Department of Communications and Networking

# State-Efficient Forwarding with In-packet Bloom Filters

Petri Laari

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
**DOCTORAL DISSERTATIONS** 108/2017

# State-Efficient Forwarding with In-packet Bloom Filters

**Petri Laari**

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall AS1, TUAS-building of the school on 14th of June 2017 at 13:00.

**Aalto University**
**School of Electrical Engineering**
**Department of Communications and Networking**
**Research Group for Protocols, Services, and Software**

**Supervising professor**
Prof. Jörg Ott, Aalto University, Finland and Technical University of Munich, Germany

**Preliminary examiners**
Dr. Bengt Ahlgren, SICS Swedish ICT, Sweden
Prof. Thomas C. Schmidt, Hamburg University of Applied Sciences, Germany

**Opponent**
Dr. Dirk Kutscher, Huawei Technologies Duesseldorf GmbH, Germany

Aalto University publication series
**DOCTORAL DISSERTATIONS** 108/2017



ISBN 978-952-60-7470-2 (printed)
ISBN 978-952-60-7469-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-7469-6

Unigrafia Oy
Helsinki 2017

Finland

Publication orders (printed book):
petri@laari.org

Aalto University

# Abstract

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

**Author**
Petri Laari

**Name of the doctoral dissertation**
State-Efficient Forwarding with In-packet Bloom Filters

**Publisher** School of Electrical Engineering

**Unit** Department of Communications and Networking

**Series** Aalto University publication series DOCTORAL DISSERTATIONS 108/2017

**Field of research** Networking Technology

**Manuscript submitted** 8 October 2015 **Date of the defence** 14 June 2017

**Permission to publish granted (date)** 16 December 2015 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

**Abstract**

The Internet Protocol (IP) has been the dominant packet forwarding mechanism in the Internet during the past decades. Despite of its merits, IP suffers from known shortcomings, such as ever expanding routing tables on network routers, limited support for multicast, and attacks that utilize destination address-based routing.

In this dissertation, we introduce a new packet forwarding mechanism that is not based on a global addressing scheme. In this solution, packet forwarding is based on strict source-routing. The calculated path from the sender to the receiver is encoded in a space-efficient Bloom filter, that is inserted in the packet header, and each router on the path can effiently verify the outgoing links where the packet should be delivered to.

Bloom filter-based forwarding has some advantages over IP forwarding. First, with Bloom filter forwarding, network routers do not need to maintain any routing table to make the forwarding decision for packets. Instead, the router verifies which of its interfaces have been included in the Bloom filter and forwards the packet out from matching interfaces. Second, the simplicity in Bloom filter verification allows a very efficient packet forwarding decision in a router. Finally, the Bloom filter forwarding offers a native multicast support without additional state in network routers.

A disadvantage of Bloom filter is that the verification may provide false positive results. With Bloom filter forwarding this means that sometimes additional packets are delivered over links that do not belong to the original path. We discuss this downside, verify the actual effects of false positives on forwarding decisions, and suggest potential solutions to overcome the problem.

Deploying a new forwarding mechanism is challenging because IP has a firm position in the Internet. We discuss about two potential deployment scenarios, where we show that Bloom filter-based forwarding can be deployed also in partial networks, replacing IP-based routing and multicast to enable more efficient networking.

**Keywords** Bloom filters, multicast, forwarding

**ISBN (printed)** 978-952-60-7470-2 **ISBN (pdf)** 978-952-60-7469-6

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

**Location of publisher** Helsinki **Location of printing** Helsinki **Year** 2017

**Pages** 179 **urn** http://urn.fi/URN:ISBN:978-952-60-7469-6

**Tekijä**
Petri Laari

**Väitöskirjan nimi**
Bloom-suodattimiin pohjautuva vähätilainen liikenteenvälitysjärjestelmä

**Julkaisija** Sähkötekniikan korkeakoulu

**Yksikkö** Tietoliikenne- ja tietoverkkotekniikka

**Sarja** Aalto University publication series DOCTORAL DISSERTATIONS 108/2017

**Tutkimusala** Tietoverkkotekniikka

**Käsikirjoituksen pvm** 08.10.2015 **Väitöspäivä** 14.06.2017

**Julkaisuluvan myöntämispäivä** 16.12.2015 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

**Tiivistelmä**
Viimeisten vuosikymmenien aikana IP on ollut hallitseva liikenteen välitysmenetelmä Internetissä. Huolimatta hyödyistään, IP kärsii tietyistä ongelmista kuten jatkuvasti kasvavista reititystauluista reitittimissä, rajoitetusta monilähetyksen tuesta sekä hyökkäyksistä, jotka käyttävät hyväkseen kohdeosoiteperusteista reititystä.

Tässä väitöskirjassa me esittelemme uuden lähdereititykseen pohjautuvan liikenteenvälitysmenetelmän, joka ei käytä maailmanlaajuista osoitteistusta. Etukäteen laskettu polku lähettäjältä vastaanottajalle lisätään tilankäytöltään tehokkaaseen Bloom-suodattimeen, joka vuorostaan sijoitetaan lähetettävän paketin otsakkeeseen. Verkon reitittimet tarkistavat paketin suotimesta onko jokin niiden ulospäin lähtevistä rajapinnoista sisällytetty suotimeen, ja lähettävät paketin ulos sisällytetyistä rajapinnoista.

Bloom-suodattimeen perustuvalla pakettien välityksellä on eräitä etuja IP-pohjaiseen välitykseen nähden. Ensinnäkin reitittimien ei tarvitse pitää yllä reititystauluja koska pakettien välitys perustuu rajapintojen identiteettien tarkistamiseen Bloom-suodattimesta. Toiseksi identiteetin tarkistaminen Bloom-suodattimesta on hyvin yksinkertainen toimenpide, ja se mahdollistaa erittäin nopean välityspäätöksen tekemisen reittimessä. Lopuksi Bloom-suodattimeen perustuva reititys mahdollistaa luontaisen monilähetyksen ilman ylimääräistä tilatietoa verkon reittimissä.

Bloom-suodattimiin perustuvissa järjestelmissä on myös haittapuolensa. Yksittäisen alkion tarkastaminen suodattimesta saattaa aiheuttaa virheellisen positiivisen vastauksen. Pakettien välityksessä tämä tarkoittaa ylimääräisen paketin kopion lähettämistä alkuperäiseen polkuun kuulumattomalle yhteydelle. Keskustelemme tästä haittapuolesta, todennamme sen haitat välityspäätöksen teossa ja ehdotamme mahdollisia ratkaisuja ongelman ratkaisuun.

Uuden välitysjärjestelmän käyttöönotto on haastavaa, koska IP:llä on hyvin vahva asema Internetissä. Keskustelemme kahdesta erilaisesta käyttöönottomahdollisuudesta ja näytämme, että Bloom-suodattimeen perustuva välitys voidaan ottaa käyttöön myös rajoitetuissa osissa Internetiä, korvaten IP-pohjaisen välityksen ja monilähetyksen, mahdollistaen tehokkaamman liikenteen välityksen.

**Avainsanat** Bloom-suodattimet, monilähetys, liikenteenvälitys

**ISBN (painettu)** 978-952-60-7470-2 **ISBN (pdf)** 978-952-60-7469-6

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

**Julkaisupaikka** Helsinki **Painopaikka** Helsinki **Vuosi** 2017

**Sivumäärä** 179 **urn** http://urn.fi/URN:ISBN:978-952-60-7469-6

# Preface

A long and extremely interesting journey has come to the end with the publication of this thesis. During the years, I have had an opportunity to work with many colleagues in a number of projects. I have also had a chance to learn lots of new things, both technical and non-technical. And also many other things have changed in my life, even my last name changed from Jokela to Laari.

The initial inspiration for the packet forwarding proposal presented in this thesis came from Tero Kauppinen when he introduced Bloom filters to me in a completely different context. The idea behind the filters sounded interesting and during one dark night I got an idea of using Bloom filters in data packets to assist forwarding in a stateless network. We further developed the idea with Jan Melén and the raw idea later became zFilters and finally in-packet Bloom Filters.

I want to thank my professor Jörg Ott for the guidance during the work. The writing process required that someone was kicking me further. I want also to thank Pekka Nikander for his support and guidance in the early stages of my work. In the beginning, he gave a good lesson when we where submitting our results into a conference: If you fail, instead of lowering the bar you can always raise it and try harder. With that in mind, the initial work was successfully accomplished and the paper to Sigcomm submitted. He was also the one who trusted on the idea (almost) from the beginning.

I am thankful to my pre-examiners Dr. Thomas Schmidt and Dr. Bengt Ahlgren for their time and effort on reviewing the thesis as well as their feedback to make the thesis better.

I want to thank also my managers, Raimo Vuopionperä, Tony Jokikyyny, Jan Melén, and Johan Torsner for their support during my studies. In addition, there is a number of colleagues from our research laboratory in

Finland who have given support making it possible to finalize the thesis. Also many people involved in the EU-funded PSIRP and PURSUIT projects have affected the work.

I want to thank also my co-authors Christian Esteve, Somaya Arianfar, Heikki Mahkonen, András Zahemszky, Jari Keinänen, Kristian Slavov, Jukka Ylitalo, Mikko Särelä, Sami Ruponen, and James Kempf for their effort to make the publications possible. There have been multiple people revieweing the thesis during the final steps. Thanks to Jimmy Kjällman, Jukka Ylitalo, Miika Komu, and Tero Kauppinen for their efforts to get the final things correct.

I am grateful to my parents Eila and Aulis Jokela who have supported me all the way from the beginning of my studies in the 1st grade to the end. One turning point that guided me into the computerized world was the Commodore VIC-20 computer that they bought me in the beginning of 1980's.

The most important support that I have received during the final years has come from my loving wife Marita. She has given her endless support for me to finalize the thesis and she has been constantly pushing me forward when I have been desperate with the progress. In addition, I would like to thank my sisters Merja and Seija for the support that I have got, especially Merja and her husband Kaj for my first apartment when I started my studies. My children Sami, Emmi, and Anni have spent their whole lives seeing me working on post graduate studies. What can I say, dad is still studying... I hope that I have been able to give them also something back that helps them in their own studies.

The work was funded by Ericsson, by EU in two separate projects (PSIRP and PURSUIT), and partly by Finnish ICT-SHOK Future Internet program.

Espoo, May 18, 2017,

Petri Laari

# Contents

# List of Publications

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

**I** P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, P. Nikander. LIPSIN: Line-Speed Publish/Subscribe Inter-Networking. *In Proc. of the ACM SIGCOMM 2009*, Barcelona, Spain, August 2009.

**II** J. Keinänen, P. Jokela, K. Slavov. Implementing zFilter based forwarding node on a NetFPGA. *In NetFPGA Developers Workshop*, Stanford, CA, August 2009.

**III** C. Esteve Rothenberg, P. Jokela, P. Nikander, M. Särelä, J. Ylitalo. Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters. *In Proc. of the European Conference on Computer Network Defense, EC2ND'09*, Milan, Italy, November 2009.

**IV** A. Zahemszky, P. Jokela, M. Särelä, S. Ruponen, J. Kempf, P. Nikander. MPSS: Multiprotocol Stateless Switching. *In Proc. of the 13th IEEE Global Internet Symposium 2010*, San Diego, CA, USA, March 2010.

**V** A. Zahemszky, P. Jokela, T. Jokikyyny. Autonomicity in Virtual Private Network Provisioning for Enterprises. *In Proc. of the IEEE International Workshop on Management of Emerging Networks and Services (IEEE MENS)*, Miami, Florida, USA, December 2010.

**VI** P. Jokela, H. Mahkonen, C. Esteve Rothenberg, J. Ott. (Deployable) Reduction of Multicast State with In-packet Bloom Filters. *In Proc. of the IFIP Networking 2013*, New York, May 2013.

# Author's Contribution

### Publication I: "LIPSIN: Line-Speed Publish/Subscribe Inter-Networking"

The author designed a link identity -based forwarding mechanism where the required link identifiers of each of the routers on the packet's intended path is inserted into a Bloom filter. The Bloom filter represents a source routed tree where all the required path information is compressed and the filter is further inserted in the packet header to be used as the packet forwarding information. In addition to a physical link, the link identifier can be internal in the node, providing a passway for the packet from the forwarding path to the node internal processing.

The author developed the virtual tree -based solution to reduce the number of bits set to 1 in the forwarding identifier for improving performance. In addition, he developed an iBF collecting procedure to create a reverse direction iBF without the need to calculate the iBF separately in the Topology Manager. Overall, the author participated as part of the team for defining the enhancements to the basic forwarding mechanism, in the design of the simulator as well as in the evaluation of the results.

The author participated in the paper writing as one of the main contributors. He was the main responsible for editing the final version to the conference.

### Publication II: "Implementing zFilter based forwarding node on a NetFPGA"

The author provided the required input to adapt in-packet Bloom filter -based forwarding ideas to the NetFPGA router environment where we

made a hardware-level implementation of the forwarding mechanism. We used the implementation to measure the iBF forwarding performance on the NetFPGA router and we compared the results with the IP reference router on the same platform. The author was designing the measurement requirements for the implementation to get the understanding of the forwarding performance of the new forwarding mechanism.

The author participated in the paper writing as the main contributor to the in-packet Bloom filter related parts and as a member of the team to the implementation and result handling part.

### Publication III: "Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters"

The author was a member of the team that defined the method to create on-line the outgoing interface's LID based on various input from the packet and the router. This method can be used to tie the LID to a certain packet flow, providing added security.

Author contributed mainly on the node operation, where the physical interfaces can be used to select the visiting order for nodes based on the incoming and outgoing interfaces and similarly enabling virtual interfaces inside the node for different services to control the visited services inside a node. There is also a related Patent Application containing detailed information that was not included in the publication.

### Publication IV: "MPSS: Multiprotocol Stateless Switching"

The author defined the OSPF network topology information usage as the source for forming the zFilter for forwarding on the edge node. In addition, the author defined the resource reservation method to be used in the MPSS system, especially the resource allocation in the nodes. The author participated also as part of the team in the definition of the MPSS system.

### Publication V: "Autonomicity in Virtual Private Network Provisioning for Enterprises"

The author was part of the team defining the GANA model usage in the MultiProtocol Stateless Switching networks to provide autonomicity in the Virtual Private Network management. He was contributing mainly to the in-packet Bloom Filter usage in the MPSS system. The paper provides information about the potential deployment of iBFs for MPSS based VPN usage.

### Publication VI: "(Deployable) Reduction of Multicast State with In-packet Bloom Filters"

The author defined the gradual deployment of iBF forwarding in IP multicast networks, including the required iBF mappings to the IP multicast group identifiers. He also implemented the simulation environment based on the existing iBF forwarding simulator for evaluating the operation itself and the potential advantage of using iBF instead of IP multicast. The simulator implementation supported also the Virtual Path system, invented and presented by the author originally in Publication I.

## List of Abbreviations

**AD** Accountability Domain

**AIP** Accountable Internet Protocol

**ALG** Application Level Gateway

**AOLS** All-Optical Label Switching

**APNIC** Asia Pacific Network Information Centre

**ARIN** American Registry for Internet Numbers

**AS** Autonomous System

**ASM** Any-Source Multicast

**BGP** Border Gateway Protocol

**BRA** Bloom filter -based Relay Architecture

**CDN** Content Delivery Network

**CE** Customer Edge

**CIDR** Classless Interdomain Routing

**CN** Correspondent Node

**DDoS** Distributed Denial of Service

**DE** Destination Edge

**DFZ** Default Free Zone

**DHT** Distributed Hash Table

**DNS** Domain Name System

**DONA** Data-Oriented Network Architecture

**DoS** Denial of Service

**EGP** Exterior Gateway Protocol

**EID** End-host Identifier

**ESP** Encapsulating Security Payload

**FIB** Forwarding Information Base

**FRM** Free Riding Multicast

**GMPLS** Generalized Multi-Protocol Label Switching

**GSE** Global, Site, and End-system address elements

**HI** Host Identifier

**HIP** Host Identity Protocol

**HTTP** Hypertext Transfer Protocol

**i3** Internet Indirection Infrastructure

**iBF** in-packet Bloom filter

**iBGP** internal Border Gateway Protocol

**ICN** Information Centric Networking

**IETF** Internet Engineering Task Force

**IGP** Interior Gateway Protocol

**IoT** Internet of Things

**IP** Internet Protocol

**IS-IS** Intermediate System to Intermediate System

**IS-IS-TE** Intermediate System to Intermediate System - Traffic Engineering

**ISO** International Standardization Organization

**ISP** Internet Service Provider

**IXP** Internet eXchange Point

**L3VPN** Layer 3 Virtual Private Network

**LAN** Local Area Network

**LDP** Label Distribution Protocol

**LId** Link Identifier

**LIT** Link Identity Tag

**LLC** Logical Link Control

**LSP** Label-Switched Path

**LSR** Label Switch Router

**LSRR** Loose Source Record Route

**MAC** Media Access Control

**MN** Mobile Node

**MPLS** Multiprotocol Label Switching

**MPSS** Multiprotocol Stateless Switching

**MTU** Maximum Transmission Unit

**NAT** Network Address Translation

**NIC** Network Interface Card

**O-E-O** Optical-Electronic-Optical

**OSI** Open System Interconnection

**OSPF** Open Shortest Path First

**OSPF-TE** Open Shortest Path First - Traffic Engineering

**OUI** Organizationally Unique Identifier

**PCE** Path Computation Element

**PE** Provider Edge

**PPP** Point-to-Point Protocol

**ProHet** Probabilistic Routing Protocol for Heterogeneous Sensor Networks

**PRoPHet** Probabilistic Routing Protocol for Intermittently Connected Networks

**PSIRP** Publish-Subscribe Internet Routing Paradigm

**PURSUIT** Publish Subscribe Internet Technology

**QoS** Quality of Service

**RFC** Request For Comments

**RIP** Routing Information Protocol

**RIPng** Routing Information Protocol - next generation

**RIB** Routing Information Base

**RIPE** Reseaux IP Europeans

**RIR** Regional Internet Registries

**ROFL** Routing on Flat Labels

**RSVP** Resource Reservation Protocol

**RSVP-TE** Resource Reservation Protocol - Traffic Engineering

**RVS** Rendezvous Server

**SAIL** Scalable and Adaptive Internet Solutions

**SDH** Synchronous Digital Hierarchy

**SE** Source Edge

**SEATTLE** A Scalable Ethernet Architecture for Large Enterprises

**SPU** Serial Processing Unit

**SSM** Source-Specific Multicast

**SSRR** Strict Source Record Route

**TCP** Transmission Control Protocol

**TE** Traffic Engineering

**TM** Topology Manager

**TTL** Time to Live

**UDP** User Datagram Protocol

**URL** Uniform Resource Locator

**VLId** Virtual Link Identifier

**VPN** Virtual Private Network

**VRF** Virtual Routing and Forwarding

**XCast** Explicit Multicast

## List of Symbols

$\alpha$ Input for function $f$ from an incoming packet

$\beta$ Input for function $f$ from the router itself

$f$ A function used in a router to make a forwarding decision

$iBF$ in-packet Bloom filter from an incoming packet

$k$ Number of hash functions used when inserting items in a Bloom filter

$k_{opt}$ Optimal value for $k$, providing best performance

$LId$ Link Identifier for an outgoing interface on a router

$m$ Length of a Bloom filter in bits

$n$ Number of items inserted in a Bloom filter

$P$ False positive probability

# 1. Introduction

A large amount of the fundamental work on computer networking protocols was carried out during the 1960s and 1970s. The Internet communication model and the base protocols designed at that time [90] are still in use.

