symmetric cryptography based protocol. End-to-end security protocols, such as IPsec [75] (at IP layer), Transport Layer Security (TLS) [37] (on top of TCP), and Secure Shell [152] (on top of TCP as well) are protocol suites implementing the key exchange protocol(s) and symmetric cryptography based protocols for data transmission.

Cryptographic hash functions (such as SHA-1 [99]) can establish a useful property of *self-certification*, which has applications in verifying the authenticity of data, as well as, more generally in establishing a secure communication channel. The cryptographic hash functions produce, like any hash function, a fixed size representation of variable sized data, but also guarantee that *a*) finding a message that produces a hash value identical to the one produced for other message, and *b*) finding any two messages that produce identical hash value, is computationally hard. These properties alone are convenient in verifying the authenticity of read-only data: it's enough to provide an application with the data together with its hash value. Moreover, since the hash is unique with high probability, applications

can use it as an identifier for the data (as is done e.g., for filesystem blocks in [84]). If the application data is mutable, the cryptographic hash functions can be applied to a public key of the data owner; then the verified public key can be used to verify the digital signature produced by the owner or to establish a secure communication channel to the owner. This ability to encode a secure representation of a public key into fixed size string is convenient for many applications. For example, in the Secure Filesystem (SFS) [88], filenames embed a cryptographic hash of a public key. In a similar manner, the hash of a public key can be embedded into the lower part of IPv6 address [7].

2.4.3 Routing

The core infrastructure of the Internet is composed of routers. It's the task of the routers to *forward* IP packets from senders to receivers through a series of routers. To *route* packets through the Internet towards their receivers, routers need to populate their routing tables with information how to reach different networks.⁸ Relying on manual configuration of the routing tables is infeasible, since these tables are constantly changing, for a variety of reasons: networks establish new interconnections, customers change their providers, new networks join the Internet, and links and routers experience transient failures, etc.

There routing tables are managed by Internet routing protocols. Using a routing protocol, a router exchanges network reachability information with other routers, thereby learning how to reach the different networks of the Internet. There are three major groups of deployed Internet routing protocols, which we now briefly discuss.

Link-state routing protocols. In link-state routing protocols like OSPF [95], routers share link status information and every router independently computes the best path to every destination network based on the network link graph it has learned. However, since each change in the status of an individual link propagates throughout the entire network, the scalability of this approach is limited.⁹ The main domain of applicability remains in the intra-domain routing, i.e., within a single network. (Although even then their practical administration can be non-trivial [52].)

Distance-vector routing protocols. In distance-vector routing protocols (such as RIP [87]), routers inform their adjacent routers about distances, in router hops, to networks they know about. If a receiving router learns a new shorter route to any network, it updates its routing tables and informs subsequently its connected routers about this better route. While distance-vector routing protocols are simple, they suffer from poor convergence and scalability properties.

Path-vector routing protocols. Interdomain routing on the modern Internet is based on BGP [116], which is a path-vector routing protocol. To improve scalability, BGP is only run between Autonomous Systems (AS) – which are collections of networks under the same administrative domain – and not between every router. In BGP, routers share their network connectivity information, but instead of expressing the reachability to

⁸In modern routers, routing tables are not used to forward packets but to populate forwarding tables of the network interface cards, which are responsible for high-speed packet forwarding.

⁹However, as shown in [83] if slightly increased stretch is acceptable, certain updates can be suppressed.

each destination as a single numeric distance, they exchange the entire path.¹⁰ This is the key to *policy routing*, which allows network administrators to choose routes based on the entire path, not just on the shortest distance. For an introduction to the current interdomain policy routing practices, see e.g., [22]. In this manner, ISPs can implement *traffic engineering*, i.e., they can optimize the use of their network resources by moving traffic from too loaded links to less loaded links. While its exactly this flexibility that is behind the success of BGP, its also exactly the source of BGP's many challenges.

Interdomain routing policies transform the routing system into an arena of battles between ISPs and can result in routing instabilities, which can lead to packet loss, increased network latency and time to converge [80]. Indeed, only long after the deployment of BGP was its practical stability verified: Gao and Rexford proved that BGP is indeed guaranteed to converge in the presence of realistic interdomain routing policies [61]. (Moreover, around the same time Griffin et al. formulated the exact problem BGP solves: unlike distance-vector and link-state routing protocols, BGP isn't a distributed algorithm for solving the shortest-path problem but stable-paths problem [64].) Even after intensive research, BGP remains difficult to analyze: its distributed nature allows only indirect means of discovering information [126]. In addition to this fundamental property of BGP, it's also worthwhile to mention that its proper configuration remain a significant challenge since to achieve high-level traffic engineering goals typically only indirect means exist.

Finally, we note that while the Internet routing protocols are relatively stable, the capability of Internet router infrastructure to accommodate new applications and better serve their needs is very limited. Indeed, the protocol implementations are largely hard-coded, and their evolution is relatively slow (mostly from crisis management by IETF). An interesting future option could be declarative routing [85], where routing protocols are defined in terms of a database query language, which could then lead to faster evolution.

2.4.4 Scalability of Routing

The scalability of interdomain routing has been a major concern on Internet, and indeed, the few changes to the Internet core architecture have been motivated by scalability concerns. As discussed earlier, CIDR [59, 115] successfully solved the pressing issues of IP address exhaustion and non-linear BGP routing table growth by introducing a variable-length network prefix in IP addresses. This allowed networks to get an IP address block meeting exactly their needs, which increased allocation efficiency. In a similar manner, ISPs could advertise an aggregate block of addresses instead of individual customer address blocks. CIDR required both software changes to BGP as well as hardware changes to the routers' packet forwarding plane. It introduced the requirement of longest prefix matching: with CIDR, a packet had to be forwarded to a next-hop indicated by a routing table entry having longest match.¹¹

The scalability of the Internet routing has recently become a concern again [57, 91].