At the dawn of networking, computers were big and they were not meant to be moved from their locations. Thus, the networking environment was relatively stable. The transmission speed of the links between the computers was slow, setting limits for the amount of data that could be delivered. On the other hand, users' needs were very different from those we experience today.

The world has changed and the network environment is more complicated nowadays. Today, almost any device can have a network connection. Devices can be big, small, portable, or fixed, and they may have different kinds of data transmission requirements. These features affect protocol development, e.g., how to support small devices with limited battery life.

User behavior has also changed since the Internet became widely available for consumers. For example, high definition video broadcast over the Internet is becoming more popular all the time and television broadcasts are moving to the Internet instead of using only the traditional air-based transmission [19].

The change in network usage has also changed security requirements of the Internet protocols. The more people use the Internet, especially for money transfers and shopping, the more attractive the network becomes for criminals. Attacks and frauds have become part of everyday life in the Internet, increasing the need to protect users and their information.

A huge amount of work has been done to fix these emerging problems with e.g. unwanted traffic, amount of state in the network, or lack of efficient multicast solution. Some of the problems can be patched using

application level solutions, for example adding application level multicast support to routers, or by adding additional nodes in the network, for instance firewalls to block unwanted traffic. We have taken a different approach to solve some of the problems.

In this thesis, we describe a new forwarding mechanism that uses a forwarding identifier that is tied to the path from the sender to the receiver or receivers. The identifier is a compact, fixed size bitstring that defines the path through the network. The bitstring as such is meaningless in routers that do not belong to the path, but at a router on a path it can be used to determine the correct output interfaces from that router. In addition to the path, the identifier can also be tied to other information, e.g. to a flow information contained in the forwarded packet. The solution provides protection against unwanted traffic, but it also introduces an efficient, quasi-stateless multicast solution.

## 1.1 The Current Internet

The Internet is a complex combination of networks and operators. In this section, we shortly introduce the current networking environment. This gives the context for our work that we present in Chapter 4.

### 1.1.1 Players in the Internet

The Internet is, as the name says, a network consisting of interconnected smaller networks. Each of these networks is operated by a service provider, which can be e.g. an association providing network connectivity for its affiliates, an Internet Service Provider (ISP) providing connectivity for its customers, or a company providing network access for its employees.

The network operators have been divided into three different categories: Tier-1, Tier-2, and Tier-3 operators (see Figure 1.1). The categorization is based on their role in the network and on the agreements that they make with each other. This division is vague and sometimes it is not clear to which category an operator belongs.

Tier-1 operators are the top-level operators. They connect to each other and provide services to Tier-2 operators. Tier-1 operators do not pay for any access, but they make settlement-free peering contracts with other Tier-1 operators. [1]

[1]This is not easy to verify because contracts between operators are not public

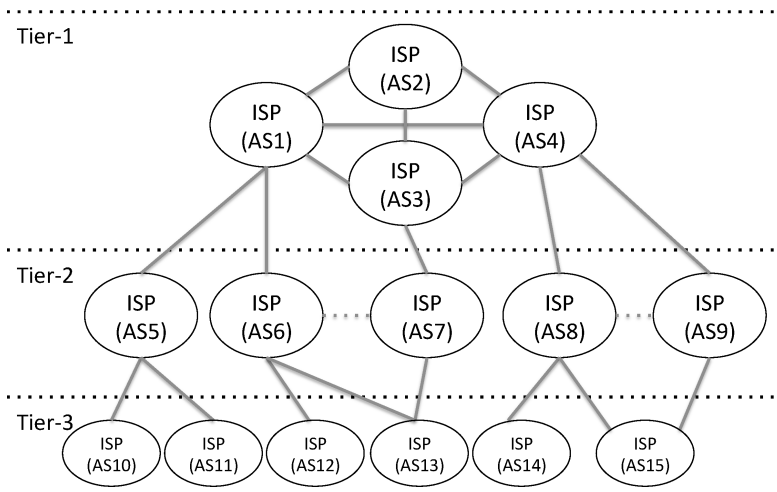



**Figure 1.1.** Internet architecture

Tier-2 operators buy connection services from Tier-1 operators. They can utilize the nationwide or international connections of Tier-1 operators to communicate with other Tier-2 networks. In addition, Tier-2 operators may sometimes make their own peering agreements with other Tier-2 operators. Tier-2 operators sell connection services further to Tier-3 operators.

Tier-3 operators provide connections to end-users. Usually, Tier-3 operators do not make peering agreements with other operators. However, they can buy connections from multiple Tier-2 operators to ensure reliable Internet access.

Each operator manages one or more Autonomous Systems (AS) and usually one AS is managed by a single operator. An AS is defined in RFC 1930 [50, p. 3] as follows:

> "An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy."

Usually, an AS is managed by a single operator.

Tier-1 networks consist of a handful of ASes. At the Tier-2 level, there exists a couple of thousand ASes. Tier-3 networks, called also Stub ASes, consists of the majority of ASes ($\sim$ 85 - 90%) [100]. The current number of ASes in the Internet is over 42 000 [1].

information and the actual content of the agreements is unknown.

### 1.1.2 Connecting the networks

Autonomous Systems consist of a set of nodes connected to each other. Starting from the smallest networks, the nodes can be connected to each other via switches that deliver traffic between the nodes. These Local Area Networks (LAN) are interconnected using specific nodes called routers. Operators interconnect their networks also with routers, enabling communication between nodes residing on different operator networks.

As we described in the previous section, Tier-1 operators provide global connectivity for other operators. They manage Internet Exchange Points (IXP) where other operators can connect their networks. These IXPs are further interconnected with high speed links.

Routing traffic between the nodes in the Internet requires protocols that exchange routing information between routers. In the following section, we shortly introduce the current Internet addressing and routing mechanisms.

### 1.1.3 Addressing and Forwarding

Packet routing and forwarding in the global Internet requires a proper addressing scheme. The Internet Protocol (IP) version 4 became the dominant addressing and communication protocol after it was introduced in the 1970s and further standardized in 1981 [92]. IPv4 uses 32 bits long addresses which are divided into network prefix and host parts. Network prefixes are managed and assigned to operators by three authorities as we describe in Section 2.1.2. The operators can further divide their prefixes into smaller subnetworks and assign them to their customers. A prefix defines the topological location of the network in the Internet. The host part is used inside a LAN to forward the packet to the correct destination host.

The number of computers connected to the Internet has been growing so fast that the 32-bit address space of IPv4 is not sufficient. At the time of writing, all IPv4 address blocks have already been allocated [48]. The problem is getting worse because the number of different kinds of devices connected to the Internet is increasing. For example, sensor networks that consist of a huge number of small devices, are emerging. In these networks each individual sensor can be connected to the Internet with its own IP address [87, p. 1].

Various workarounds have been developed to mitigate the lack of IPv4

addresses. Solutions, such as Network Address Translation (NAT) [114] and Classless Inter-Domain Routing (CIDR) [97], were introduced when it became obvious that the IP address space is going to run out some day in the future. A NAT allows using a private address space [99] in the network behind it and only the NAT device requires a public IP address on the interface connected to the Internet. The NAT device makes address translation between the local and the public IP addresses. CIDR defines usage of variable length network prefixes making it possible to allocate prefixes with better precision to the needs of the operator and wasting less of the address space when there is no need for a large address space. However, NATs and CIDR provided only temporary solutions and the IPv4 address space has already been depleted. As a permanent solution, a next generation Internet protocol has been developed and taken into use, namely IP version 6 [36]. The deployment of IPv6 is still ongoing at the time of writing.

IP routing is based on the IP address hierarchy. As mentioned, the address consists of two parts. The network prefix defines the topological location of the network in the Internet and the host part identifies the host at its network. Intra-domain routing handles the routing information exchange inside an AS; it exchanges detailed information of the routers and connections between them so that the routers can build the routing tables (see Section 2.2.4). These protocols, however, cannot be used to distribute routing information between ASes, e.g. because the amount of information would be huge. Thus, a different protocol is used for the inter-domain routing purposes. We discuss this in Section 2.2.5.

Routing state is maintained in the network routers. They store network prefix information and map them to the next hop routers. The only information needed to deliver the packet to the destination is the IP address of the destination host. We describe the IP routing mechanism in Section 2.2. For some specific purposes, IP allows adding some routing state in the packet header. Source routing adds a path in the IP header that lets the source node to define the path that the packet must follow (see Section 2.3.3).

The communication model in the Internet is currently host-centric. Information is identified using the host name as part of the identifier, e.g. when retrieving a page from the web where the Uniform Resource Locator (URL) [72] starts with the host name. The host name is further resolved to the host's IP address using the Domain Name System (DNS) [75]. When

the information is retrieved, the connection is established between the two hosts. Content Delivery Networks (CDN) [83] are a step away from the host based information identification. Information is stored in the network and the retrieving host does not know up front from where the data arrives. In networking research, this kind of content based identification has been taken further. For example, in the Publish-Subscribe Internet Routing Paradigm (PSIRP) [3] and PURSUIT [4] architectures information is identified using an identifier that is not tied to the location of the information. The information can be linked to the publisher using cryptographic means; the retrieving user is usually interested in the publisher, not in the location of the information. The pure information centric approach does not require a forwarding mechanism that is also used to identify the information. This enables the use of other forwarding mechanisms instead of IP.

Group communication in the Internet is communication from one node to multiple receivers (one-to-many) or between multiple nodes where information is distributed to all group members (many-to-many). Group communication can be implemented as either network level or application level multicast. There are many implementations of group communication in the Internet - for example video conferencing or text-based instant messaging services utilize it.

The need for group communication in the network has increased because the communication habits of users have changed. For example, the amount of live media streaming over the Internet is increasing [10, p. 3]. These streams have usually one source and multiple destinations. In such scenarios, point-to-point connections are inefficient because the data is sent in multiple copies over the same links.

Multicast support was added to IP during the 1980s [38], [53], but it has never been deployed widely [94, p. 33]. After that, other multicast solutions have been deployed, such as P2P networks. One alternative is to implement multicast at application level. This is easier to deploy because the application-level protocol does not have to be supported by all routers in the network. However, application-level multicast implementations are not as efficient as lower layer implementations, since packets are processed at the application level in all the branching routers where the packets are multiplied.

Neither IPv4 nor IPv6 base protocols support mobile hosts natively. Mobility support was added to the protocols afterwards and it is specified

in RFC 5944 [84] for IPv4 and in RFC 6275 [86] for IPv6. However, mobile IP protocols are rarely used in the end-hosts. When user devices are connected to cellular networks, the mobility protocols of the cellular systems handle terminal mobility and IP operates on top of those connections. Thus, the terminal is using a fixed IP address when connecting to the Internet and it is not mobile from the Internet perspective.

### 1.1.4 Enhancing Forwarding

Since the introduction of IP, various enhancements have been proposed and taken into use to achieve more efficient forwarding. One enhancement is the Multiprotocol Label Switching (MPLS) protocol [104] which is based on creating predetermined paths for connections through the network. Each router identifies a path with a pair of small labels and a corresponding outgoing interface. The label in the incoming packet's header is used to determine the outgoing link and a new label for the outgoing packet. Label matching makes the packet forwarding faster than with IP forwarding because there is no need to check a routing table.

The IP-based packet forwarding is not optimal in all environments and there is still space for enhancements. For example, optical fibers are used for high speed data transmission. However, in optical routing IP requires typically Optical - Electronic - Optical (O-E-O) conversion for the packet header at the optical routing points; it is difficult to verify IP routing tables in the optical domain and the routing decision in current commercial routers is done in the electronic domain [95, p. 653]. Moving a packet - or a packet header - from the optical domain to the electronic domain slows down packet forwarding. It would be more efficient to make the packet forwarding decision on the optical layer. For optical packet forwarding, MPLS-like label switching provides a better alternative. MPLS label switching can be done in the optical domain but these All-Optical Label Switching (AOLS) devices do not scale well. For each label that passes the router, the router needs a separate piece of hardware [113, p. 1329]. Hence, there is a need for scalable all-optical forwarding solution.

## 1.2 Research Problem Area

IP is currently the default communication protocol that is used globally in the Internet. However, the current Internet has some limitations.

First, the ease of IP-based communication allows also malicious traffic to be delivered in the network. With IP, any node in the network can send traffic to any other node just by inserting the destination IP address in the packet and the network handles the packet delivery. The destination node cannot decide if it wants to receive traffic from the sending host or not. This opens possibilities for different kinds of attacks against the host. We discuss the potential attacks in Section 3.1.2.

There are ways to limit the attack possibilities. For this purpose, specific nodes are used in the Internet, for example firewalls. However, additional nodes add complexity to the network. One of our research topics is to try to find solutions that could limit the attack possibilities without adding complexity.

Second, packet routing decisions in the IP routers are based on routing tables. A packet is forwarded to the next hop router if the packet's destination address matches a routing table entry. This procedure has been optimized, but in certain environments the complexity is still a problem. For example, optical fibers are used to achieve fast data delivery but the routing decision at an optical router has to be made at the electrical domain. Thus, a simple mechanism to make forwarding decision at the optical domain would increase the efficiency of the network.

Third, energy consumption in routers increases the more traffic they handle. Energy consumption has bad side effects, such as increased heating in large computer halls in addition to increased energy cost. One way to decrease energy consumption is to increase the forwarding efficiency. If we manage to decrease the complexity of forwarding decisions, we may also be able to reduce energy consumption.

Fourth, audio and video transmissions are currently taking an increasing share of the network bandwidth both in mobile and fixed networks [10, p. 5]. Media streaming services could take advantage of multicast if it was implemented in the network. Instead of setting up point-to-point links from a server to all the clients, it is more bandwidth-efficient to duplicate data packets closer to the receivers. As mentioned earlier, IP multicast has not made a breakthrough and Internet is still lacking an efficient, low-level multicast support.

Finally, we can consider the tight binding between information identification and location, i.e. IP address, as one potential problem. Information is usually identified using the host name as one part of the information identifier and this ties the information to a certain location. Currently,

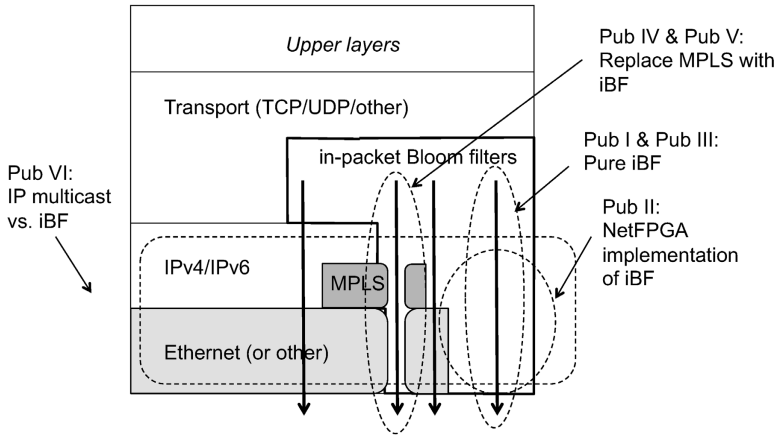



**Figure 1.2.** The research area and publication contributions

the binding between the information identification and the IP address forwarding makes it hard to deploy other forwarding mechanisms in the network. This situation is slowly changing, opening possibilities for new kinds of forwarding solutions.

## 1.3 Contributions

In this thesis, we propose a new in-packet Bloom filter (iBF) packet forwarding mechanism that provides new solutions to the research problems presented in Section 1.2. iBF forwarding is not dependent on any global addressing schemes and maintains only minimal amount of state in the routers. With theoretical studies, simulations, and prototype implementations we proved the concept. The solution mainly affects the lower layers when mapped to the current protocol stack (see Figure 1.2). The figure shows also the areas where we have been contributing in our publications.

First, the proposed iBF solution provides a packet forwarding mechanism that requires knowledge of the link identifiers forming the path from the packet source to the destination. While in IP-based forwarding the source node can send traffic to random IP addresses and the network delivers the traffic to the destination using the routing information in the network routers, with iBFs forwarding the source needs to have a precreated iBF for a specific destination node or set of nodes for delivering traffic to the desired destinations. The solution rules out the possibility to massively send traffic to destination nodes just by guessing the iBF in the packet header. This basic solution is presented in Publication I and

it was further enhanced in Publication III, where we included some additional information included in the iBF calculation. Now we can tie the packet forwarding to the flow information in the packet and to the used path depending on the input and output interfaces on a forwarding node.

Second, the iBF forwarding can solve the problem of growing routing tables in the network nodes. While the IP-based forwarding is using large routing tables in the routers, we do not need to have that amount of information as the forwarding decisions are forwarding node specific ( Publication I) and the forwarding information is compressed in a Bloom filter in the packet header. We compared our iBF mechanism to an existing multicast solution, namely Mobile IPv6, in Publication VI.

Although we theoretically estimated that it is possible to reduce the energy consumption in the nodes by reducing the complexity of forwarding decisions, we did not make any measurements on energy consumption. However, in Publication II, we showed that the iBF forwarding decisions are less complex compared to IP-based solutions. This gives a hint that also energy consumption is lower, which has been shown later by work done by others in this field [129].

As a fourth point, we provide a multicast solution that reduces the amount of state that is required in the network routers and providing more efficient ways to do forwarding. This was simulated in Publication VI. With iBF-based forwarding, we do not need to set state for each of the multicast stream that is passing a router. Thus, we save a considerable amount of memory in these routers.

As a use case, we introduced MPLS using iBFs. This shows that it is possible to deploy iBF and benefit from it without affecting the IP based routing. The same result can be seen in Publication VI with higher layer solution where IPv6 can be replaced with iBFs.

We studied the scalability of the basic solution as well as scalability with certain enhancements. The results show that in a network topology with 141 nodes, we achieve relatively good results with few tens of receiving multicast nodes. Because the efficiency of forwarding is measured calculating the additional traffic caused by false positive matching in routers, the results depend on the allowed amount of additional traffic. For our case, we typically used 95 percent of correct traffic as the threshold value. The performance is also affected by the selected properties for the Bloom filter. Increasing size, the results are typically better, but the downside is that packets get bigger. This has been discussed in Section 4.1.4.

| | | Publication | | | | | |
|---|---|---|---|---|---|---|---|
| | | I | II | III | IV | V | VI |
| Methodology | Prototyping | X | X | | | | |
| | Simulation | X | | | | | X |
| | Design | X | | X | X | X | |
| | Performance evaluation | X | X | | | | X |
| | Scalability analysis | X | X | | X | X | X |

**Table 1.1.** Methodologies used in our publications

In the work presented in the listed publications, we have used different methods for evaluating the proposed iBF forwarding solution. Table 1.1 summarizes the different research methods used in the publications. The reasoning for the particular methods is described in the publications. For Publication I, the approach was to have the basic design on paper to understand how the concept works. We had to make scalability analysis and performance evaluation to understand if the system performs well also with larger number of nodes. Basically, we had to use both simulations for the scalability and prototyping for understanding the lower level operations and to verify that the selected approach is feasible.

For Publication II, prototyping was the essential way of proving the operation on the selected hardware platform. Using the measured results we could make the performance and scalability analysis. Publication III showed the security solution on the design level which gave the required information to understand the potential solution for the problems described in the paper. Publication IV and Publication V were both showing the design of the respective systems and scalability analysis was done to prove that they can be also used in networks that are practical.

In Publication VI we used simulation for getting results as we needed larger topologies than were possible to do in research environment with real hardware. Based on the results we could do performance evaluation and scalability analysis.

During his earlier career, the author has been working with e.g. IPv6, mobility, and Host Identity Protocol (HIP). This work has been published in various conferences and workshops, for example [89], [61], [60], [59], [126]. The author has also been active in IETF with HIP standardization, resulting in two Standards Track Requests For Comments (RFCs) [76], [57].

## 1.4 Structure

The rest of this thesis is organized into four chapters.

In Chapter 2, we describe the relevant background information. We start from IP routing, including the routing information exchange in the network, and we go further into packet forwarding. In addition, we present the current IP multicast technologies.

In Chapter 3, we introduce the problems with the current Internet and present research results on some new ways to handle internetworking. We give also an introduction to Bloom filters and how they have been proposed to be used to help packet routing and forwarding in networks.

In Chapter 4, we present a new packet forwarding mechanism that is based on in-packet Bloom filters. In addition to the basic solution, we provide security and performance enhancements. Introducing new technology in the Internet is not straightforward. Thus, Chapter 4 describes also various deployment scenarios, where the forwarding mechanism can be deployed gradually in the network, improving the performance of the operator's network. Finally, Chapter 5 concludes the thesis.

# 2. Forwarding in Packet Switched Networks

The Internet is a collection of different kinds of networks with a huge diversity of nodes. Communication between nodes is not trivial in this heterogeneous environment. Information sent from a source node must usually cross multiple network borders before reaching the destination node.

Communication between two distant nodes requires mechanisms to contact the peer node and to deliver information through the network. In this chapter, we describe the packet forwarding and routing mechanism used today in the Internet. The Internet Protocol (IP) [92] is globally deployed, but it has some limitations that have emerged e.g. due to changes in users' behavior.

## 2.1 Internet Protocol

The communication environment in the Internet is challenging. The variety of different types of nodes and networks is huge and sets different kinds of requirements for the communication protocols. The nodes have different hardware and operating systems and they use different access technologies and still they should be able to communicate with each other using standardized protocols.

In a heterogeneous environment, communication between nodes requires protocols that can support data exchange between the nodes over the network borders. IP was introduced during the 1970s and it became quickly the default communication scheme for networked hosts. IP creates a virtual view of the underlying physical network. It hides the physical setup of the network from the higher layer protocols. In Figure 2.1, the layered model of the Internet Protocol suite is depicted. IP is a layer 3 protocol, below the transport control (Transmission Control Protocol (TCP) [93] and

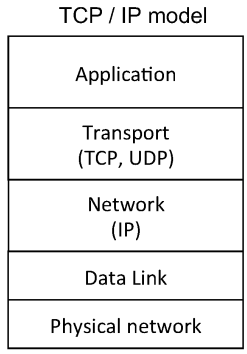



**Figure 2.1.** TCP/IP layered model [116, p. 18]

User Datagram Protocol (UDP) [91]) and above the link layer and physical layer. For various reasons, the protocol handling is not so strictly bound to the separate layers [27].

### 2.1.1 Functional Division

The communication model in the Internet consists of different functions that are needed for establishing and maintaining a connection [111]: *naming*, *addressing*, and *routing*.

*Naming* is used to assign a name for a host. The name is usually a human readable string and it provides user friendliness. Computers cannot handle character strings efficiently enough, thus strings are present usual only on the application level. To make processing more efficient, the host resolves the name string into a computer friendly IP address using some resolution mechanism, such as the Domain Name System (DNS) [75].

*Addressing* gives topologically meaningful addresses to hosts. Each node, or in fact, each interface of a node that is connected to the network, needs a globally unique address. The address identifies the node and describes its topological location in the network. Using this address, other nodes in the Internet can reach this specific node.

Finally, *routing* creates predetermined paths for delivering packets from the source to the destination node. Routing can be divided into two different areas: intra domain routing handles routing inside a single administrative domain and inter-domain routing between different administrative domains. To clarify the difference between $routing$ and $forwarding$, we use the $forwarding$ term to refer to the operation of a router or a forwarding node, when the node makes packet forwarding decision based on routing information that has been defined earlier.

### 2.1.2 Addressing in the Internet

Each interface of a node connected to the Internet must have one or more IP addresses. For different reasons, an interface can have multiple IP addresses, e.g. one IPv4 and one IPv6 addresses, or multiple addresses from the same address family.

At the time of writing, the most widely used version of IP is IPv4 [92]. IPv4 addresses are 32 bits long, consisting of network and host parts. Originally, the network part length was 8, 16, or 24 bits and they were named as class A, B, or C networks, correspondingly.

The growth of the Internet has caused the IPv4 address space to be exhausted. This was discovered already in the beginning of 1990's and in 1993 Classless Inter-Domain Routing (CIDR) [97] was introduced. CIDR provides variable length network addresses, which were no longer limited to the aforementioned three classes. Although this eased the situation at that time, the problem has gotten worse after that and the address space has now been completely used. The number of networked devices is still increasing quickly, e.g. in the Internet of Things (IoT) scenarios small sensors connect to the network with their own IP addresses to enable data transfer [16, p. 2799].

Another solution to overcome the problem of IPv4 address exhaustion is Network Address Translation (NAT) [114]. NAT devices are used to hide networks behind a single (or a few), globally unique IP address. A network that connects to the Internet via a NAT device, uses internally addresses from a private address space [114, p. 3]. These private addresses are not routable in the Internet and they can be reused in multiple private networks. The NAT device maintains the required address mappings between the private address of an internal node and its own global address. It performs the required address translations for incoming and outgoing traffic. When a node behind a NAT communicates with a peer node in the Internet, the peer sees the IP address of the NAT device while the host itself is using a private address. Usually the hosts are unaware of the address translation.

Although NATs provide a simple solution for connecting more nodes to the Internet, they also have problems in certain occasions. One such problem occurs when an application uses the IP address of the host for its own purposes and includes the address in the application data. This is against the principles of layered communication because lower layer information

is directly used in upper layer data. Another problem is initiating connections from the public network to hosts behind a NAT. There are various solutions for these problems, such as different kinds of Application Level Gateways (ALG) [115]. These solutions increase the complexity of the network.

As a permanent solution for the IPv4 exhaustion problem, IPv6 was standardized in 1995 (RFC 1883 [35]) and it was later updated in 1998 (RFC 2460 [36]). The deployment of IPv6 is ongoing and it is expected that IPv6 will replace IPv4 in the future. The most apparent difference between IPv4 and IPv6 is the address length. The 128-bit IPv6 address provides a drastically larger address space than the 32-bit IPv4 address.

The IP address allocation is administered by three Regional Internet Registries (RIR): American Registry for Internet Numbers (ARIN), Reseaux IP Europeans (RIPE), and Asia Pacific Network Information Centre (APNIC). They assign portions of the IPv4 and IPv6 address spaces to the requesting service providers. Depending on the size of the network of the requesting service provider, the assigned network prefix length can vary. The prefix length determines the size of the network and the number of hosts that can be connected to the network. The network part is the same for all the nodes inside the network and the host part can be assigned within the provider's network according to their needs.

Each service provider can partition their networks into smaller subnetworks. For each of these subnetworks, the service provider can assign a part of the network prefix that it received from the administrative registry. With subnetting, the amount of broadcast traffic can be reduced in broadcast networks. Subnetting allows also connecting different types of networks to each other via specific routers. The provider can create subnetworks by allocating some bits from the host part of the assigned address prefix for that purpose (see Figure 2.2). Depending on the routing protocol used inside the network of the service provider, the subnetwork prefix part may have to be fixed size, e.g. 4 bits, or then it may be variable length.

### 2.1.3 Data packets

Data transferred in the network can be e.g. streamed audio or video or a large data file. The transmitted data is divided into smaller pieces for the actual transmission over the network. Depending on the protocol, the name of the encapsulated piece of data varies. TCP refers to encap-

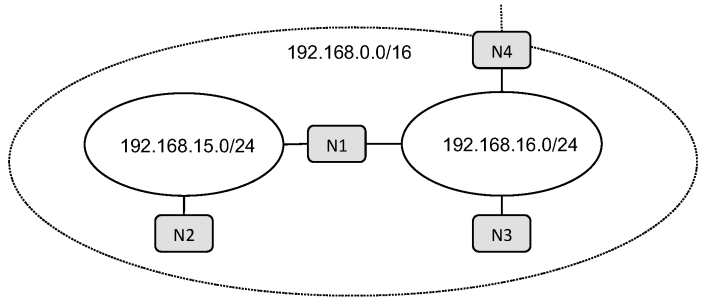



**Figure 2.2.** IP Subnetworks

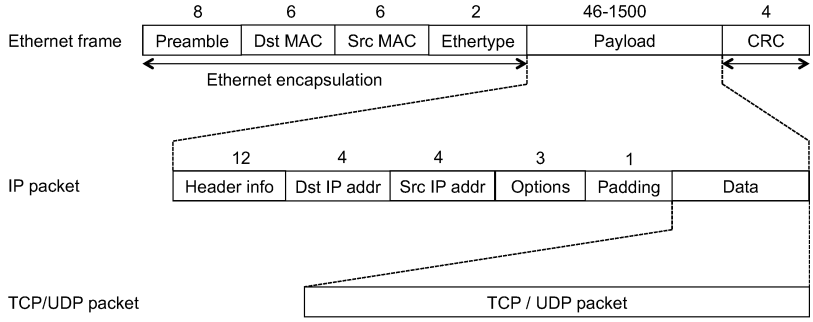



**Figure 2.3.** Ethernet and IP encapsulation

sulated data with *segment*, while at the IP layer the encapsulated data is *datagram*. In this thesis we use term *datagram* when we discuss about the data unit that is exchanged between the IP layers in different nodes, but when we refer to the single unit that is delivered through the network, we talk about *packets*.

The properties of the links between computers define the maximum size of a single data unit that can be transmitted over the link without fragmentation. This Maximum Transmission Unit (MTU) [92, p. 26] [36, p. 3], sets the upper bound for the size of the packet.

If the transmitted data does not fit into a single packet, it is divided into smaller pieces so that the packet size does not exceed the MTU value. When the packets arrive at the receiver, the original information must be restored from the received pieces.

Each upper layer data unit is encapsulated in a packet containing the required protocol headers. Figure 2.3 shows the encapsulation and naming of the encapsulated units both for Ethernet [88, p. 114] and IP [88, p. 239]. The IP header part may contain various pieces of information, required at the routers in the network or at the receiving end-host. Typically, each IP packet contains transport and IP headers. The transport header is used for transmission control above the IP layer at the end-host.

An IP header, on the other hand, contains addressing information that is used at the routers in the network for making forwarding decisions. In some cases, the transport header can be examined by middle boxes in the network, e.g. firewalls.

Each of the packets is delivered from the source to the destination node independently of each other. The routing system of the network takes care of delivering the packets to the correct receiver, but the path is not guaranteed to be the same for all the packets. The receiver, in turn, is responsible for merging the received data chunks into the original information item.

### 2.1.4 IP over local area broadcast networks

IP is said to be a Layer 3 protocol (see Figure 2.1). Below that, Layer 2 consists of protocols that are used to deliver data to other nodes in the same network, or to communicate directly to nodes in other networks using point-to-point connections. There are various Layer 2 technologies on top of which IP can be run, such as Point-to-Point (PPP) or Token ring. The most common Layer 2 technology today is Ethernet [6].

Layer 2 can further be divided into Logical Link Control (LLC) and Media Access Control (MAC) sublayers. LLC layer provides multiplexing for Layer 3 protocols over MAC sublayer. In addition, it provides flow and error control. MAC layer provides services that enable a node to communicate in a network environment where multiple nodes are connected to the same network using a shared medium, such as Ethernet. In the context of this thesis, we discuss only about broadcast networks.

Each node is connected to the network using a Network Interface Card (NIC). This card can either use wireless or wired technology. A NIC has a MAC address identifying the interface with an identifier that is typically, but not necessarily, globally unique. The MAC address has some hierarchy which usually does not reflect to the topological location of the NIC. It the most common form, a MAC address consists of an Organizationally Unique Identifier (OUI) describing the manufacturer of the card and a NIC specific part which is a unique identifier for the card from this manufacturer. MAC can also have other variations, such as locally defined addresses, but they are less common.

From the topological point of view, the MAC address space can be considered to be *flat*. However, there are proposals, such as Portland [79], where data center nodes are configured with topologically meaningful pseudo

MAC addresses.

Physically connected nodes, i.e. those connected to the same wire, communicate with each other using MAC addresses. A MAC address cannot be used directly for communication between distant nodes, residing in separate LANs. However, there are activities to broaden the Ethernet usage, e.g. using MPLS and Synchronous Digital Hierarchy (SDH) in optical networks [80].

## 2.2 Routing Scheme

In the previous section, we described the addressing scheme used in the Internet. An IP address determines the topological location of the host in the Internet. When the network delivers the packet from the source to the destination node, each router on the path has to make a forwarding decision, i.e. where to send the incoming packet if it is not destined to itself. This decision is based on the routing information configured on the router. In this section, we describe the mechanisms how the network topology information is currently exchanged between routers and how the router can build the routing tables from the received topology information.

### 2.2.1 Aggregation

The Internet is constantly expanding [31]. The number of ASes has increased roughly two to three thousands every year. It is infeasible to add a new prefix entry to all routers in the network when a new small network is connected to the Internet, if there is a possibility to combine the routing information in a logical way. Aggregating the routing information [67] saves space at a router. When two network prefixes or the beginnings of the prefixes are similar and they are routed to the same next hop router, it is enough to store only a single entry for that prefix in the routing table.

In an optimal case, networks with similar prefixes would be located topologically close to each other, allowing heavy aggregation in the network routers (see Figure 2.4). However, this aggregation has failed for various reasons [73, p. 4] [26], such as multihoming, traffic engineering, non-aggregatable address allocations, and business events, e.g. acquisitions of companies. Some of the reasons cannot be avoided. Multi-homing and load balancing are needed in certain occasions to get more reliable network services.

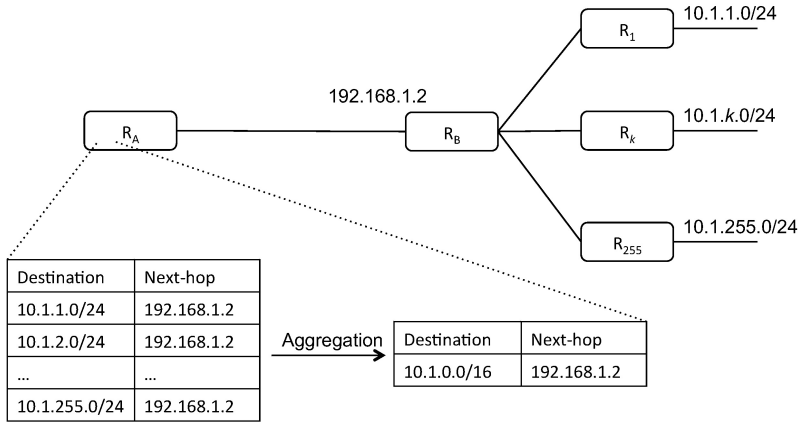



**Figure 2.4.** Route Aggregation

### 2.2.2 Routing Information Base

Each router in the network has a routing table, or a Routing Information Base (RIB), containing the necessary information to route the incoming packets to the next hop router. The network prefix-based routing information is maintained in aggregated format. The length of the prefixes vary depending on the aggregation level or the prefix length in general. The RIB is updated, when the router receives changed information related to the address prefixes or to availability of other routers in the network.