¹⁰In BGP, the messages about network reachability updates embed a list of ASes to use to reach the destination.

¹¹To implement longest prefix matching in high-speed router interface cards, both special algorithms (see e.g., [81, 139]) and components are required (e.g., TCAMs).

However, unlike at the time of CIDR's deployment, this time the reasons behind the concern are more diverse:

- **Multihoming.** The increasing importance of IP has made site-multihoming increasingly popular approach to improve resilience of the network connectivity. Unfortunately, it requires use of so-called Provider Independent (PI) addresses, which largely nullify the aggregation CIDR established: once customer's address block is not anymore a part of its provider's own address block, the provider can't advertise the block as a part of its own address block.
- **Traffic engineering practices.** For example, it is a common practice to split network prefixes and advertise them separately with differing AS paths in order to obtain more fine-grained control over the traffic entering the ISP network.
- **Non-aggregatable address allocations.** Due to historical reasons, certain address blocks are not aggregatable.
- **Business events.** IP addressing is not a nice fit with company mergers and acquisitions, and therefore, companies end up having multiple address blocks.

The CIDR's increasing ineffectiveness has introduced a class of proposals attempting to re-establish the aggregation by building tunnels over the core of the Internet (e.g., [43,141,155]). To avoid changes to hosts, the edge routers (or routers near the edge) encapsulate the packets and transmit the packets over the core using highly aggregated addresses. While this is merely transforming a BGP problem into another, the new problem is expected to be somewhat easier to solve: it is the problem of distributing relatively stable mappings between end-site IP addresses and their aggregates.

The concern over the routing table growth has sparked theoretical interest on routing algorithms as well. Namely, the research field of *compact routing* studies the limits of scalability of routing in graphs in general and it has been recently tried to adapt to the problem of Internet routing [79]. While compact routing can provide even sub-linear routing table (relative to size of the network), with modest path stretch (after all, compact routing is a departure from the shortest path routing), the required algorithms only work well on rather static network topologies. Moreover, they don't typically support flat addressing.

Finally, in addition to the routing table growth, the scalability of the BGP in terms of convergence and stability remains a concern. As discussed above, these challenges stem from the distributed nature of the BGP and the flexibility it provides for ISPs. As a solution, using a path-vector routing protocol (enabling policy routing) only in the very core of the Internet, among Tier-1 operators¹², and a link-state protocol (providing good converge and stability properties) in the ISP hierarchies below them has been proposed [133].

¹²Unlike other ISPs, Tier-1 operators don't have many route changes in their connectivity with each other.

2.4.5 Securing Routing

Since interdomain routing is so vital for the Internet, its security is a great concern. While the distributed nature of BGP provides a reasonable foundation for its operational goals, the distribution is also the root cause for the security challenges it faces:

First, local BGP configuration errors may have global significance. Indeed, as the incidents in the past have shown; a simple error may result in national or even global Internet shutdown requiring manual intervention from ISPs. For example, in two well-known major incidents [32,127], an ISP falsely announced prefixes they didn't own. (In the 1997 case an ISP announced C class subnet per *every* available IP prefix with devastating consequences for the routers of the Internet.) These simple mistakes led to prefixes that were unreachable from major portions of the entire Internet. While major incidents are relatively rare, minor misconfigurations are common and continue to plague the BGP [86].

Second, while defending against such non-intentional errors is challenging, it is even harder to defend against intentional attempts to hijack (i.e., to receive traffic destined to a prefix) or intercept (i.e., to become a relay for the traffic destined to a prefix) prefixes.¹³ While ISPs can implement safety measures to filter out suspicious prefix announcements, in general it is impossible to verify *a*) a route announcement originated from the actual prefix owner or *b*) a router relaying the announcement actually has a path towards the prefix. Prefix hijacking has long been known to be trivially easy, but it has been recently shown that prefix interception is also possible for a significant number of prefixes [10].

There is a class of more local security problems that are easier to resolve and solutions to them have been already deployed to certain extent. For instance, the security of BGP connections can be protected by end-to-end security protocols. We don't discuss these issues here.

To address the fundamental security challenges of BGP, there are practical approaches available. For instance, it is common practice to exercise careful versioning for router configurations [113]. Moreover, there are tools that can analyze the BGP configuration to certain extent *before* it's deployed (e.g., [44]), as well as, tools for detecting suspicious route advertisements, either with the help of centralized repositories (e.g., [63]) or by analyzing the updates and their impact locally before applying them (e.g., [73]). However, these practices and tools don't address the fundamental source of the problem: there's no clear concept of prefix ownership at the BGP level.

Secure BGP [76] was one of the first complete proposals offering a remedy for the BGP security vulnerabilities. It addresses the prefix ownership problem by introducing a PKI that is rooted at Internet Corporation for Assigned Names and Numbers (ICANN), which among other responsibilities is the root authority of the Internet addresses. ICANN then authorizes regional address space allocation authorities to allocate their address spaces as they best see, by issuing certificates to them. The regional authorities further authorize ISPs to assign addresses for their blocks by issuing certificates to ISPs (and so

¹³Prefix hijacking has implications beyond connectivity blocking and traffic interception, e.g., it facilitates sending of untraceable spam [110].

on). As a result, the digital signatures by address block owners in the route advertisements can now be verified by any BGP router as long as it trusts to the root certificate of ICANN.

Secure Origin BGP (soBGP) [146] avoids the deployment of global ICANN rooted PKI and instead assumes the PKI roots are more *ad-hoc* in nature: in soBGP, a small number of trusted third parties, such as Tier-1 ISPs and commercial CAs, certify the bindings between address blocks and their owners. The owners of these address blocks then certify owners of the smaller address blocks.