The router creates a Forwarding Information Base (FIB) based on the RIB it maintains. The FIB is used to make the actual forwarding decision. While the RIB contains all the routing information configured or learned from other routers, the FIB contains either full route information or just a subset of the routes stored in the RIB. The actual FIB structure depends on the implementation. The purpose of the FIB is to optimize the prefix matching so that the forwarding decision can be done as efficiently as possible.

### 2.2.3 Populating Routing Tables

The routing tables are populated using two different types of protocols. Interior Gateway Protocols (IGP) are used inside ASes to create routing tables for a single routing domain. Routers inside of an AS exchange information with each other to collect information about the network topology in their immediate neighborhood. Based on the collected information, each router can build a routing table describing the next hop routers for all incoming IP packets.

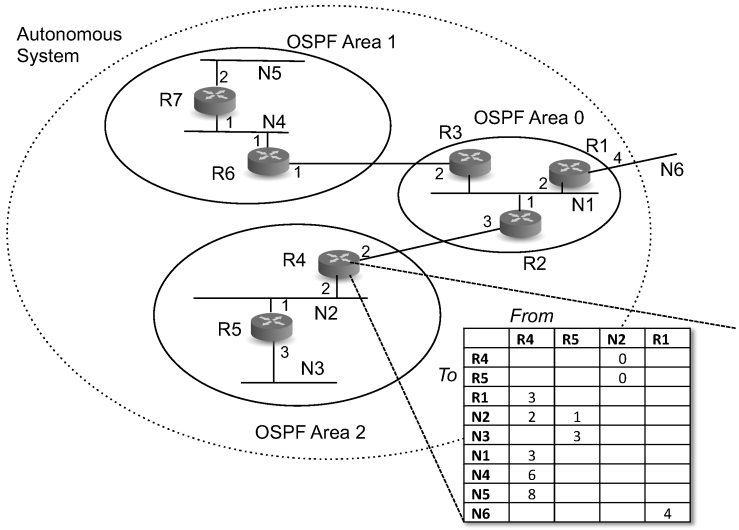



**Figure 2.5.** Open Shortest Path First (OSPF)

ASes are connected to each other and the routing information has to be exchanged between the AS border routers, i.e. routers that connect the site to other sites, in order to enable communication between the ASes. The border routers exchange only prefix level information, so that the amount of exchanged information can be minimized. The border routers use different routing protocols, called Exterior Gateway Protocols (EGP).

### 2.2.4 Interior Gateway Protocols (IGP)

IGPs operate at the intra-AS level, exchanging routing information between routers inside one autonomous system. One large IGP domain can be divided into sub-areas where IGP exchanges network level information between its subareas. However, IGPs do not operate between different AS networks.

IGPs can be divided into two different categories: distance vector and link state protocols. Distance vector protocols count directly the distance between two nodes based on the number of intermediate nodes. The routers exchange the distance information and they are not aware of the full topology information of the network. Link state protocols, on the other hand, exchange information about the active links they have and distribute also the link information that they have received from the other nodes in the network. At the end, all the nodes belonging to the same area have full knowledge of the local topology.

Open Shortest Path First (OSPF) and Intermediate System to Interme-

diate System (IS-IS) [82] are the most widely used IGP link state routing protocols. OSPF Version 2 is used for IPv4 [78] and Version 3 defines extensions for IPv6 [34]. These protocols use Dijkstra algorithm [40] to calculate the optimal path through the network. Each router collects the network topology information based on advertisements received from the neighboring routers. Once the router has enough information about the topology, it calculates the paths to different destination nodes and creates the routing tables. Each router uses the routing table only for selecting the next hop router. The next router, in turn, has its own similarly built routing information table. Both of these protocols support variable length subnetwork prefixes used in CIDR. See Figure 2.5 for an example of a simple OSPF setup with three areas and a simplified routing table of router R4 in Area 2. The routing table contains full Area 2 information and the reachability information for the other networks with the cost of using that connection.

Despite of similarities, there are some differences in these two routing protocols. For example, OSPF runs on top of IP, while IS-IS works directly on layer 2.

The Routing Information Protocol (RIP) [51] is one example of distance-vector based protocols. The first version of RIP did not support variable length subnetwork prefixes needed in CIDR, but this has been fixed in the later versions of RIP. In RIPng [70] IPv6 support has been added to the protocol. RIP is based on sending information about the network prefix and the distance from the sending node. Thus, the receiver can create forwarding information based on the distance to the destination networks.

### 2.2.5 Exterior Gateway Protocols (EGP)

IGPs described in the previous section exchange full network information between network routers. These protocols cannot be used when routing information is exchanged globally in the Internet [106, p. 3]. First, the amount of exchanged information would be huge causing severe scalability issues. Second, operators are not willing to disclose information about their policies and network structures which would happen if IGPs were used between operators. Third, separating inter-domain and intra-domain routing gives operators more flexibility to design routing in their networks. They may select the best IGP for their purpose or even use static routing if needed. Due to these reasons, an EGP exchanges only prefix-level information of the networks.

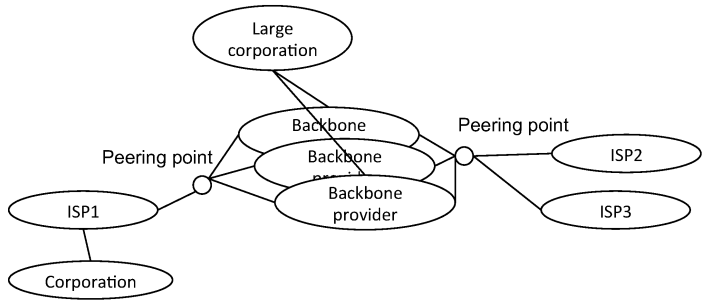



**Figure 2.6.** ASes and Border Gateway Protocol (BGP)

The first inter-domain routing protocol was the Exterior Gateway Protocol (EGP) [74] (note that the same name is nowadays used to describe all the protocols handling inter-domain routing). EGP was later replaced with the Border Gateway Protocol (BGP) [98] that fixed the limitations of EGP. For example, EGP supported only a tree-like Internet structure, while BGP assumes that the Internet is an arbitrarily interconnected set of ASes. BGP was designed to hide AS internal policies and information [106, p. 3]. It exchanges only the network prefix information between different ASes providing enough information to route packets towards the correct ASes. As Peterson and Davie mention in their book [88, p. 311], route optimization in inter-domain case is not a trivial task and inter-domain routing is more about reachability, not about optimizing the paths.

Figure 2.6 shows a typical way of connecting the networks. ISPs connect normally to the backbone providers via peering points, while larger corporations can have direct connections to one or more backbone providers.

ASes can be divided into three different categories [88, p. 311], see Figure 2.7:

*Stub AS*: This kind of AS is connected to other networks only using one border router. This type of AS is only communicating via the parent AS to the rest of the Internet and does not handle any additional traffic. In Figure 2.7, AS1 represents this type of network.

*Multihomed AS*: This AS is connected to more than one other AS. It does not carry transit traffic, but achieves more robust operation while it is not dependent on the availability of a single service provider. In Figure 2.7, AS2 shows such a network.

*Transit AS*: This AS is connected to multiple other ASes and, in addition to its own local traffic, is delivering also transit traffic between two connected ASes. AS3, AS4, and AS5 in Figure 2.7 depict these transit

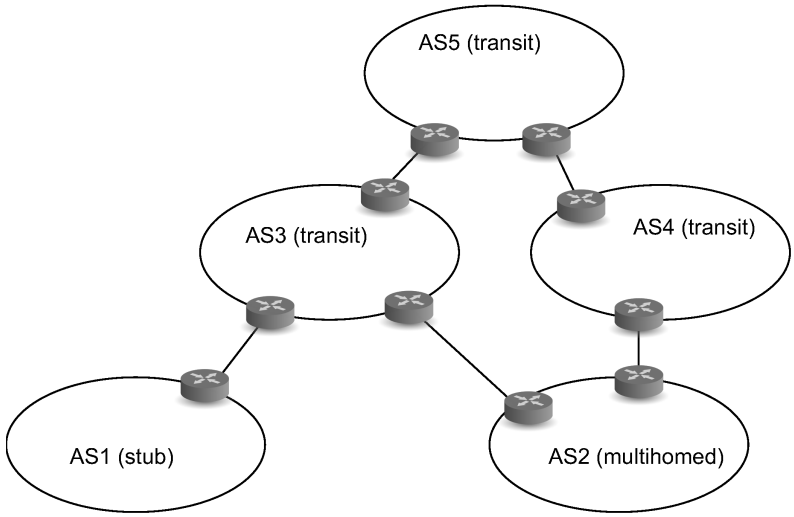



**Figure 2.7.** Different types of AS networks

ASes.

In addition to their own address prefixes, ASes advertise reachability of other networks via them. To avoid potential loops, a BGP message contains the full AS path to reach the advertised destinations. Any AS can determine from the received messages potential loops for certain prefixes and remove these entries from its routing table. As an example, if three ASes are connected to each other (AS1, AS2, and AS3), and AS1 can be used to reach network 10.0/16, it advertises this to AS2. AS2, in turn advertises this to AS3 which again sends this to AS1. As a result, this potentially creates a routing loop. With full AS path information, AS1 can notice that it is in the AS list, and not add that entry to its information base.

When an AS is connected to peer ASes with multiple border routers, the reachability information needs to be exchanged also between these border routers. However, BGP loop avoidance is based on advertising the full AS path together with the network prefixes. If BGP is used between the border routers of a single AS, there would not be any path information leading to potential loops in the network.

IGPs are used inside the AS to create the required routing tables, but they cannot be used to exchange external reachability information. IGPs have not been designed to cope with such a huge amount of network prefixes that exist outside an AS. Thus, another instance of BGP, called internal BGP (iBGP) [98], is used to exchange the reachability information internally between the border routers.

iBGP establishes full mesh connections between all the routers in the AS. Each router advertises the reachability information that they have to the peer routers. Routers do not advertise further the received routes. The need for full mesh connections cause a scalability problem when the AS network grows. However, there are other solutions, such as route reflection [20] and confederation [121], that scale better than the full mesh iBGP.

## 2.3 Packet Forwarding

IP packet forwarding is based on routing tables, described in Section 2.2. A routing table provides the required next hop forwarding information that the router uses to deliver an incoming packet towards the destination determined by the IP address in the packet. This is repeated at each router on the path from the source to the destination. This one-to-one communication is called unicast. Sometimes it is beneficial to deliver data from one source to many receivers simultaneously. This is called multicast.

### 2.3.1 Unicast

In unicast, data is sent from a single source to a single receiver. The sending host encapsulates the data to be delivered into an IP packet, and sets the destination host's IP address in the header.

At each network router, the routing table determines the next hop router for the incoming packet. The destination IP address in the packet header is matched with the routing table entries. Routing tables are sorted based on the length of the prefix and the matching always starts from the longest prefix, preferring more specific routes over the more generic ones.

There are various operations that the router can do, such as queuing data packets and Quality of Service (QoS) management. These features are essential for certain connections, such as voice over IP applications, where data needs faster processing for keeping the latency low. Such processing does not affect the actual routing and forwarding decisions, and is not considered further in this work.

The recent development in the Internet has led to a situation, where unicast is no longer always the most efficient way of delivering data. When a huge number of clients are contacting a server, that is sending e.g. live

video stream, each of the connections is set up separately between the client and the server. Thus, the server has to deliver multiple copies of the same data, each of them destined to different receivers. This may create heavy traffic load close to the sending server.

### 2.3.2 Multicast

Unlike unicast, multicast is a delivery mechanism, where one or more data sources can send data and the network replicates this data to all the receivers. Using multicast, the source does not have to send individual copies of the data packet to all the destination hosts, but it is enough that it sends one copy of the data and the network handles the multiplication of the packet closer to the receivers. This saves network resources, especially close to the sending node. The source node does not need to be aware of all the receivers of the stream.

IP multicast was introduced as an add-on to the current IP protocol. The proposed solution adds state to the network routers to enable the data delivery with a flat multicast group identifier. The multicast group does not have any network topology related hierarchy and it does not determine any location for the receiving hosts. State in the routers is used to determine the directions, where the multicast packet must be delivered and if there is a need to make multiple copies of the packet, i.e. if the router is a branching point for the data belonging to this multicast group. The delivery path from the source to all the destination hosts is called a multicast delivery tree.

A share of IP address space is allocated for multicast use and these addresses identify multicast groups. When packets are sent to a multicast group, the destination IP address is set to the group identifier instead of the actual destination host. In IPv4, the address range 224.0.0.0 - 239.255.255.255 is allocated for multicast use. In IPv6, the corresponding allocation is for addresses beginning with 0xFF.

Any-Source Multicast (ASM) [37] uses a globally unique identifier $G$ to identify a multicast group. The notation for an ASM group is $(*, G)$, where the asterisk determines that any host can send data to the multicast group. When the source node sends data packets using the group address $G$ as the destination address, the network will deliver the packets to all the clients that have indicated that they want to receive that multicast group. Because the ASM group identifier must be unique, identifier allocation must be centrally controlled.

In Source-Specific Multicast (SSM) [52], a multicast channel is identified using a combination of a group address $G$ and sender's IP address $S$. This channel identifier is denoted by $(S, G)$. An SSM address $G$ does not have to be allocated globally because the source address makes it globally unique. $G$ can be determined by the source node.

Binding the source address to the group address restricts also the sending of data. Data can only be sent from the source node having the IP address $S$. The multicast router needs to check the validity of the source address, i.e. that the packet arrives from the correct direction when verified from the routing table. In a standard router this verification is called Ingress Filtering [44] and it is normally used to block potential DoS attacks that use IP address spoofing. If this verification is not deployed, spoofed source addresses can be used to deliver data to multicast groups from fake senders.

The commercial deployment of IP multicast in the Internet has been slow. The authors of "Deployment issues of the IP multicast service and architecture" [41] discuss the potential reasons for the slow deployment, listing for example the network management and configuration issues as one reason.

The need for multicast and the lack of support for IP multicast, has created a number of other multicast proposals. For example, Peer-to-peer networks can be used to distribute data more efficiently than with unicast [21, p. 11]. Explicit Multicast (XCast) [23] is also one type of application level multicast solution, where the transmission is based on including the branching points already in the packet header in source routing style.

### 2.3.3 Source Routing

The host sending data cannot control the packet's path through the network. Usually there is no need for the sender to do it either. Packet forwarding from the source to the destination is based on the routing information on each of the routers on the path. For various reasons the path may vary even during an existing connection and two successive packets going to the same destination may take different paths through the network.

Source routing [92, p. 18] is a technique that can be used to let the sender determine the path of the packet through the network, either fully or partially. The former option is called Strict Source Record Route (SSRR) and the latter Loose Source Record Route (LSRR). Using these

options, the sender of the packet can define the routers the packet has to visit before reaching the destination.

In SSRR, the sender determines all the routers with their IP addresses, through which the packet needs to travel. This means that the full path is visible in the packet header. On the other hand, in LSRR only certain routers in the network are listed and between these routers the network routers forward the packet according to their forwarding tables. In both cases, the option field contains a list of IP addresses where the packet has to visit. The header contains a pointer to the next destination address, and the routers on the path forward the packet towards that address. When the packet reaches that router, it changes the pointer to the next destination IP address.

The problem with LSRR is that it can be used also for hacking purposes [33, p. 183]. For example, when there is a NAT device in the network hiding a private address space behind it, the LSRR can be used to reach first the NAT device from where the host in the private address space can be reached. Hence, LSRR packets are typically blocked in the Internet for security reasons.

### 2.3.4 Variations for Forwarding

IP-based packet forwarding is not optimal for all types of communication. For different reasons, data delivery may need other mechanisms, e.g. to enhance security, to provide better services for users, or to provide more efficient transmission through the network.

*Virtual Private Networks*

Virtual Private Networks (VPN) are used to connect remote LAN sites to each other over the Internet, forming a virtual network. The edge routers of the connected remote networks create a connection between them. Data packets destined to the other network are encapsulated in a tunnel and they are delivered through the Internet to the edge router of the other network. The nodes inside these LANs see the created virtual network as it was a physical network and they are not aware of the connection over the Internet. Tunnels over the Internet are typically encrypted to protect the information transmitted between the customer sites.

VPNs can be implemented at various layers, for example on Layers 2 [15] and 3 [30]. Also Layer 1 VPN [119] has been specified in the Internet Engineering Task Force (IETF). The common idea is to carry the

data packet from the originating site to the destination site without modifications.

Depending on the VPN implementation, a packet that is delivered over the tunnel can contain different segments of the original packet. Layer 2 VPNs carry the full Ethernet packet inside the VPN tunnel. The sites belong to the same Ethernet segment and the Ethernet header is also transmitted through the tunnel. On the other hand, Layer 3 VPNs carry IP packets without the Ethernet header.

*Multiprotocol Label Switching (MPLS)*

MPLS was originally designed to achieve faster packet forwarding through the network. However, the development in technology has reduced the performance advantage of label switching compared to plain IP forwarding. MPLS is still used, but the purpose is now different. In their book Peterson and Davie [88, p. 340] give alternative reasons for using MPLS. The list includes enabling forwarding on nodes that do not have IP forwarding capability, allowing to forward traffic along a predetermined path, including support for Traffic Engineering (TE), and also enabling support for different types of VPNs.

MPLS packet forwarding uses predetermined paths through the network and these paths are named with short labels. Figure 2.8 shows an MPLS network with Provider Edge routers (PE) and Label Switch Routers (LSR) inside the MPLS network. PE routers are further connected to the Customer Edge routers (CE) outside the MPLS network.

The path for a connection through the MPLS network is computed either in a distributed or centralized manner. The corresponding forwarding labels identifying the path are created at each of the routers. Each router has a label pair and a next hop router assigned for each stream. When a packet arrives at an LSR, the label in the packet is used as the index value to search for the corresponding new label and the next hop router. The old label is switched to the new one and the packet is sent further. Label switching allows local selection of labels at the routers for each of the connections and eliminates the need to coordinate the assigned labels between the routers to avoid label collisions.

A typical use case for MPLS is to implement a VPN. RFC 4364 [103] defines a mechanism to provide VPN connections over the IP backbone using MPLS. Data traffic is tunneled over the IP network using MPLS. The required routing information for the customer networks is exchanged

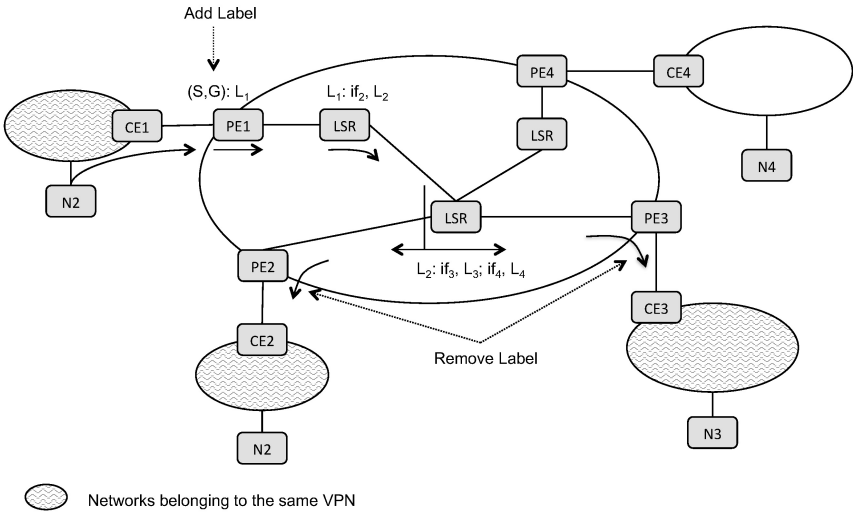



**Figure 2.8.** MPLS network

between the PE routers using BGP.

As shown in Fig. 2.8, the incoming packet is encapsulated at the incoming router $PE_1$ (Provider Edge router) and sent to the MPLS network. Using MPLS label switching, the packet is forwarded towards the destination PE routers. At LSRx, the packet is copied to two destinations, i.e. MPLS has multicast support enabled. At the destination edge nodes $PE_2$ and $PE_3$ the encapsulation is removed and the packet is delivered to the corresponding customer networks via the CE routers ($CE_2$ and $CE_3$).

In MPLS, the routing decision is made at the incoming PE router which is different compared to IP routing where the routing decision is made at each router on the path. The PE decides which predetermined path the packet takes through the MPLS network.

The path setup through the MPLS network can be done with various protocols. When the Label Distribution Protocol (LDP) [14] is used, the routers themselves assign labels for connections based on the IGP information they have. These label bindings are communicated to the neighbor routers. Using this method, LSRs create label switched paths (LSP) through the network.

The Resource Reservation Protocol (RSVP) [24] was originally designed to support Integrated Services (IntServ) architecture in the Internet. The purpose is to provide QoS for the end hosts by allowing them to request the needed resources. A traffic engineering extension for the RSVP protocol has also been defined (RSVP-TE) [17] to handle the specific requirements set for the MPLS protocol [18]. RSVP-TE protocol is used to setup

LSPs between the MPLS edge routers.

Both OSFP and IS-IS have been extended to support Traffic engineering (OSPF-TE [62], IS-IS-TE [69]). These extensions enable advertising different link characteristics, such as bandwidth. When an MPLS LSP is created, the characteristics of the routers can be taken into account. The LSP can be created either at the ingress node based on the received network topology information, or at a separate entity, such as a Path Computation Element (PCE) [43] which is described in Section 2.3.5.

RFC 4364 [103] defines that a VPN network uses unicast connections between customer sites. When multiple customer sites are connected using a VPN, multicast can be used to enable more efficient data transmission. Multicast support for MPLS is defined in RFC6513 [102].

### 2.3.5 Path Computation Element

The Path Computation Element (PCE) [43] collects network topology information, calculates a path using the received information and transmits the required path information to the network routers. The topology information can be collected using e.g. OSPF or IS-IS.

The PCE allows the route calculation to use network constraints in an efficient way. It can be implemented as a centralized function residing in a single node, or it can be distributed in the network, where one or more nodes can take part in the path calculation process. The element can be located either on one of the network nodes, or outside at an external node. The specification allows various ways to implement the PCE functionality.

A PCE supports constraint-based path forming in different kinds of network environments and it is typically used in MPLS and Generalized Multi-Protocol Label Switching (GMPLS) [71] networks. The purpose of PCE is not to be a global routing solution. It is used to provide routing for a specified area in the network, e.g. inside a single domain.

## 2.4 Summary

IP has been around for decades and provides a well defined way to communicate between nodes. However, networking environment has changed drastically since the introduction of IP. For example, users want to stream lots of videos and live streams through the network and companies want to connect distinct sites to a single corporate network. IP networks have

been enhanced with different kinds of features, such as MPLS and VPNs, to better support the requirements of the customers.

There has not yet appeared any real alternative to replace the current IP-based networking. One reason is that the Internet has grown into a huge, global network and replacing the basic protocols in the whole Internet would be an enormous operation. On the other hand, patching the existing protocols with new features does not necessarily provide the best performance and is not sustainable in the long run. This has initiated various research projects and resulted in alternative proposals for addressing and routing.

# 3. Alternative Routing Solutions

IP is the sole protocol for addressing and routing in the Internet. IP has limitations and problems mostly due to the fact that Internet has grown rapidly during the past three decades. In this chapter, we shortly describe the existing problems related to IP and forwarding with it. After that, we introduce some of the most relevant solutions that have been proposed to fix certain problems described. This thesis focuses on packet forwarding in fixed networks. For completeness, we introduce some ad-hoc related protocols which have some similarities to the forwarding mechanism that we present in Chapter 4.

## 3.1 Problems with the current Internet Architecture

The problems in the Internet can be considered, at least to some extent, to be caused by the old design. IP was originally designed in a relatively small environment with only a few nodes. During the past three decades, the Internet has expanded rapidly. IP is scalable, but still the number of different types of networks, e.g. mobile and ad-hoc networks, bring new kinds of challenges to handle packet delivery.

In this section, we present some of the problems with the current Internet. This is not a complete list, but describes the most important problems at the time of writing.

### 3.1.1 Routing Table Growth

The routing table in a router maintains information about the destination addresses based on their prefixes. With route aggregation each router does not have to store all the networks in its routing table, but they are stored based on the common prefix if they can be reached via the same outgoing interface from the router. This is possible because IP addresses are

hierarchical. With flat addresses that have no structure, the aggregation would not be possible and destination address-based routing information would be needed in each router.

If route aggregation would work optimally, the sizes of routing tables could be kept at reasonable levels. As described in Section 2.2.1, de-aggregation is growing the Default Free Zone (DFZ) routing tables [54]. During the past fifteen years, the number of BGP RIB entries has increased from 50 000 to 500 000. The de-aggregation has happened because e.g. customers have changed their ISPs, peerings have changed, and the scarce IP address resources have been assigned to customers independent of their topological locations.

The growth of BGP RIB leads to different kinds of problems [73, p. 4]. For example, it effects the requirements for RIB and FIB memory sizes as well as increases the number of BGP UPDATE messages injected into the DFZ, causing the need for additional processing at the routers.

### 3.1.2 Malicious Operations in the Network

The Internet is not a safe place for a node that is not protected properly. Nodes can become victims of information stealing or they can be hijacked and used for larger attacks in the network without the user of the computer noticing it.

Malicious software is a huge problem for users. From the end-user perspective, malicious software can steal e.g. bank account information, lock the computer and blackmail money for opening the computer, or use the victim host to send spam e-mail without the user knowing it. An attacker can distribute such malicious software e.g. as hidden in some other software that the user installs or as an e-mail attachment. One way to get a host infected is to use open ports in a host and install the software using a vulnerability in the operating system. The attacker can do this by running port scanning in the network and trying to find hosts with certain vulnerable ports open. With IP, this is possible because the attacker can guess or scan addresses and directly contact hosts. These attacks have to be prevented e.g. using firewalls.

Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) attacks [49] can be launched against any node reachable in the Internet. The aim of a DoS attack is to consume resources on the victim nodes to prevent the legitimate use of the provided service. In a simple case, an attacking host can initialize connections with the victim host so that it cannot respond to

any other requests. More powerful attacks can be launched using many attacking hosts simultaneously as is done in DDoS attack.

In DDoS, an attacker takes control over multiple hosts connected to the Internet before the actual attack. It can do this, for example, by infecting a large number of hosts that are not properly protected as mentioned when we discussed about malicious software. Once the distribution of attacking software is wide enough, the attack is initialized towards the victim server from all the infected hosts.

The presented problems give only a hint of the security problems that exist in the network. We are not going deep into the security area; solving security problems require work on different layers, e.g. starting from the link level forwarding and going up to the users behavior. These problems are examples of attacks that can take advantage of IP forwarding, where the packet can be delivered to the destination host without any request by the destination.

### 3.1.3 Energy Efficiency

One of the key problems that must be solved when future network technologies are designed, is the energy consumption [56, p. 2]. The energy consumption of a router increases nearly linearly when the amount of traffic increases [112, p. 334]. The development in hardware energy efficiency is not enough to cope with the increase in energy consumption due to growing traffic [124]. This means that the situation is becoming worse in the future. The energy consumption increases directly the operational expense of the operator and it has also side effects, such as increased heat dissipation. The per-packet, or per-byte, forwarding cost in terms of energy becomes even more important in the future.

Energy efficiency is also critical for some constrained devices connected to the Internet. These devices may have only a small battery and they should be able to run very long times without service. Especially when such devices act also as routers e.g. in sensor networks, the used routing and forwarding mechanism may have a great impact on energy consumption.

## 3.2 Routing Mechanisms Based on Different Addressing Schemes

Researchers have been challenging the IP-based routing and forwarding during the years, but no solution has emerged that could have changed its dominant role. In the following sections, we discuss various solutions that have been proposed to replace or supplement the IP-based routing and forwarding.

### 3.2.1 Hierarchical address structures

Hierarchical addressing is based on creating a structure for the address that describes the topological location of the node having the address. The address can be divided into several segments, each part reflecting different level of hierarchy. Fixed line telephone numbers are one example of hierarchical addresses [55] that were used in the past to establish calls between the caller and the callee.

IP addressing provides a hierarchical addressing in the Internet. An IP address can be divided into segments, describing the topological location of the node with different granularity. The address structure reflects the structure of the Internet which is divided into networks and subnetworks.

The IP routing system uses the address hierarchy for creating the routing information in the network. The routing information is distributed in the network nodes so that when a packet arrives, the router can determine the next hop router for the packet from some part of the address prefix.

### 3.2.2 Non-hierarchical addressing

Flat addresses, such as MAC addresses [6], do not have any hierarchy that reflects the topological location of the address in the Internet. They may still have some internal structure for other purposes. These kinds of addresses are not easy to use for routing packets in global network because there is no possibility for route aggregation as is with hierarchical addresses. In a naive solution, each router in the Internet would need to store all the destination node addresses and corresponding next hop information for each of the address. This is clearly not a scalable solution; it would be very hard to exchange such routing information between the routers in the network. However, there are various routing system proposals that are based on flat addresses, or flat labels, such as: A Scalable

Ethernet Architecture for Large Enterprises (SEATTLE) [64], Routing on Flat Labels (ROFL) [28], Metro Ethernet [46], and Accountable Internet Protocol(AIP) [13].

In SEATTLE, packet forwarding is based on end-host MAC addresses. SEATTLE uses a link-state protocol to enable switch level information exchange between the switches in the network. SEATTLE does not distribute end-host information in the link-state protocol, but enables one-hop Distributed Hash Table (DHT) solution on the network level. The switches in the network form a DHT network. SEATTLE forwards packets based on resolving first the destination switch from the $(IP, MAC)$ DHT mapping table, and then further the $(MAC, location)$. The solution is aimed to mitigate the managing and configuring of large networks.

In Routing on Flat Labels (ROFL), all nodes are identified with an identifier. All routers belong to a Chord [118] DHT. Instead of maintaining any location information, the routers are aware of the predecessors and successors in the DHT and packets are routed using source routing with router identities. When a packet is sent from an end-host towards another host with a certain identifier, the DHT system uses the source routing information that they have for the identifiers and delivers the packet towards the destination. Intra- and inter-domain routing are separated from each other; in the intra-domain routing the source routing is done, as mentioned, with the router identifiers while in the inter-domain routing the source routing is based on AS-level identifiers.

The Accountable Internet Protocol (AIP) uses Accountability Domains (AD) for identifying domains or subdomains. These ADs are added in front of the End-host Identifier (EID) to form the routing part of the address. ADs are flat in general but may contain some hierarchy internal to an operator. The routing in the network is based on delivering the packet to the next hop network based on the next AD in the header.

Metro Ethernet was designed to extend the simplicity of Ethernet transmission to wider area networks. Metro Ethernet, later known as Carrier Ethernet, provides standardized services such as QoS for the customers. Customers can, for example, connect their remote sites together on Ethernet level using the services provided by Carrier Ethernet provider.

IP multicast can be considered to be a flat address routing and forwarding mechanism. As described in 2.3.2, an SSM source node selects an SSM address $G$ which is a locally unique identifier for the “channel”. Further, the full channel address is formed by combining this SSM address with

the IP address of the source node $S$ and is given as $(S, G)$. This channel does not have any location information for the receivers. Each router participating in packet forwarding needs to have a piece of state for each of the channels passing it. Thus, this introduces a potentially huge number of mappings in the routers if IP multicast is widely deployed.

### 3.2.3 Identity and Location Split

One of the problems with IP is the tight binding between the topological location and the node identity [107]. Removing this binding simplifies some operations. For example, keeping connections up during node movement becomes trivial. The binding prevents, or at least makes it harder, to use other than IP for either node identification or for routing and forwarding. Removing the binding would allow us to use other mechanisms for forwarding and potentially also to use alternative forwarding solutions in different parts of the network. However, it creates a need for a mapping from the node identity to its current location and potentially other mechanisms to create the forwarding information for a packet.

In this section, we describe a couple of proposals utilizing the identity and location split for managing routing and forwarding. These solutions are usually proposing a new identifier layer above IP, hiding the location information from the upper layers and allowing different kinds of forwarding solutions under it.

When IPv6 was designed, the "Global, Site, and End-system address elements (GSE)" proposal [81] was one attempt to re-organize the address into location and identity parts. In the GSE proposal, the topological routing part, Routing Goop, was used to deliver the packet through the network and it was rewritten in the network routers when necessary. The unchanged identity part of the address was used only for identifying the node.

Since the GSE proposal, various solutions for splitting the location and identity information from each other have been proposed, such as Internet Indirection Infrastructure (i3), Host Identity Protocol (HIP), and TRIAD [32]. They all use IP addressing only for packet delivery and the nodes are identified with other identifiers.

i3 [117] defines a new abstraction for data identifiers. Each data item is identified with an identifier $id$, and the delivered data is represented with a pair $(id, data)$, where data is the IP payload. The receiver sends a $trigger$ with $(id, addr)$ identifying the data to be received with the $id$. The

address of the receiver is sent in the $addr$ field and it is the IP address - port pair.

i3 is an overlay network, operating on top of IP. The system consists of senders, receivers and a set of servers that store triggers and further forward packets using IP to the receivers when a match is found. Data packets are never stored in the servers, but they are only forwarded to the receivers. The system supports mobility, multicast, and anycast. The servers maintain only triggers and handle the required operations for data delivery. These operations are hidden from the sender and the receiver.

HIP [77] is based on identifying the host with a cryptographic Host Identifier (HI). Applications see only the HI of the target host. This HI is dynamically mapped to the IP address of the destination node in a shim layer between the transport and network layers. The mapping between HI and IP address can easily be changed because the connection on the transport layer is bound to HIs instead of IP addresses.

HIP provides security with cryptographic HIs that can be used to identify the hosts securely. HIP specifies also usage of Encapsulating Security Payload (ESP) [63] [58] to encrypt the transmitted data.

TRIAD is a NAT-like solution extending the standard NAT functionality and introducing a relay agent concept for the node that is making address translations. TRIAD assumes that the leaf networks have their own address realms. The relay agents create the needed mappings when hosts in the different realms initiate a communication.

The common thing for the presented identity and location split proposals is the independence from the forwarding mechanism. Although i3 and HIP use IP as the forwarding technique, it is obvious that IP forwarding can be replaced with other forwarding mechanisms.

Information Centric Networking (ICN) research takes an alternative approach. Instead of identifying and contacting a node, information is identified with an identifier. Users request $data$ from the network instead of connecting directly to a $node$. Various projects, such as A Data-Oriented (and Beyond) Network Architecture (DONA) [66], Content Centric Networking (CCNx) [2], Publish-Subscribe Internet Routing Paradigm (PSIRP) [3], Publish Subscribe Internet Technology (PURSUIT) [4], and Scalable and Adaptive Internet Solutions (SAIL) [5], have approached the problem from slightly different viewpoints. Similarly to the presented identity/location split solutions, data forwarding does not have to be IP-

based.

## 3.3 Maintaining Routing Information

In Section 2.2, we described IP routing. IP requires that the routing information exchanged with IGP and EGP is stored in the network routers. The routing information is stored in the router in certain data structures. Source routing added some routing information in the packet header. Still, IP source routing requires that the IP routing information is still on the routers. Source routing adds just some control for selecting the path through the network.