Finally, we note that it has been argued the secure routing proposals have a wrong focus [145]: instead of focusing on providing a single secure path, the network should focus on guaranteeing availability between end-points by providing end-points with a number of paths to select from. Then if a host determines a path is not leading towards the host or service it intended to connect or that the path has insufficient quality, it could switch to another. Once such mechanism is provided, the standard end-to-end security measures can assist hosts in detecting when there's a reason to change a path.

2.4.6 Communication Abstractions

A network without applications is rather meaningless, and hence, together as operating systems added support for IP they added Application Programming Interfaces (APIs) for applications to access the communications services of the network stack. While in the beginning different operating systems had their unique APIs for applications to interface with the network stack, eventually the API of the 4.4BSD Unix, *Sockets API* [82], became a de facto standard. To this date, the Sockets API remains to be the most popular application interface to the TCP/IP stack; all modern Unix systems provide it and it's emulated by many other operating systems, e.g., by Microsoft Windows.

The Sockets API is a generic interprocess communications interface, with few limitations on the network technologies it can support. Its main abstraction is a *socket*, which is essentially a communication end-point. The Sockets API then provides applications with primitives to operate on sockets; applications can open and close them, connect them (e.g., to remote hosts), wait for inbound connections, and send and receive data (bytes and datagrams). Applications must define the properties of the sockets they open, such as the socket type (whether it's a stream socket, datagram socket, etc.), the communication domain (whether socket is a TCP/IP socket or some other socket), and give a name to the socket by binding it to a network address (or to another name relevant for the chosen communication domain). Hence, while the Sockets API can accommodate a wide range of interprocess communication mechanisms, the applications' responsibility to be in control of the communication details results in tight coupling between the network stack and applications.

At the same time, applications expose little of their semantics for the network stack. The stack merely sees operations revolving around byte streams or datagrams, so the stack can't provide applications with performance matching their needs. While ALF [31] argued for the benefits of exposing more application level semantics in form of Application Data Units (ADU), only relatively recent transport protocols and their APIs (e.g., SCTP [130]) have adapted message-orientation and several research projects taken initial steps towards

exposing more application-level semantics. For example, in [111], Raman and McCanne demonstrate how exposing the actual application data *structure* (by means of hierarchical naming of ADUs instead of typical stream-oriented sequential naming) benefits in implementing reliable multicast. In [50], Ford demonstrates how replacing the distinction between datagrams and streams with a single abstraction of hierarchical, structured streams facilitates transport services that match, and even exceed, the performance of the Sockets API based application protocol implementations embedded into the applications today.

The Sockets API leaves application developers with a responsibility to implement all the application (and session) protocols. However, since the needs of applications are often very alike, over the years various application frameworks have arisen. These *middleware* systems come in many forms (see, e.g., [92, 102, 135]); some provide only basic Remote Procedure Call (RPC) services layered on top of sockets, while some provide a developer with a range of high-level services useful in building distributed systems. For example, in addition to RPC, enterprise-oriented middleware systems often include implementations for distributed transactions and database access protocols. However, the most common provided service among RPC is *publish/subscribe*-like messaging service, which has been proven to be extremely useful in integration of heterogeneous (enterprise) systems. (Tibco [137] is a classic example of such a *messaging-oriented middleware system*.) However, unlike with the different implementations of the TCP/IP stack, no single API for these middleware systems has emerged – even though modern programming languages do often include standardized interfaces for various types of middleware services (e.g., Java has a generic messaging interface, [134]). In the research community, such common APIs have been proposed, though (see, e.g., [106] for a common API for the publish/subscribe systems).

In practice, middleware systems are mostly used in the server-side implementations of enterprise applications, and not in the communications between them and their clients. Recent proposals haven't take steps towards simplifying Internet application development, however. Data-Oriented Transfer (DOT) [108, 138] is a practical attempt to decouple data transferring functionality from the applications. In DOT, applications are provided with a data transfer interface that can take the burden of bulk data transfers away from application protocols leaving the application developers merely responsible for the implementation of the “signaling” plane. The interface simplifies application developers, but it also lets transfer mechanisms to evolve and be deployed independently from the applications and their application protocols. In Huggle [132], the communications interface revolves purely around data and the related meta-data; and the burden of communications is entirely shifted from the applications to so-called “communications managers” below the API. This decoupling of applications from the details of the communications is argued to be especially attractive in (future) mobile ad-hoc networks, where the Sockets API would render the communications part of applications complex due to very dynamic and diverse communication conditions.

Finally, we note about the abstractions developed by the parallel computing com-

munity. The various shared-memory approaches, as well as low-latency messaging services, are common in computation clusters, but *tuple-space* systems are an example of alternative, very powerful programming abstraction the community has produced. In Linda spaces [62] (and in its later embodiments, e.g., [56,148]), associative memory storing tuples facilitates distributed computation. Applications are provided with primitives to operate on these tuples, concurrently. Unfortunately, while primitives are very generic and powerful, their implementation results in tight synchronization between the communication peers, and hence, tuple spaces as an abstraction are challenging to provide beyond tightly coupled computation clusters.

2.5 Summary

In this chapter, we reviewed the essential background of this thesis. We began with the history and current state of the core of the Internet architecture: *addressing* and *naming*. We then discussed the challenges the Internet faces today that motivate the changes proposed herein: the trend towards *data-orientation* in the Internet usage, extending the reach of the Internet to *challenged networks*, the need to *control access* in the traditionally open Internet, as well as the difficulties of supporting *delivery models* beyond the traditional unicast model, such as anycast, multicast, and mobility. We concluded the chapter by discussing several issues that are necessary context for the proposed changes: the *design principles* of the Internet, *essential cryptographic mechanisms*, *routing*, and basic *communications abstractions*.