In some network environments, it is not necessarily feasible to store the routing information in the forwarding nodes. For example in ad-hoc networks, it may happen that the configuration of the network changes fast and routing protocols designed for fixed networks cannot provide fast enough exchange of routing information between the routers. Thus, alternative solutions are required to maintain the routing information. While IP is maintaining all information in the routers, changing this would mean that some, or all, routing information is stored somewhere else than in the routers.

Bloom filters have been proposed to be used in various network nodes to store partial routing information in a compact data structure. In this section, we describe some of the proposals and how Bloom filters are used in them.

### 3.3.1 Bloom Filters

Bloom filters are probably the best known probabilistic data structure. Howard Bloom introduced them in 1970 [22] and they have been used in various purposes since their introduction.

Bloom filters are used to store information about elements of a set in a fixed size, space efficient filter. The filter supports adding items to it and later verifying if a data item has been inserted or not, i.e. if the data item belongs to the set. Due to the probabilistic nature of the filter, there is some (controllable amount of) uncertainty in the query result.

*A Bloom Filter: Inserting and Verifying*

A Bloom filter is an $m$-bit string that is initialized to 0. Each item from a data set is added to the filter by hashing the inserted data item with

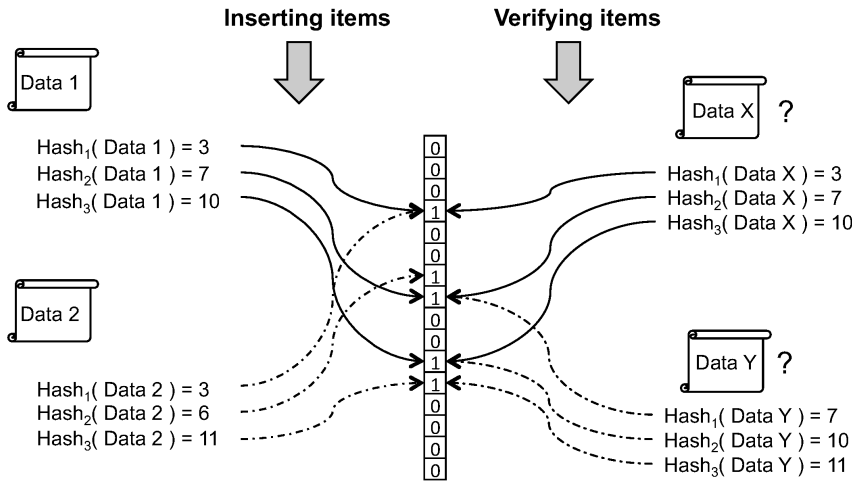



**Figure 3.1.** Example use of Bloom Filters ($m$=16, $k$=3)

$k$ different hash functions. The selection of hash functions depends on the implementation, but once the hash functions have been selected, the same set of functions is used during the lifetime of the filter. Each of these hash operations results in an integer value between zero and $m$-1. The resulting $k$ values are used as indexes to the Bloom filter bit string. The corresponding bits in the Bloom filter are set to 1 (see Figure 3.1, inserting items *Data 1* and *Data 2*). If an indexed bit in the filter is already 1 due to some previously added data item, it is left unmodified.

Once all the items of a set have been inserted in the filter, it can be used to verify if an arbitrary data item has been inserted in it or not. The verification process starts with calculating the $k$ different index values using the hash functions. The corresponding $k$ values are verified from the filter and if all of them are set to 1 (see Figure 3.1, *Data X* verification) the verification process returns a *positive* answer. In this case, we assume that the data item belongs to the set. In the figure, we see that Data X is actually the same as Data 1. However, if any of the verified bits is 0, the verification returns *negative* answer and the data item is not a member of the set.

In general, there can be two kinds of false answers as a result of a verification. The verification could result in a positive answer even though the data item has not been inserted in the filter. This means that all the bit locations are 1 but those were generated by inserting two or more different data items in the filter. This is called as a *false positive* answer. *False negative*, in turn, occurs when the verification results in a negative answer even if the data item has been inserted in the filter. In Bloom filter

this would mean that there is 0 in some bit position that should be 1.

We can easily see from the Bloom filter insertion and verification operation that a Bloom filter can never return a false negative answer. If the insertion and verification has been done correctly, all the bits of an inserted item are 1 and verification of the same item must return positive answer. However, false positives are possible as we can see from the Figure 3.1 when verifying data item $Y$. As a result of inserting Data 1 and 2, the $k$ index bits calculated from Data Y are all set, providing a wrong answer.

Due to the possible false positives, we can say that a data item belongs to a set with some probability if the verification returns a positive answer. This probability depends on many parameters, such as size of the filter $m$, the number of hash operations $k$ per item, and the number of items that has been added to the filter. Roughly, the more bits we set to 1 in the filter, the greater the probability for false positives.

The false positive probability can be calculated using Equation 3.1 [25, p. 2], with size $m$, number of hash functions $k$, and number of items inserted in the filter $n$. Increasing the size of the filter provides better results. However, the implementation environment may set limits to the size of the filter. We can also see from the equation that increasing the number of elements inserted the filter will increase the false positive probability. In addition to size of the filter, we can affect the $k$ value. The used number of hash functions can be optimized to achieve the best result; it cannot be too small, but also too large value will increase the false positive probability. Table 3.1 shows the calculated false positive rates for some $k$ and $n$ values, when $m$ = 256. A larger k-value provides better performance when the number of inserted items is low in the filter. However, when the number of items increases, smaller k-values perform better.

$$P = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \approx (1 - e^{-kn/m})^{k} \tag{3.1}$$

*Bloom Filter Enhancements*

Since the introduction of Bloom filters, several enhancements have been proposed for it resulting in many variants [120, p. 136]. Enhancements have been proposed for example to allow the item removal from the filter by using counters instead of single bits for one bit position in the filter [42]. Another example is to reduce the number of required hash functions without increase in false positive ratio [65]. The authors utilize

| n \ k | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|
| 5 | 0.0000319 | 0.0000070 | 0.0000018 | 0.0000006 | 0.0000002 |
| 10 | 0.0004379 | 0.0001758 | 0.0000832 | 0.0000449 | 0.0000269 |
| 15 | 0.0019057 | 0.0010563 | 0.0006782 | 0.0004881 | 0.0003846 |
| 20 | 0.0051883 | 0.0035357 | 0.0027462 | 0.0023536 | 0.0021761 |
| 25 | 0.0109340 | 0.0086047 | 0.0076010 | 0.0073024 | 0.0074656 |
| 30 | 0.0196105 | 0.0171418 | 0.0165792 | 0.0172076 | 0.0187707 |
| 35 | 0.0314874 | 0.0297786 | 0.0307439 | 0.0336442 | 0.0382642 |
| 40 | 0.0466482 | 0.0468451 | 0.0507120 | 0.0575365 | 0.0671627 |

**Table 3.1.** Examples of Bloom filter false probability rates, m=256

hash function simulation where new hash functions can be simulated by letting $g_i(x) = h_1(x) + ih_2(x)$, where $i = 0..(k-1)$, giving k different hash functions.

The nature of the original Bloom filter makes it impossible to remove items from the filter after they have been inserted. When a specific bit has been set to 1, it remains 1 independent of the number of inserted items that have the same bit index set to 1. Thus, it is impossible to clear a single bit during removal because more than one inserted element can have caused that particular bit to be set to 1.

In this section, we describe only one Bloom filter variant, namely Counting Bloom filters [42]. We use them later to remove items from a filter. Counting Bloom filters associate a counter with each of the bits in the Bloom filter. When an item is inserted in the filter, the corresponding counter indexed by the calculated hash value is incremented by one instead of setting the Bloom filter bit to 1. When the Bloom filter is used, each bit location where the corresponding counter value is greater than zero is considered to be 1. Removing items from the filter becomes trivial; for each of the removed item, all the counters indexed with the $k$ calculated values are decreased by one. When a single counter decreases to zero, also the corresponding bit in the Bloom filter is considered to be 0. While a standard Bloom filter is compact and space efficient, inserting counters increase the size of the filter. In “Theory and practice of Bloom filters for distributed systems” [120, p. 136], the authors discuss about the required size of the counter.

### 3.3.2 Alternative Routing Proposals

During the years, alternative proposals for routing and forwarding have been presented. The proposals are solving different kinds of problems, such as fixing routing and forwarding in challenging environments, e.g. in ad-hoc networks, or enhancing IP-based forwarding with new inventions. In this section, we list only few relevant routing and forwarding proposals that are based on Bloom filters and take a look at the ways they have been used.

Longest Prefix Matching is used in network routers to find the correct next hop router for incoming IP packets. One proposal to enhance the lookup performance in the routers is to use Bloom filters for matching the prefixes [39]. The solution does not affect the IP traffic itself. A router uses Bloom filters locally and this does not have any relation to the operation of other routers. The probabilistic nature of this method provides faster lookup but it does not affect the actual routing decision or delivery accuracy.

Free Riding Multicast (FRM) [96] uses Bloom filters for two purposes. First, a Bloom filter is used to maintain the multicast group (G) information at the AS network. Second, the AS-level multicast tree is included in a Bloom filter, located in a shim header of a multicast packet.

Each AS collects the multicast groups for which there are receivers in the AS network. These groups are encoded in a Bloom filter which is transmitted to other ASes using BGP. The AS where the multicast source is located, determines the destination AS networks using the received Bloom filters. Thus, an AS-level tree can be built for the multicast group T(G). Each of the ASes forming the tree is encoded in the Bloom filter placed in the packet's shim header after the IP header. Each AS verifies from a list of destination ASes, which of them have been encoded in the filter, i.e. which ones belong to the multicast tree T(G), and sends the packet to all the matching AS networks. This solution is vulnerable to attacks because the AS level information is public and anyone can create packets that are destined to the different ASes.

Buffalo [127] uses Bloom filters for determining the next hop routers for the incoming packets. The router maintains one Bloom filter per next hop router. Each of these Bloom filters contain the addresses that must be sent to that particular next hop router. When there are changes in the network topology, the Bloom filters on the routers have to be updated

accordingly. For each incoming packet the required hash functions are calculated and the results are compared to the stored Bloom filters. The next hop router is selected based on the result. This solution aims to enhance the performance of making the forwarding decision at the router, but does not change the actual routing infrastructure.

In "Exploring Efficient and Scalable Multicast Routing in Future Data Center Networks" [68], the authors define a loop free forwarding mechanism where the loop avoidance is based on the distance measurement from the source node. Each router has to maintain the distance information to all the potential source nodes, as well as the neighboring nodes' corresponding values. The packet is routed to the next hop router only if the next hop router's distance to the source node is greater than from this node.

## 3.4 Summary

In this chapter, we described alternative routing and forwarding solutions that have been proposed to replace or supplement IP. These proposals provide typically solutions for some very specific problems. However, they do not fix any of the major problems that were described in Section 3.1.

A new routing and forwarding mechanism should meet at least the following requirements: First, the new system should be simple and scalable. Second, it should be possible to deploy the new system gradually; it is impossible to replace IP with a new mechanism overnight. The new system must be able to provide advantages over IP also when deployed in some limited areas in the network and it should be easily compatible with IP at the edges so that simple conversion between these forwarding mechanisms is possible.

In Chapter 4 we present a new forwarding mechanism. It will take a different approach compared to the solutions that we presented in this chapter. The proposed solution can be gradually deployed and provide a very easy-to-configure solution in small networks.

# 4. In-packet Bloom Filter based Forwarding

In this chapter we introduce a new packet forwarding mechanism called in-packet Bloom filters (iBF) presented in Publication I. iBF is not based on any global addressing scheme but on uniquely identified links in the network. iBF forwarding uses strict source routing where the complete routing information of a packet is carried in the packet header. There is no need to maintain routing tables in the forwarding nodes. Instead, the forwarding nodes maintain only link identity information making them nearly stateless. We use Bloom filters to compress the route information into a fixed length header providing reasonable packet size. Bloom filters provide also a fast lookup of the next hop node at the forwarding nodes. While the basic solution does have some shortcomings related to security, we introduce specific enhancements in Publication III that provide a new way of calculating link identifiers in real-time while packets are entering a forwarding node.

Performance and scalability are issues that have to be solved before a new forwarding method can be deployed either in large networks or even globally. In our work presented in Publication II and Publication VI, we show that in certain network environments iBF performs better than IP in forwarding and saves resources e.g. when compared to IP multicast.

IP has been deployed globally and it is impossible to replace it overnight. In practice, gradual deployment is the only way to deploy a new forwarding mechanism. Later in the chapter, we present solutions that allow operators to take advantage of the iBF forwarding, still being connected to the legacy IP networks and iBF forwarding is transparent for the neighboring networks as we show in Publication IV and Publication V.

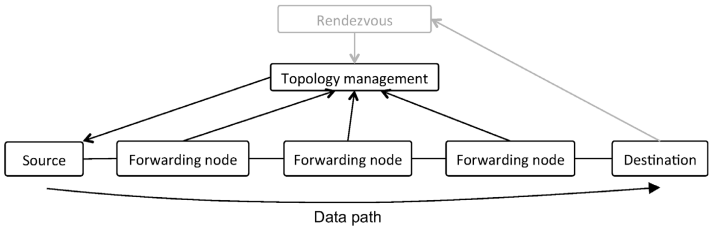



**Figure 4.1.** Network functions

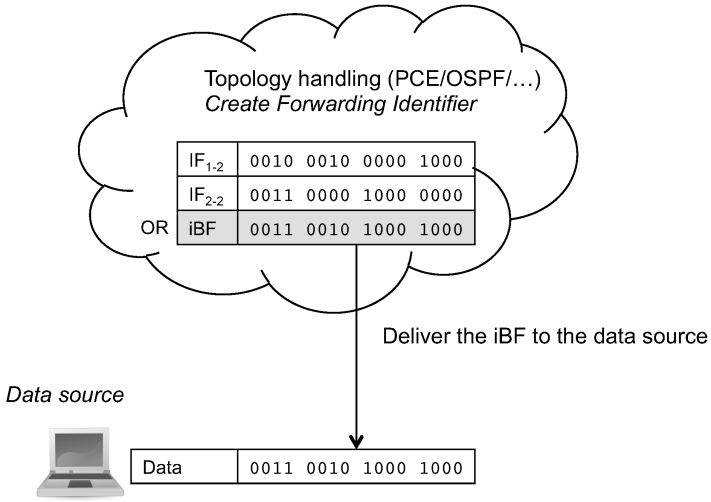



**Figure 4.2.** Creating an iBF

## 4.1 Basic in-packet Bloom Filter Forwarding

In this section, we describe the basic properties of the iBF packet forwarding as well as some simple enhancements to improve performance. This section covers the intra-domain forwarding within a relatively secure environment.

### 4.1.1 Link Identifiers and In-packet Bloom Filters

iBF is based on source routing. Instead of using node identities to describe the path of the packet, we introduce a Link Identifier (LId) that identifies a single link between two nodes. A LId is a unidirectional, $m$-bit long bit vector, containing $k$ bits set to 1 where $k$ is considerably smaller than $m$. Typical values in the current iBF implementations are $m$ = 248 and $k$ = 5. The LId resembles an item that is being added to a Bloom filter after hashing the data with $k$ hash functions.

Each LId is meaningful only locally in a forwarding node. Nodes do not need to be aware of the LIds of the peer nodes. The amount of information

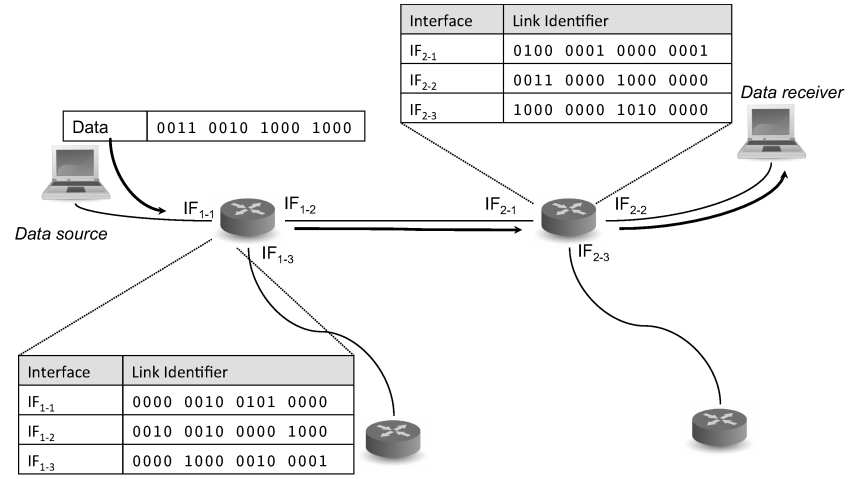



**Figure 4.3.** Forwarding with iBF

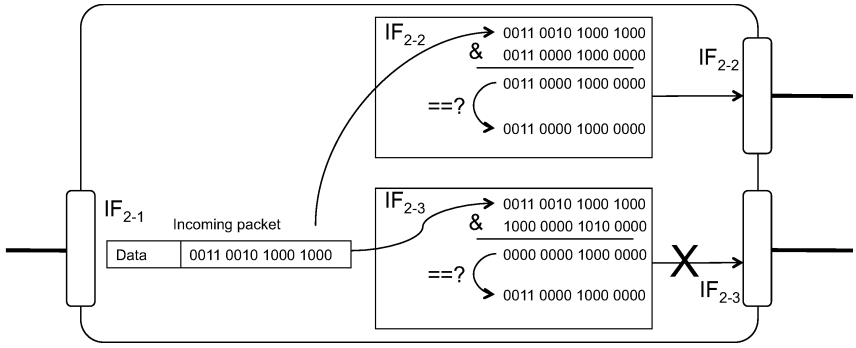



**Figure 4.4.** Verifying if a LId is included in the iBF

needed at the forwarding node can be kept minimal. The easiest way to generate LIds for the nodes' interfaces is to allow them to generate the LIds locally by themselves, e.g. by randomly selecting the $k$ bits that are 1. With the typically used $m$ and $k$ values, the probability for LId collisions is negligible.

Having two similar LIds in two interfaces in different parts of the network does not necessarily have any effect. In some rare cases this may lead to a false routing decision on the node belonging to the delivery tree and having a colliding LId on a link that does not belong to the delivery tree.

An iBF is used to store the packet delivery tree in a compact form in the packet header. The delivery tree is described by the LIds forming the tree. This set of LIds is inserted in the iBF and used as the forwarding information for the packet.

The procedure of creating an iBF is depicted in Figure 4.2 where the

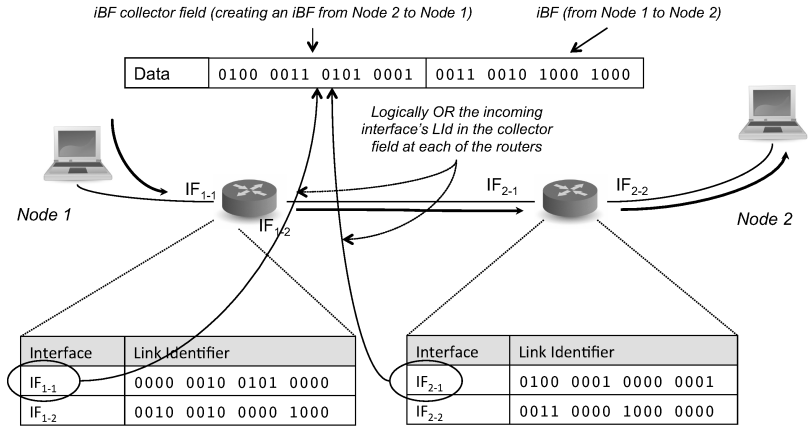



**Figure 4.5.** Collecting reverse direction iBF en-route

path through the network in Figure 4.3 is created. First a Topology Manager (TM) [1] creates a path between the data source and the data receiver. The data has to travel via two interfaces in the network, $N_{1\text{-}1}$ and $N_{2\text{-}2}$. The corresponding LIds are inserted into the iBF by bitwise applying a logical OR operation between the inserted LId and the iBF. Adding LIds into the iBF is a similar operation as inserting data items into Bloom filters. Once the iBF is created, it is handed to the data source node.

In some cases, it is not necessary to calculate the iBF in the TM, but a reverse direction iBF can be collected en-route. In Figure 4.5, the collecting process is shown. The iBF collector field is initialized to zero at Node 1. The packet is forwarded from Node 1 to Node 2 using the iBF in the packet header. At each forwarding node, the LId of the interface from where the packet arrived is added to the collector field using a logical OR operation. When the packet arrives at Node 2, the collector field contains an iBF that can be used to forward packets in the reverse direction from Node 2 to Node 1.

Unlike IP source routing, an iBF does not reveal the path of the packet. First, the LId does not tell anything about the location of the identifier and second, after the LIds have been inserted in the Bloom filter there is no possibility to extract individual LIds from it.

[1] Routing in iBF network depends on a Topology Manager that calculates routes and the corresponding forwarding iBFs. The Topology Manager will be described in Section 4.2.1

### 4.1.2 Packet Forwarding

Once the data source node has received the created iBF, it can start sending data packets with the iBF in the packet header. Forwarding nodes compare the LIds of their interfaces with the iBF in the incoming packet. If a match is found the packet is forwarded out on the corresponding interface. The verification procedure is simple (see Figure 4.4): the result of the bitwise AND operation between the LId and the iBF is compared with the LId. If they match, each of the bits in the LId has been set in the iBF, the verification results in a positive answer, and the packet is forwarded out on that link. This is similar to a standard Bloom filter verification, where we verify if a data item belongs to a set or not. In this case, the set is the LIds of the delivery tree and the item is a single LId.

Multicast forwarding is an inherent property of iBF forwarding. If two or more interface LIds of a single forwarding node are included in the iBF, the verification will return positive answer for multiple interfaces. The packet is sent out from the node on each of the matching links. There is no need to maintain any state related to multicast groups.

As discussed in Section 3.3.1, Bloom filter verification may return a false positive answer. In iBF forwarding this means that the packet is delivered out from the node on a link that was never added to the iBF. Depending on the environment, some false positives are not a problem although they generate additional traffic in the network. In Section 4.1.3, the false positive issue is discussed more closely.

iBF forwarding can be used also for sending control traffic to network nodes, for example to the forwarding nodes inside a corporate network. Control packet forwarding can be implemented using e.g. an internal LId on the nodes. The internal LId redirects a copy of the incoming packet to local processing. It is possible that multiple forwarding nodes share the same internal LId, providing a way to deliver control information to multiple forwarding nodes in the network with a single internal LId included in the iBF. The iBF is calculated normally covering the route from the source to all the final destinations and including the required internal LIds in it. Each forwarding node that has the matching control LId gets the packet for local processing. In addition, the packet is forwarded through the actual delivery tree.

We can generalize the packet forwarding decision making. Equation 4.1 shows the forwarding decision rule. When the forwarding function re-

turns zero on a particular interface, the packet is forwarded on that link. The function parameters are $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, being the set of parameters transmitted with the packet and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, being the set of parameters stored on the forwarding node and linked either to the node itself or a particular interface of that node.

$$f(\alpha, \beta) = 0 \tag{4.1}$$

The iBF packet forwarding is based on a simple function consisting of two logical operations. The function is shown in Equation 4.2. In the function, $\alpha$ is the iBF, retrieved from the packet, and $\beta$ is the LId that is associated with the an interface of the forwarding node. In practice, the XOR operation is a comparison between the AND operation result and the LId.

$$f(iBF, LId) = (iBF \wedge LId) \oplus LId \tag{4.2}$$

### 4.1.3 iBF Anomalies

iBF forwarding has some anomalies that needs to be taken care of before the solution can be efficiently used for data forwarding.

*False Positives and Forwarding Efficiency*
iBF routing decisions may result in false positives. This means that sometimes the packet is delivered over a link that does not belong to the correct delivery tree. However, false negatives are not possible and the packet is always delivered over the links intended.

Figure 4.6[2] shows the false positive ratio as well as the corresponding forwarding efficiency for multicast when simulated in a Rocketfuel AS6461 topology [9]. The false positive rate describes how many false positive routing decisions have been made compared to the total number of forwarding decisions. The forwarding efficiency, on the other hand, describes the share of packets transmitted over links that are needed to deliver the data to the correct receivers compared to the total number of packets delivered over the links. The figure shows correlation between the number of receiving nodes and the number of false positive routing decisions in the network. The more destination nodes we have, the more 1s we have in the iBF, and the more false positive routing decisions we get. The

[2]in-packet Bloom filters were initially called zFilters, but the name was later changed.

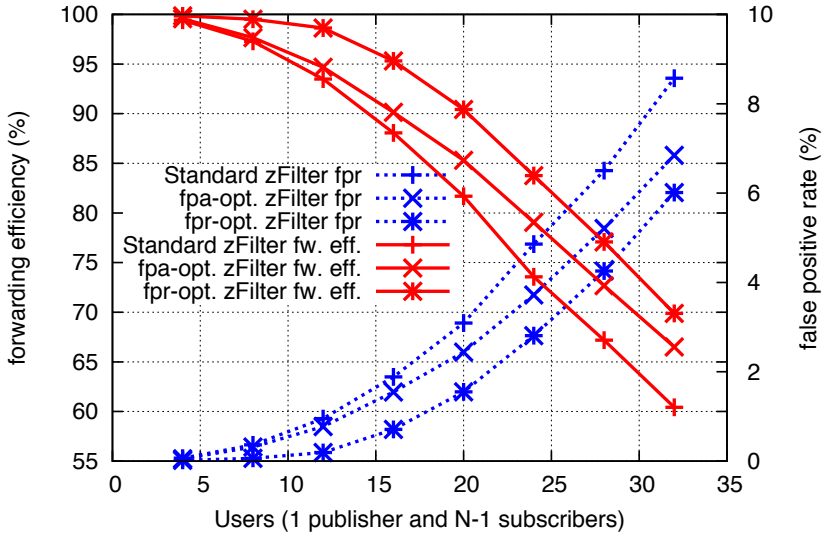



**Figure 4.6.** False positives and forwarding efficiency with standard iBF and with fpr and fpa Link Identity Tag optimization (see Section 4.1.4) using AS6461 topology. k=5 and for fpa and fpr, number of LITs n=8

same applies to the forwarding efficiency that drops when the number of clients to receive the multicast transmission increases. The different curves for false positives and forwarding efficiency refer to enhancements discussed in Section 4.1.4 and the results are discussed there, correspondingly.

*Loop Detection and Avoidance*

Loops in a network are harmful because they create lots of unnecessary traffic. With iBF, a series of false positives may cause a packet to enter the correct path again from another location. If the entering point is in the delivery tree before the place where a wrong forwarding decision was made, the packet enters a loop. A looping iBF packet will also generate additional packet delivery on the correct path to the destination. The receiver will get the same packet every time the packet loops and bandwidth is wasted also on the correct path.

Loops can be detected in different ways. One method to stop packet loops is to add a Time To Live (TTL) field in the packet. The sender sets the value to the estimated number of hops that the packet must travel and each forwarding node decreases the field value by one. When the TTL value goes to zero the packet is dropped.

Another alternative, as we present in Publication I, is to use short lived caches in forwarding nodes. These nodes maintain information about the

forwarded packets together with the interface information from where the packet arrived. If the same packet arrives again, but from another interface, the packet is considered to be looping and is dropped.

In "Forwarding anomalies in Bloom filter-based multicast" [109, p. 4], the authors propose a loop prevention mechanism that is based on saving some history information in the packet. This history information is generated by permuting the iBF in the packet when it travels through the forwarding nodes. The creation of the iBF for such a packet is not as trivial as for the basic iBF. It requires that the iBF is generated starting from the leaves of the delivery tree. After adding the LId in the iBF the iBF is permuted with a bit pattern specific for that forwarding node.

*Link Failures*

In any network environment, a connection between two forwarding nodes may be lost due to a physical or a configuration problem. To deal with this problem, the network has to be able to create alternative routes that can be used to bypass the broken part of the network.

In Publication I, we present two alternative ways to avoid broken links in the network. As the first alternative, we use a preconfigured virtual path as a backup path around a node to be avoided. This virtual path has a Virtual LId (VLId)[3], that is same as the actual LId towards the link to be avoided. The virtual path is inactive when the network works properly. If the link is broken, the virtual path is activated and the packet is delivered on that alternative path. There is no need to modify the iBF in the packet header while the VLId is same as the broken link's LId, but additional signaling is needed to activate the bypassing path.

The other alternative is to use precalculated paths around the broken link. The iBF for this fallback route is calculated using the LIds of that route. In case of link failure, the iBF of the fallback route is added to the incoming packet's iBF using the bitwise OR operation. This solution does not add signaling because there is no need to activate the fallback path. However, the number of 1s in the iBF increases, leading potentially to more false positive routing decisions.

In "Fast reroute for stateless multicast" [128], the authors further developed and simulated these reroute solutions. The VLId-based solution performed better in the simulated environments, but required more signaling. The authors came to the conclusion that a hybrid solution of these

[3]Virtual LIds will be discussed in Section 4.1.4

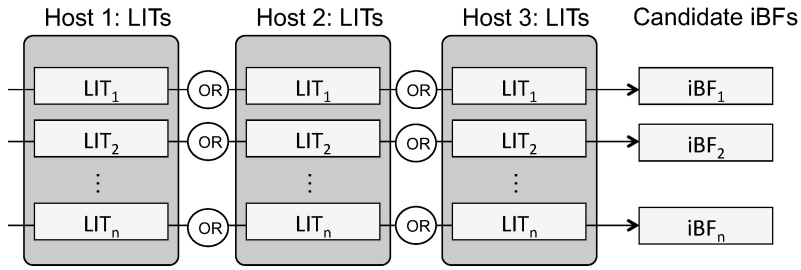



**Figure 4.7.** Creating candidate iBFs from $n$ LITs

would be the best for the operator to deploy. When the multicast trees are small, the precalculated paths is a good alternative with less signaling. However, when the multicast trees are large, the VLId based solution performs much better, thus justifying the usage of additional signaling.

### 4.1.4 Enhancing the Forwarding Efficiency

As discussed in the previous sections, there are some identified limitations in the iBF forwarding. For example, the reasons behind the false forwarding decisions due to false positives, loops, and security issues have to be carefully studied and the potential problems solved to improve e.g. scalability.

*Link Identity Tags*

The forwarding efficiency can be improved with a simple modification. Instead of one LId, we assign $n$ Link Identity Tags (LITs) to each interface, numbered from 1 to $n$ (see Figure 4.7). Each LIT is similar to a LId, with the same $m$ and $k$ values. The bit pattern of each of the LITs is different. During the iBF creation, $n$ candidate iBFs are calculated. Each candidate $\text{iBF}_\text{x}$ is calculated using the $\text{LIT}_\text{x}$ of each of the interfaces on the path. Once the candidate iBFs have been calculated, one of them is selected as the iBF to be used for the packet delivery. The iBF can be selected using different methods. We have used two alternative ways to select one of the iBFs: $Fpa$ and $Fpr$. LIT-based optimization requires that a Topology Manager calculates the iBF.

*Fpa* method tries to estimate the best performance by minimizing the number of 1s in the iBF. Thus, the number of bits set to 1 in each of the candidate iBFs is calculated and the candidate with lowest number of bits set is selected as the iBF.

*Fpr* is the lowest observed false positive rate. In this method, the iBF

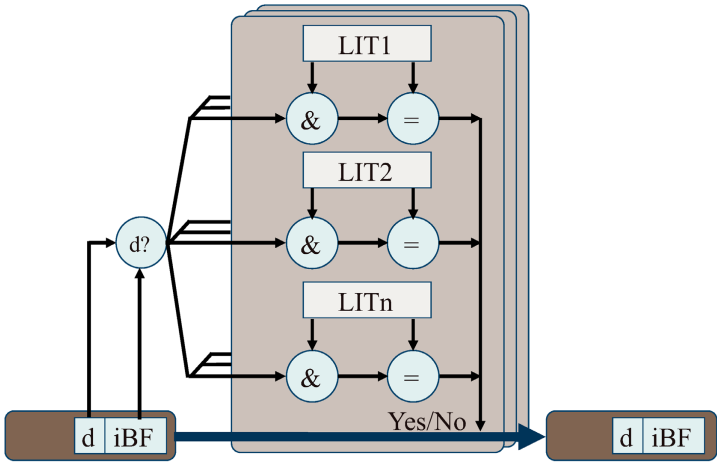



**Figure 4.8.** Selecting the correct LIT and matching it to the iBF

candidates are tested against all the routers and all their interfaces on the path to the destinations. This method provides better accuracy than the *fpa* which is based on an estimate. *Fpr* requires, however, more knowledge of the network, and consumes more resources when the iBF is created.

The LIT in the iBF is not unambiguous and the sending node must include the used index value in the packet header so that the forwarding nodes can select the correct LIT for verification process. This process is shown in Figure 4.8 where the $d$ determines the LIT index to be used in verification.

In Figure 4.6 we can see the results of simulations with plain LIds (standard zFilter in the figure) and two different optimizations with LITs (Fpa and Fpr optimized). The figure shows curves for the forwarding efficiency using solid lines and for false positive rates using dotted lines. Simulations with the single LId on each interface give us the reference value to which the optimizations can be compared. To simulate the optimizations, we used eight LITs per interface. The figure shows that with the Fpa optimization we get slightly better results than with the single LId. However, as one could expect, with Fpr optimization we get even better results because the iBF is selected based on the tested false positive matches on the path. With Fpr optimization and for 31 receivers, the false positive rate drops from 9% to 6% and forwarding efficiency goes up from 60% to 70%.

The LIT solution can be modified to support loop avoidance. Instead of using the same $d$ value on each router, the $d$ can be made variable. The Topology Manager calculates the $n$ candidate iBFs by starting from one LIT but instead of using the same value at the next hop router, the LIT

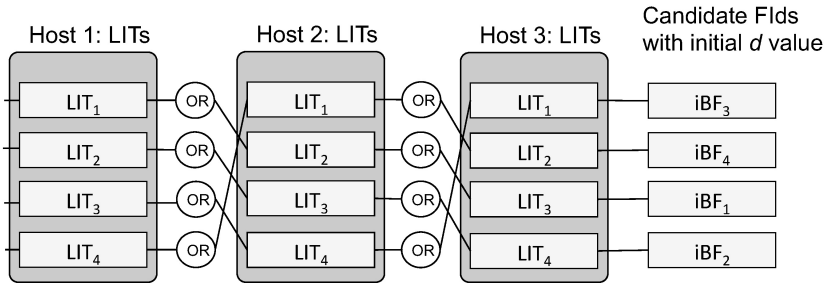



**Figure 4.9.** Creating candidate iBFs with increasing $d$ value

index value is increased by one (modulo n) for each hop (see Figure 4.9). When the packet is sent from the source node using the selected iBF, each router increases the $d$ field in the packet when they forward the packet. If the packet is in a loop, the $d$ in the packet is not the same that was used during the previous verification at the same node, unless the loop is exactly n (or its multiple) hops long. This reduces the risk that the packet is sent out again from the forwarding node, either on the false or the correct link.

*Selecting the length of a LId and $k$ for a LId*

The main disadvantage of the iBF forwarding is the forwarding due to false positive verification results. The theoretical estimation for false positive ratio was presented in Equation 3.1 with some example values in Table 3.1.

Equation 3.1 shows that $m$, $k$, and $n$ values affect the false positive ratio. When we establish the forwarding environment, we cannot affect the $n$ value but $m$ and $k$ we can select (almost) freely.

The size of the LId in the current implementations is roughly double the size of an IPv6 address, i.e. the length of a source-destination address pair in the header of an IP packet. Typically, shorter filters provide more false positives than longer filters. In Publication V, we show the forwarding efficiency for various sizes of iBFs and for different payload sizes. It is clear that a longer iBF provides better bandwidth usage when the number of receivers increases. However, typical simulations presented in this thesis use 248-bit LIds. Eight bits were reserved in the simulator for other purposes e.g. to carry the $d$ value in the packet for selecting the correct LIT on the forwarding nodes.