Chapter 3

Conclusions

In this thesis, we have described the foundations for a clean-slate network stack that better accommodating the requirements of current Internet usage than the existing Internet architecture. The resulting naming scheme and name resolution mechanisms serves the current data-oriented usage of the Internet and properly takes the new “middle” stakeholders into account. The proposed form of addressing at the network layer embeds a firm notion of accountability into the foundations of the Internet, and gives users a weapon against denial of service attacks. In addition to naming and addressing, we proposed initial steps towards decoupling applications from details of the communications stack, in the hope of facilitating their independent evolution.

The design of this clean-slate network stack is far from complete. The proposals presented here are starting points, not finished products. In all areas, but especially regarding the communications APIs, we feel that we have merely touched the surface of the problem domain. For instance, distilling the API proposal into an exact, implementable interface description is a non-trivial design exercise, and until that has been done our proposal is merely speculative.

In addition, a wide range of issues, beyond the focus of this thesis, remain unaddressed. For instance, the current architectural approaches to interdomain routing, congestion control, and network management are lacking in one or more important respects, and new approaches, whether incremental or radical, must be found.

As said in the beginning of the thesis, the ultimate test of an architecture is how good fit it is with its current *and* unexpected usage. While we have solved a few of the current issues the Internet faces today, and hence, partially “upgraded” its architecture to match certain known problems, predicting the future usage of the Internet is impossible. After all, the one thing that can be reliably predicted is that the Internet will continue to seed new innovative applications that are impossible to foresee.

This uncertainty of the future Internet usage leaves us with a perplexing question: *what will be the exact location of the spanning layer in the future Internet?* That is, what will be the lowest layer providing interoperability? As demonstrated by us and others, IP, naming, and data-oriented communication abstractions are all candidates; yet, they provide varying levels of capabilities to integrate network technologies – the higher a spanning layer is located in the communications stack, the less restrictions are imposed on the technologies below, and the less freedom provided for the layers above. The future course of the Internet’s evolution cannot be settled in the pages of this thesis, or by its learned commentators. It is the eventual usage of the Internet that defines which of these (or other) possible spanning layers will prevail.

We unconditionally admit that this uncertainty about the future makes it impossible to predict what role the proposals presented here will play in the future Internet. However, we dare to claim that the individual components proposed herein – naming, addressing, and communications abstractions – have the potential to provide important benefits no matter which course the future Internet takes.

References

- [1] Mark B. Abbott and Larry L. Peterson. Increasing Network Throughput by Integrating Protocol Layers. *IEEE/ACM Transactions on Networking*, 1(5):600–610, 1993.
- [2] Andrew Adams, Jonathan Nicholas, and William Siadak. *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*. Internet Engineering Task Force, January 2005. RFC 3973.
- [3] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The Design and Implementation of an Intentional Naming System. In *Proc. 17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC, December 1999.
- [4] Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching Events in a Content-Based Subscription System. In *Proc. 18th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 53–61, Atlanta, GA, May 1999.
- [5] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. *DNS Security Introduction and Requirements*. Internet Engineering Task Force, March 2005. RFC 4033.
- [6] Katerina Argyraki and David R. Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *Proc. USENIX Annual Technical Conference*, Anaheim, CA, April 2005.
- [7] Tuomas Aura. *Cryptographically Generated Addresses (CGA)*. Internet Engineering Task Force, March 2005. RFC 3972.
- [8] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A Layered Naming Architecture for the Internet. In *Proc. of ACM SIGCOMM '04*, pages 343–352, Portland, OR, USA, August 2004.
- [9] Hitesh Ballani and Paul Francis. Towards a Global IP Anycast Service. In *Proc. ACM SIGCOMM*, Philadelphia, PA, August 2005.
- [10] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A Study of Prefix Hijacking and Interception in the Internet. In *Proc. ACM SIGCOMM*, Kyoto, Japan, August 2007.
- [11] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core Based Trees (CBT). In *Proc. ACM SIGCOMM*, pages 85–95, San Francisco, CA, September 1993.
- [12] Gerco Ballintijn, Maarten van Steen, and Andrew S. Tanenbaum. Scalable Human-Friendly Resource Names. *IEEE Internet Computing*, 5(5):20–27, 2001.
- [13] Tony Bates, Yakov Rekhter, Ravi Chandra, and Dave Katz. *Multiprotocol Extensions for BGP-4*. Internet Engineering Task Force, June 2000. RFC 2858.
- [14] Steve Bellovin. *ICMP Traceback Messages, Internet-Draft, draft-bellovin-itrace-00.txt, Work in Progress*, March 2000.
- [15] Tim Berners-Lee, Roy Fielding, and Larry Masinter. RFC 3986: Uniform Resource Identifier (URI): Generic syntax. RFC 3986, IETF, January 2005.

- [16] Robert Beverly and Steven Bauer. The Spoofer Project: Inferring the Extent of Source Address Filtering on the Internet. In *Proc. SRUTI Workshop*, Cambridge, MA, July 2005.
- [17] Bittorrent, June 2008.
<http://www.bittorrent.com>.
- [18] Marjory Blumenthal and David Clark. Rethinking The Design of The Internet: The End-to-End Arguments vs. The Brave New World. *ACM TOIT*, pages 70–109, 2001.
- [19] Robert Braden, Ted Faber, and Mark Handley. From Protocol Stack to Protocol Heap: Role-Based Architecture. *ACM Computer Communications Review*, 33(1):17–22, 2003.
- [20] Torsten Braun and Christophe Diot. Protocol Implementation Using Integrated Layer Processing. In *Proc. ACM SIGCOMM*, Cambridge, MA, September 1995.
- [21] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, Scott Shenker, and Ion Stoica. ROFL: Routing on Flat Labels. In *Proc. ACM SIGCOMM*, Pisa, Italy, August 2006.
- [22] Matthew Caesar and Jennifer Rexford. BGP Routing Policies in ISP Networks. *IEEE Network Magazine*, pages 5–11, November 2005.
- [23] Brian Carpenter and Scott Brim. *Middleboxes: Taxonomy and Issues*. Internet Engineering Task Force, February 2002. RFC 3234.
- [24] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. In *Proc. 19th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 219–227, Portland, OR, July 2000.
- [25] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-Tolerant Networking Architecture. RFC 4838, IETF, April 2007.
- [26] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live Migration of Virtual Machines. In *Proc. 2nd USENIX NSDI*, pages 273–286, Boston, MA, May 2005.
- [27] David Clark, Robert Braden, Aaron Falk, and Venkata Pingali. FARA: Reorganizing the Addressing Architecture. *ACM Computer Communications Review*, 33(4):313–321, 2003.
- [28] David Clark, John Wroclawski, Karen Sollins, and Bob Braden. Tussle in cyberspace: Defining tomorrow’s Internet. In *Proc. ACM SIGCOMM*, pages 347–256, Pittsburgh, PA, August 2002.
- [29] David D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proc. ACM SIGCOMM*, pages 109–114, Stanford, CA, August 1988.
- [30] David D. Clark. Interoperation, Open Interfaces, and Protocol Architecture. *The Unpredictable Certainty, Information Infrastructure Through 2000: White Papers*, pages 133–144, 1997.
- [31] David D. Clark and David L. Tennenhouse. Architectural Consideration for a

- New Generation of Protocols. In *Proc. of ACM SIGCOMM '90*, pages 200–208, Philadelphia, USA, 1990.
- [32] CNET News.com. Router Glitch Cuts Net Access. <http://news.com.com/2100-1033-279235.html>, April 1997.
- [33] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield. Plutarch: An Argument for Network Pluralism. In *Proc. of ACM SIGCOMM FDNA '03*, pages 258–266, Karlsruhe, Germany, August 2003.
- [34] Yogen K. Dalal and Robert M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.
- [35] Stephen E. Deering and David R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, 1990.
- [36] Stephen E. Deering and Robert M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. Internet Engineering Task Force, December 1998. RFC 2460.
- [37] Tim Dierks and Christopher Allen. *The TLS Protocol Version 1.0*. Internet Engineering Task Force, January 1999. RFC 2246.
- [38] Zhenhai Duan, Xin Yuan, and Jaideep Chandrashekar. Constructing Inter-Domain Packet Filters to Control IP Spoofing Based on BGP Updates. In *Proc. IEEE INFOCOM*, Barcelona, Spain, March 2006.
- [39] Carl Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylönen. SPKI Certificate Theory. RFC 2693, IETF, September 1999.
- [40] Brandon Enright. Exposing Stormworm, October 2007. http://noh.ucsd.edu/~bmenrigh/exposing_storm.ppt.
- [41] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.
- [42] Kevin Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proc. ACM SIGCOMM*, pages 27–34, Karlsruhe, Germany, August 2003.
- [43] Dino Farinacci, Vince Fuller, Dave Oran, and Dave Meyer. *Locator/ID Separation Protocol (LISP)*. Internet Engineering Task Force, November 2007. <http://www.ietf.org/internet-drafts/draft-farinacci-lisp-05.txt> Work in progress, expires May 17, 2008.
- [44] Nick Feamster and Hari Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, May 2005.
- [45] Zongming Fei, Samrat Bhattacharjee, Ellen W. Zegura, and Mostafa H. Ammar. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proc. IEEE INFOCOM*, San Francisco, CA, March 1998.
- [46] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas. *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*. Internet Engineering Task Force, August 2006. RFC 4601.
- [47] Bill Fenner and David Meyer. *Multicast Source Discovery Protocol (MSDP)*. Internet

- Engineering Task Force, October 2003. RFC 3618.
- [48] Paul Ferguson and Daniel Senie. *Network Ingress Filtering*. Internet Engineering Task Force, May 2000. BCP 38, RFC 2827.
 - [49] Roy Fielding, Jame Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol: HTTP/1.1*. RFC 2616, IETF, June 1999.
 - [50] Bryan Ford. *Structured Streams: a New Transport Abstraction*. In *Proc. of ACM SIGCOMM '07*, Kyoto, Japan, August 2007.
 - [51] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. *Persistent Personal Names for Globally Connected Mobile Devices*. In *Proc. of OSDI 2006*, pages 233–248, Seattle, WA, USA, November 2006.
 - [52] Bernard Fortz and Mikkel Thorup. *Optimizing OSPF/IS-IS weights in a changing world*. *IEEE JSAC*, 20(4):756–767, May 2002.
 - [53] Paul Francis and Ramakrishna Gummadi. *IPNL: A NAT-extended Internet Architecture*. In *Proc. ACM SIGCOMM*, pages 69–80, San Diego, CA, August 2001.
 - [54] Michael J. Freedman, Eric Freudenthal, and David Mazières. *Democratizing content publication with Coral*. In *Proc. First Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, March 2004.
 - [55] Michael J. Freedman, Karthik Lakshminarayanan, and David Mazières. *OASIS: Anycast for Any Service*. In *Proc. of NSDI '06*, pages 129–142, San Jose, CA, USA, May 2006.
 - [56] Eric Freeman, Susanne Hupfer, and Ken Arnold. *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley Professional, July 1999.
 - [57] Vince Fuller. *Scaling issues with routing+multihoming*, February 2007. Plenary session at APRICOT, the Asia Pacific Regional Internet Conference on Operational Technologies.
 - [58] Vince Fuller, Tony Li, Jessica Yu, and Kannan Varadhan. *Supernetting: an Address Assignment and Aggregation Strategy*. Internet Engineering Task Force, June 1992. RFC 1338.
 - [59] Vince Fuller, Tony Li, Jessica Yu, and Kannan Varadhan. *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*. Internet Engineering Task Force, September 1993. RFC 1519.
 - [60] Daichi Funato, Kinuko Yasuda, and Hideyuki Tokuda. *TCP-R: TCP Mobility Support for Continuous Operation*. In *Proc. of ICNP '97*, pages 229–236, Atlanta, GA, USA, 1997.
 - [61] Lixin Gao and Jennifer Rexford. *Stable Internet Routing without Global Coordination*. *IEEE/ACM Transactions on Networking*, pages 681–692, December 2001.
 - [62] David Gelernter and Nicholas Carriero. *Coordination Languages and their Significance*. *Communications of ACM*, 35(2):97–107, 1992.
 - [63] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick

- McDaniel, and Aviel Rubin. Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing. In *Proc. NDSS*, San Diego, CA, February 2003.
- [64] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Transactions on Networking*, 10(1):232–243, 2002.
- [65] Mark Gritter and David R. Cheriton. TRIAD: A New Next-Generation Internet Architecture. <http://www-dsg.stanford.edu/triad/>, July 2000.
- [66] Saikat Guha and Paul Francis. An End-Middle-End Architecture for Secure Internet. In *Proc. ACM SIGCOMM*, Kyoto, Japan, August 2007.
- [67] Mark Handley and Albert Greenhalgh. Steps Towards a DoS-resistant Internet Architecture. In *Proc. of ACM SIGCOMM FDNA '04*, pages 49–56, Portland, OR, USA, August 2004.
- [68] Ted Hardie. *Distributing Authoritative Name Servers via Shared Unicast Addresses*. Internet Engineering Task Force, April 2002. RFC 3258.
- [69] Hugh W. Holbrook and David R. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. In *Proc. ACM SIGCOMM*, pages 65–78, Cambridge, MA, September 1999.
- [70] Mark R. Horton. *UUCP Mail Interchange Format Standard*. Internet Engineering Task Force, February 1986. RFC976.
- [71] Christian Huitema. Multi-homed TCP. Work in progress (draft-huitema-multi-homed-01), May 1995.
- [72] Van Jacobson. <http://yuba.stanford.edu/cleanslate/jacobson.pdf>, February 2006.
- [73] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Pretty Good BGP: Protecting BGP by Cautiously Selecting Routes. Technical report, University of New Mexico, October 2005. TR-CS-2005-37.
- [74] Dina Katabi and John Wroclawski. A Framework for Scalable Global IP-Anycast (GIA). In *Proc. ACM SIGCOMM*, pages 3–15, Stockholm, Sweden, September 2000.
- [75] Stephen Kent and Randall Atkinson. *Security Architecture for the Internet Protocol*. Internet Engineering Task Force, November 1998. RFC 2401.
- [76] Stephen Kent, Charles Lynn, and Karen Seo. Secure Border Gateway Protocol (S-BGP). *IEEE JSAC*, 18(4):582–592, April 2000.
- [77] Tom Killalea. *Internet Service Provider Security Services and Procedures*. Internet Engineering Task Force, November 2000. RFC 3013.
- [78] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.
- [79] Dima Krioukov, kc claffy, Kevin Fall, and Arthur Brady. On Compact Routing for the Internet. *ACM Computer Communications Review*, 37(3):41–52, July 2007.
- [80] Craig Labovitz, G. Robert Malan, and Farnam Jahanian. Internet Routing Instability. *IEEE/ACM Transactions on Networking*, 6(5):515–526, 1998.

- [81] Karthik Lakshminarayanan, Anand Rangarajan, and Srinivasan Venkatachary. Algorithms for Advanced Packet Classification with Ternary CAMs. In *Proc. ACM SIGCOMM*, pages 193–204, Philadelphia, PA, August 2005.
- [82] Sam Leffler, Robert Fabry, William Joy, Phil Lapsley, Steve Miller, and Chris Torek. An Advanced 4.4BSD Interprocess Communication Tutorial.
- [83] Kirill Levchenko, Geoffrey M. Voelker, Ramamohan Paturi, and Stefan Savage. XL: An Efficient Network Routing Algorithm. In *Proc. of ACM SIGCOMM '08*, pages 15–26, Seattle, WA, USA, August 2008.
- [84] Jinyuan Li, Maxwell Krohn, David Mazières, and Dennis Shasha. Secure Untrusted Data Repository (SUNDR). In *Proc. 6th USENIX OSDI*, pages 121–136, San Francisco, CA, December 2004.
- [85] Boon Thau Loo, Joseph M. Hellerstein, Ion Stoica, and Raghu Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *Proc. of ACM SIGCOMM '05*, Philadelphia, PA, 2005.
- [86] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP misconfiguration. In *Proc. ACM SIGCOMM*, pages 3–17, Pittsburgh, PA, August 2002.
- [87] Gary Malkin. *RIP Version 2*. Internet Engineering Task Force, November 1998. RFC 2453.
- [88] David Mazières, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. Separating Key Management from File System Security. In *Proc. of SOSP '99*, pages 124–139, Charleston, SC, USA, December 1999.
- [89] Alfred J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, Boston, MA, 1993.
- [90] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed Packet Switching for Local Computer Networks. *Communications of the ACM*, 19(7):395–404, 1976.
- [91] David Meyer, Lixia Zhang, and Kevin Fall. *Report from the IAB Workshop on Routing and Addressing*. Internet Engineering Task Force, September 2007. RFC 4984.
- [92] Microsoft. .NET Framework, June 2008.
<http://www.microsoft.com/net/>.
- [93] Paul V. Mockapetris and Kevin J. Dunlap. Development of the Domain Name System. In *Proc. ACM SIGCOMM*, pages 123–133, Vancouver, British Columbia, Canada, September 1998. also in *Computer Communication Review* 18 (4), Aug. 1988.
- [94] Robert Moskowitz and Pekka Nikander. *Host Identity Protocol (HIP) Architecture*. Internet Engineering Task Force, May 2006. RFC 4423.
- [95] John Moy. *OSPF Version 2*, March 1994. RFC 1583.
- [96] Andrew Myles, David B. Johnson, and Charles Perkins. A Mobile Host Protocol Supporting Route Optimization and Authentication. *IEEE JSAC*, 13(5), June 1995.
- [97] Animesh Nandi, Aditya Ganjam, Peter Druschel, T.S. Eugene Ng, Ion Stoica, Hui Zhang, and Bobby Bhattacharjee. SAAR: A Shared Control Plane for Overlay

- Multicast. In *Proc. of NSDI '07*, pages 57–72, Cambridge, MA, USA, April 2007.
- [98] Pekka Nikander, Jari Arkko, Tuomas Aura, Gabriel Montenegro, and Erik Nordmark. *Mobile IP Version 6 Route Optimization Security Design Background*. Internet Engineering Task Force, December 2005. RFC 4225.
- [99] NIST. Secure Hash Standard (SHS). In *FIPS Publication 180-1*, 1995.
- [100] Masataka Ohta. *8+8 Addressing for IPv6 End to End Multihoming*, January 2004. draft-ohta-multi6-8plus8-00 (Expired IETF Draft).
- [101] Tatsuaki Okamoto and Jacques Stern. Almost Uniform Density of Power Residues and the Provable Security of ESIGN. In *ASIACRYPT*, volume 2894 of *LNCS*, pages 287–301, December 2003.
- [102] OMG. Common Object Request Broker Architecture: Core Specification, March 2004.
<http://www.omg.org/docs/formal/04-03-12.pdf>.
- [103] Kihong Park and Heejo Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In *Proc. ACM SIGCOMM*, San Diego, CA, August 2001.
- [104] Craig Partridge, Trevor Mendez, and Walter Milliken. *Host Anycasting Service*. Internet Engineering Task Force, November 1993. RFC 1546.
- [105] Peter Pietzuch and Jean Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *International Workshop on Distributed Event-Based Systems (DEBS '02)*, Vienna, Austria, July 2002.
- [106] Peter Pietzuch, David Eyers, Samuel Kounev, and Brian Shand. Towards a Common API for Publish/subscribe. In *International Workshop on Distributed Event-Based Systems (DEBS) '07*, pages 152–157, 2007.
- [107] Jon B. Postel. *Internet Protocol*. Internet Engineering Task Force, Information Sciences Institute, Marina del Rey, CA, September 1981. RFC 791.
- [108] Himabindu Pucha, David G. Andersen, and Michael Kaminsky. Exploiting Similarity for Multi-Source Downloads using File Handprints. In *Proc. 4th USENIX NSDI*, Cambridge, MA, April 2007.
- [109] Costin Raiciu, David S. Rosenblum, and Mark Handley. Revisiting Content-based Publish/Subscribe. In *Proc. of the 5th Intl. Workshop on Distributed Event-Based Systems (DEBS)*, Lisbon, Portugal, July 2006.
- [110] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM*, Pisa, Italy, August 2006.
- [111] Suchitra Raman and Steven McCanne. Scalable Data Naming for Application Level Framing in Reliable Multicast. In *Proc. of the Sixth ACM International Conference on Multimedia*, Bristol, England, September 1998.
- [112] Venugopalan Ramasubramanian and Emin Gün Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. In *Proc. ACM SIGCOMM*, pages 331–342, Portland, OR, August 2004.
- [113] Really Awesome New Cisco Conflg Differ (RANCID). <http://www.shrubbery.net/rancid/>, 2004.

- [114] Sylvia Ratnasamy, Andrey Ermolinskiy, and Scott Shenker. Revisiting IP Multicast. In *Proc. ACM SIGCOMM*, pages 15–26, Pisa, Italy, August 2006.
- [115] Yakov Rekhter and Tony Li. *An Architecture for IP Address Allocation with CIDR*. Internet Engineering Task Force, September 1993. RFC 1518.
- [116] Yakov Rekhter, Tony Li, and Susan Hares. *A Border Gateway Protocol 4 (BGP-4)*. Internet Engineering Task Force, January 2006. RFC 4271.
- [117] Sean C. Rhea, Brighten Godfrey, Brad Karp, John Kubiawicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. OpenDHT: A Public DHT Service and Its Uses. In *Proc. ACM SIGCOMM*, Philadelphia, PA, August 2005.
- [118] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
- [119] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, and Eve Schooler. *SIP: Session Initiation Protocol*. Internet Engineering Task Force, June 2002. RFC 3261.
- [120] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end Arguments in System Design. *ACM Transactions on Computer Systems*, 2:277–288, November 1984.
- [121] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Network Support for IP Traceback. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.
- [122] Michael D. Schroeder, Andrew D. Birrell, and Roger M. Needham. Experience with Grapevine (Summary): The Growth of A Distributed System. *Operating Systems Review*, 17(5):141–142, October 1983.
- [123] Henning Schulzrinne and Elin Wedlund. Application-layer Mobility Using SIP. *ACM Computer Communications Review*, 4(3):47–57, July 2000.
- [124] Marianne Shaw. Leveraging Good Intentions to Reduce Unwanted Network Traffic. In *Proc. USENIX Steps to Reduce Unwanted Traffic on the Internet workshop*, San Jose, CA, July 2006.
- [125] John F. Shoch. Inter-Network Naming, Addressing, and Routing. In *Proc. of the Seventeenth IEEE Conference on Computer Communication Networks*, pages 72–79, Washington, D.C., 1978.
- [126] Georgos Siganos and Michalis Faloutsos. Analyzing BGP Policies: Methodology and Tool. In *Proc. IEEE INFOCOM*, Hong Kong, March 2004.
- [127] Thor Lancelot Simon. oof. panix sidelined by incompetence... again. <http://merit.edu/mail.archives/nanog/2006-01/msg00483.html>, January 2006.
- [128] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking*, 10(6), December 2002.
- [129] Mark Alexander Connell Snoeren. *A Session-based Architecture for Internet Mobility*. PhD thesis, MIT, Department of Electrical Engineering and Computer Science,

- February 2003.
- [130] Randall Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns J. Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream Control Transmission Protocol. RFC 2960, IETF, October 2000.
 - [131] Ion Stoica, Daniel Adkins, Shelley Zhaung, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *Proc. ACM SIGCOMM*, pages 73–86, Pittsburgh, PA, August 2002.
 - [132] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal de Lara, Christophe Diot, Ashvin Goel, Menghow Lim, and Eben Upton. Hagggle: Seamless Networking for Mobile Applications. In *Proc. of Ubicomp*, pages 391–408, September 2007.
 - [133] Lakshminarayanan Subramanian, Matthew Caesar, Cheng Tien Ee, Mark Handley, Morley Mao, Scott Shenker, and Ion Stoica. HLP: A Next Generation Inter-domain Routing Protocol. In *Proc. ACM SIGCOMM*, Philadelphia, PA, August 2005.
 - [134] Sun Microsystems. Java Message Service (JMS), June 2008. <http://java.sun.com/products/jms/>.
 - [135] Sun Microsystems. Java Platform Enterprise Edition (EE), June 2008. <http://java.sun.com/javaee/>.
 - [136] Fumio Teraoka, Yasuhiko Yokore, and Mario Tokoro. A Network Architecture Providing Host Migration Transparency. In *Proc. ACM SIGCOMM*, pages 209–220, Zurich, Switzerland, September 1991.
 - [137] Tibco. Tibco Enterprise Messaging Service, June 2008. <http://www.tibco.com/software/messaging/>.
 - [138] Niraj Tolia, Michael Kaminsky, David G. Andersen, and Swapnil Patil. An architecture for Internet data transfer. In *Proc. 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.
 - [139] George Varghese. *Network Algorithmics*. Morgan Kaufmann, 2007.
 - [140] Arun Venkataramani and Don Towsley. A Swarming Architecture for Internet Data Transfer, November 2007. <http://www.nets-find.net/Funded/Swarming.php>.
 - [141] Patrick Verkaik, Andre Broido, kc Claffy, Ruomei Gao, Young Hyun, and Ronald van der Pol. Beyond CIDR aggregation. Technical Report TR-2004-01, CAIDA, February 2004.
 - [142] David Waitzman, Craig Partridge, and Stephen E. Deering. *Distance Vector Multicast Routing Protocol*. Internet Engineering Task Force, November 1988. RFC 1075.
 - [143] Michael Walfish, Hari Balakrishnan, and Scott Shenker. Untangling the Web from DNS. In *Proc. of NSDI '04*, pages 225–238, San Francisco, CA, USA, March 2004.
 - [144] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes No Longer Considered Harmful. In *Proc. of OSDI 2004*, pages 215–230, San Francisco, CA, USA, December 2004.
 - [145] Dan Wendlandt, Ioannis Avramopoulos, David G. Andersen, and Jennifer Rexford. Don't Secure Routing Protocols, Secure Data Delivery. In *Proc. of Hot Topics in*

- Networks*, Irvine, CA, USA, November 2006.
- [146] Russ White. Securing BGP through secure origin BGP. *The Internet Protocol Journal*, 6(3), September 2003. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-3/ipj_6-3.pdf.
 - [147] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On The Scale and Performance of Cooperative Web Proxy Caching. In *Proc. 17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC, December 1999.
 - [148] Peter Wyckoff. T Spaces. *IBM Systems Journal*, 37(3):454–474, 1998.
 - [149] Abraham Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.
 - [150] Xiaowei Yang, David Clark, and Arthur W. Berger. NIRA: A New Inter-domain Routing Architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, 2007.
 - [151] Xiaowei Yang, David Wetherall, and Thomas Anderson. A DoS-limiting Network Architecture. In *Proc. ACM SIGCOMM*, Philadelphia, PA, August 2005.
 - [152] Tatu Ylönen and Chris Lonvick. *The Secure Shell SSH Protocol Architecture*. Internet Engineering Task Force, January 2006. RFC 4251.
 - [153] Victor C. Zandy and Barton P. Miller. Reliable Network Connections. In *Proc. ACM Mobicom*, Atlanta, GA, September 2002.
 - [154] Victor Charles Zandy. *Application Mobility*. PhD thesis, University of Wisconsin-Madison, 2004.
 - [155] Xinyang Zhang, Paul Francis, Jia Wang, and Kaoru Yoshida. Scaling IP Routing with the Core Router-Integrated Overlay. In *IEEE International Conference on Network Protocols (ICNP)*, Santa Barbara, CA, November 2006.



ISBN 978-951-22-9559-3
ISBN 978-951-22-9560-9 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)