Calculating the optimal number of hash functions is not trivial. Equation 4.3 shows one way to get an estimation for an optimal $k$ [120, p. 133].

The optimal $k$ value depends both on the size of the filter $m$ and the number of inserted items $n$. The authors derived the equation by minimizing the false positive probability presented in Equation 3.1.

$$k_{opt} = \frac{m}{n} \ln 2 \approx \frac{9m}{13n} \tag{4.3}$$

Another way to approach the $k$ optimization is shown in Equation 4.4 [101, p. 2793]. The authors show that $k$ depends on two independent values. In the equation, we can set the desired overhead $x$ for the traffic and the average degree of network nodes $d$. Setting the desired network overhead to 5-20% and node degree to 3-6, results in $k$ values between four and seven and is in line with the authors' experimental results.

$$k_{opt} = log_2(\frac{x+1}{x}) + log_2(d-1) \tag{4.4}$$

In our simulations $k$ was usually set to five. This is close to the values calculated from the Equation 4.3 with $m$ = 248 and the number of inserted LIds being around 30 to 35. From the equation, we get 4.90 < $k$ < 5.73.

The number of items included in a filter may vary a lot depending on the size of the delivery tree. This suggests different optimal $k$ values for different iBFs. This is not practical when packet forwarding decisions must be fast and variable values for $k$ would add complexity in the forwarding nodes and increase the processing time. The network design also affects the selected parameters. For example, in some data center environments shorter iBFs can provide optimal forwarding. It is even possible to implement a false-positive-free forwarding with only 96-bit iBFs [105].

Although we used typically 248 bits as the size of the iBF, we showed in Publication V that longer iBFs can save the bandwidth even if the header size is longer. For example, with 512-bit long iBF, increasing the size of the header reduced the overall bandwidth usage when the number of receivers for the multicast group exceeded ten. The more receivers were added, the bigger the gap against shorter iBFs was. Thus, this indicates clearly that longer iBFs perform better overall. However, from the practical point of view, the packet header cannot be enlarged too much without affecting the size of the payload, thus a practical limit will be of the order of 512 bits.

*Better Performance with Virtual Trees*

As we have seen, there are different parameters that affect the false positive ratio with iBF forwarding. With any LId size or $k$ values, the false

positive ratio increases the more links are added in the iBF. In Publication I, we propose a Virtual Tree solution to improve the forwarding efficiency in the network. A virtual tree can be a full delivery tree from the source to all the destination nodes or a partial tree covering only a part of the full delivery tree. The virtual tree is treated as a single link, having a LId of its own called Virtual LId (VLId). The VLId is $m$ bits long and has $k$ bits set to 1 and is similar to any other LId for unique links.

The generated VLId is configured on each of the interfaces spanning the virtual tree beside the existing LIds and potential other VLIds. When a packet arrives at the forwarding node, the node verifies the iBF in the packet header both with the LId and with all the VLIds configured on the interface. If a match is found in any of the verifications, the packet is sent out on that interface.

In Figure 4.10 we see the setup where a part of the delivery tree is replaced with a virtual tree. We can see also that the same virtual tree can be shared by different delivery trees.

The idea of virtual trees is to decrease the amount of false routing decisions by having less 1s in the iBF. The cost of the solution is the additional state required in the corresponding forwarding nodes. The more we create virtual trees in the network, the more state we will have in the routers. Also when the number of different LIds increase in a single router, it increases the probability for false routing decision on that router. We can get better performance with virtual trees when the number of installed virtual trees is low and when a large number of edge-to-edge trees use the same virtual trees. This could be the case e.g. with a transit network, where certain edge-to-edge paths are heavily loaded.

The optimization of virtual trees is not a trivial task and requires more processing when creating the iBF and also a procedure to configure the VLIds on the corresponding interfaces. In Publication VI (Section 4.3.3) we discuss the usage of virtual trees and their performance.

## 4.2 Intra and Inter-Domain Forwarding

Protocols designed to operate in small networks typically do not scale to millions of nodes. The reason may be e.g. that a protocol exchanges lots of information which is practical only when the number of nodes is limited. In addition, the operator boundaries and potentially different policies of the operators create challenges in the global scale.

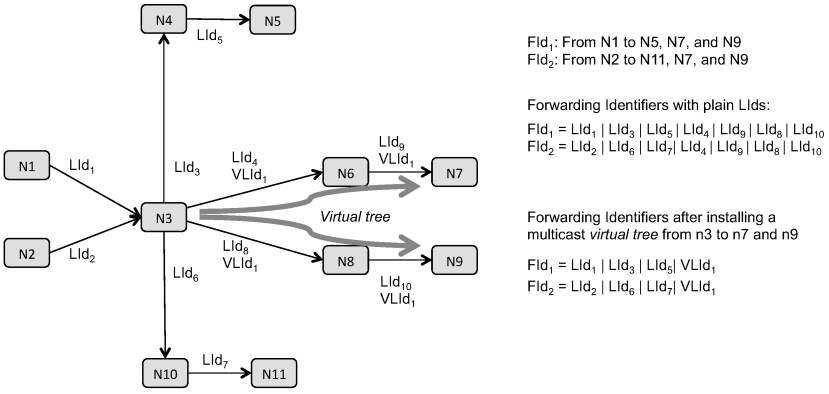



**Figure 4.10.** An example of a virtual tree

With the basic iBF, the scalability issue emerges due to the limited number of LIds that can be included in a single iBF. For large networks, operated by many different operators, the basic iBF forwarding as described earlier is not scalable enough because the paths can be long. Thus, we need a mechanism that can either manage an iBF swap in the network or store more information within the packet header. In our work, we used the former method. One way to solve the problem is to use certain relay nodes as we describe in Section 4.2.2. A relay node changes the iBF of a packet to another iBF using e.g. a stream identifier as the key for the next iBF. There are also proposals that use the latter method (e.g. [125]).

### 4.2.1 Creating a Bloom Filter for Forwarding

In this work, we have used a Topology Manager (TM) to handle the routing information. The TM is responsible for collecting and maintaining the network topology information and using this information for building the delivery trees for traffic. In addition, it is responsible for creating the iBF based on the delivery tree (see Figure 4.1).

Building a tree for iBF creation requires that the TM has enough knowledge about the network topology. In addition, it has to know the LId information for each link in the network. One alternative to collect this information is to use existing link state based protocols such as OSPF and extend them slightly to carry the required link identity information.

The TM function is typically considered to be centralized in local networks, meaning that each AS has one or more TM entities. If an OSPF-based system is used, each of the OSPF areas might have a TM entity of their own. This is, however, not a strict requirement because it is possible

to envision a system where each forwarding node can act as (part of) a distributed TM.

A PCE can also be used as a fully centralized TM solution. An IGP, e.g. OSPF, can be used to feed the PCE entity with the topology information from the network. The PCE is further responsible for creating the delivery tree based on external requests between a source a set of destination nodes.

The simple TM operation can easily be scaled to AS level where the AS operator can configure the system so that the required topology information is exchanged between the different TM nodes. However, for business and political reasons, it is not necessarily possible to exchange the required topology information between different operators. This creates a need for another inter-domain TM mechanism.

There are various alternatives to build the intra-domain delivery trees, Dijkstra being one example. The iBF is simply created by collecting the related LITs from the delivery tree information, and using bitwise OR operation, the LITs are included in the iBF. Once the iBF is calculated, it must be delivered to the actual data source node, together with some information about the data that needs to be delivered. For inter-domain forwarding, one obvious solution is to use branching points at the network edges, where the intra-domain iBF is replaced with a next one based e.g. on some stream identifier in the packet.

### 4.2.2 Bloom filter based Relay Architecture (BRA)

When there is a large number of receivers residing in various locations in the Internet, large multicast trees are needed to deliver data to all the receivers. This increases the number of LIds that are needed to be inserted in the filter lowering the forwarding efficiency.

The Bloom filter based Relay Architecture (BRA) [122, p. 25] describes a mechanism where specific relay points are established in the network. In figure 4.11, the main concept of BRA is depicted. The Rendezvous (RVS) function in the figure is used for the high level matching of the requests from the clients and services provided by the servers. It also assigns tasks to the TM to create delivery trees and corresponding iBFs.

Initially, a group of nodes in *network 2* are receiving data from the server that is residing in *network 4*. The data is delivered using a single iBF containing the whole delivery tree from the server to all the receiving nodes.

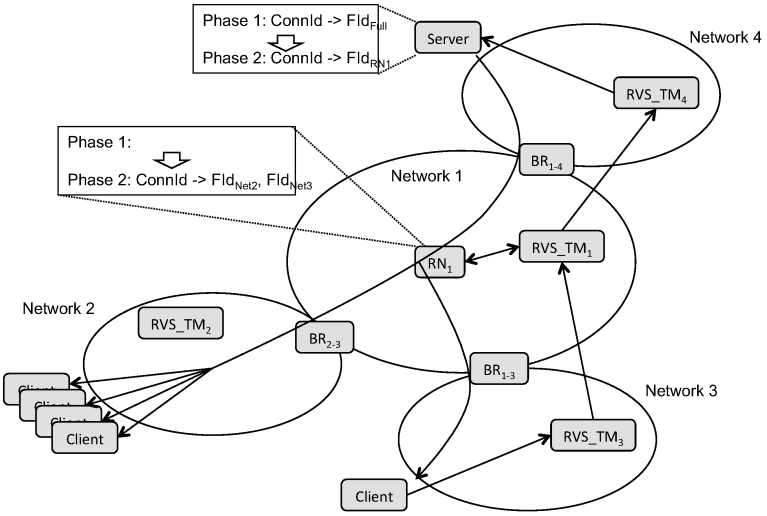



**Figure 4.11.** Bloom filter based Relay Architecture

When a new node in *network 3* sends a request to subscribe to the same multicast transmission, it contacts the local rendezvous node RVS_TM$_3$. The rendezvous node verifies the location of the source node and notices that it is not residing in the local network. Before contacting the rendezvous system in the neighboring network, the rendezvous node creates an iBF leading from the edge router BR$_{1\text{-}3}$ to the requesting client. RVS_TM$_3$ forwards the request to the rendezvous node RVS_TM$_1$ in the neighboring network.

RVS_TM$_1$ has already knowledge about the iBF for this multicast transmission because the clients in Network 2 have subscribed earlier to it. RVS_TM$_1$ calculates a new candidate for the packet forwarding including the original iBF, the received iBF from the neighboring rendezvous node, and the iBF for the path from the potential branching point to the edge router BR$_{1\text{-}3}$. In this case, RVS_TM$_1$ determines that adding all this information would clearly increase false positives and it decides to create a branching point at RN$_1$.

RVS_TM$_1$ calculates both iBFs from the RN$_1$ to BR$_{2\text{-}3}$ and BR$_{1\text{-}3}$, creating two separate iBFs from the RN$_1$ to the clients in *network 2* and to the clients in *network 3*. RN$_1$ makes a new mapping from the multicast transmission identifier to these new sub-iBFs. The mapping is based on the multicast connection identifier that is carried in data packets.

The RVS_TM$_1$ calculates a new iBF from BR$_{1\text{-}4}$ to RN$_1$ and sends it to the neighboring *network 4*. RVS_TM$_4$ calculates a new iBF for the multicast group leading from the server to RN$_1$ and sends it to the server.

The server replaces the iBF for the multicast connection identifier and continues using the new iBF for delivering the data.

In $RN_1$, the multicast connection identifier is used to find the next step iBFs, the multicast transmission packet is duplicated, and both of the copies are delivered with the new iBFs towards the clients in *network 2* and *network 3*.

## 4.3 Deployment

In this section, we present deployment related issues. First we show that it is practical to implement the iBF forwarding in hardware, providing enhanced performance compared to IP-based forwarding. After that, we show that deploying iBF in certain environments is beneficial for the operator. We also show that deployment does not affect the neighboring legacy IP networks.

### 4.3.1 IBF on a NetFPGA Programmable Router

As described in Section 4.1, iBF forwarding decisions are based on simple bitwise logical operations. The simplicity in making a forwarding decision on a forwarding node suggests that it would be possible to implement iBF forwarding directly in hardware. Hardware implementation can provide better performance than a pure software-based implementation.

We selected NetFPGA [7] programmable router as the platform for our iBF hardware implementation. NetFPGA provides an open and flexible environment for creating an own forwarding node implementation. A NetFPGA router comes with an IP reference implementation that we used to compare the performance of our iBF solution with the IP forwarding. In Publication II, we describe our NetFPGA implementation of the LIT enhanced iBF forwarding.

We used the NetFPGA's IP reference implementation as the starting point for our implementation. In Figure 4.12 the main changes to the reference implementation can be seen. On the left side, the main modules for IP packet processing in the NetFPGA reference implementation are listed. We removed the *output_port_lookup* from the reference implementation, and implemented our own *output_port_selector* in the *output_queues* module. Figure 4.13 shows the main functions that were implemented in the *output_queues*.

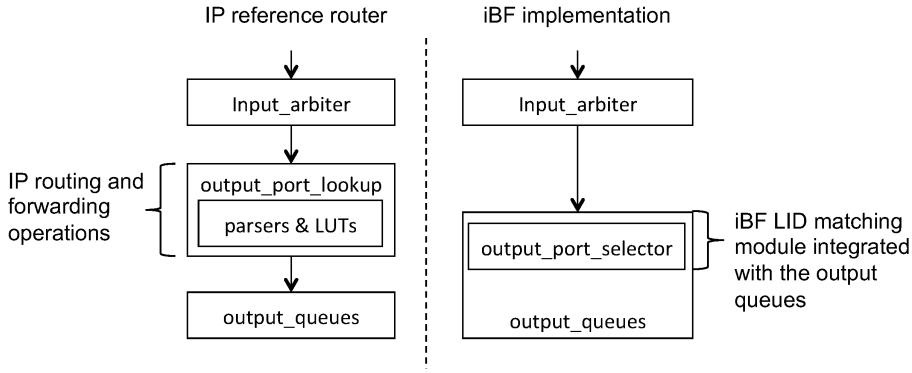



**Figure 4.12.** iBF implementation on a NetFPGA: changes to the IP reference implementation

The *output_port_selector* makes some sanity checks for the packet. Each packet contains a TTL field that is decreased by one when the packet is forwarded out from the router providing a simple loop prevention mechanism. The *bit_counter* calculates the number of bits set to 1 in the packet to avoid attacks that utilize the property of filling the iBF with 1s. This attack is described later in Section 4.4.1. The third verification for the packet is the Ethertype field determining if the packet is redirected to the LIT verification. This check is not necessary in environments where iBF is the sole packet forwarding method.

The packet forwarding array is initialized to contain all 1s. In *do iBF* the main LIT matching is done in 64-bit chunks. If there is a non matching block, i.e. there is 0 in some location where the interface has 1, the corresponding forwarding array bit for that interface is changed to 0. Once all the LITs are verified, the forwarding array has 1s on each of the interface where the packet should be sent out.

Each *id_store* maintains one link identifier and they are configured via the register interface. The implementation collects also some statistics for the packet forwarding.

The implementation efficiency was tested with a ping application. A total of 100,000 packets were sent through the implementation and the test run was repeated with various setups. First, we ran the system as a plain wire without any forwarding operations. Second, the Stanford IP reference implementation for NetFPGA was used to get the time required for processing IP forwarding. And finally, we tested the same forwarding with iBFs. The results shown in Table 4.1 indicate that the iBF forwarding is faster than IP forwarding, pretty close to plain wire packet transit. This shows the efficiency of the implementation in hardware.

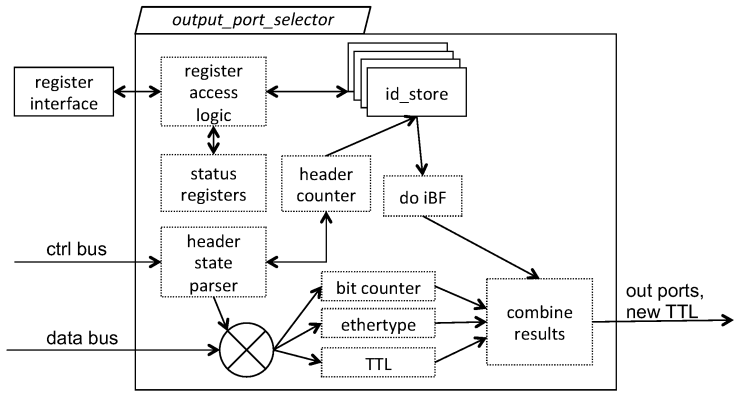



**Figure 4.13.** NetFPGA implementation, output_port_selector module

| Path | Avg. latency | Std. Dev. |
|---|---|---|
| Plain wire | 94$\mu$s | 28$\mu$s |
| IP router | 102$\mu$s | 44$\mu$s |
| iBF | 96$\mu$s | 28$\mu$s |

**Table 4.1.** Ping times for different implementations

### 4.3.2 Multiprotocol Stateless Switching

MPLS was described in Section 2.3.4. MPLS is based on installing connection specific state as labels in network nodes and switching the packet label when the packet is delivered through the network. Especially multicast support in MPLS suffers from complexity ([102], Publication IV). As we have discussed earlier in this chapter, these issues can be solved using iBF forwarding.

In Publication IV, we show the basic idea of replacing the MPLS label switching with iBF, getting rid of the state in the forwarding nodes and simplifying the multicast transmission. We call the solution Multiprotocol Stateless Switching (MPSS).

MPLS uses a centralized path creation mechanism, such as PCE. An enhanced PCE can also be used for creating the iBFs needed for the MPSS. Instead of setting the labels to the network nodes, it is enough to give the iBF to the Provider Edge (PE) node at the source edge. The PE delivers the packet to the network with the corresponding iBF in the header.

As a use case, the MPSS can be applied to L3VPN. In Figure 4.14, the basic operation of the MPSS is shown. Compared to the MPLS L3VPN solution presented in Section 2.3.4, we replace the outer label with an iBF for PE to PE forwarding and leave the inner MPLS label, i.e. the

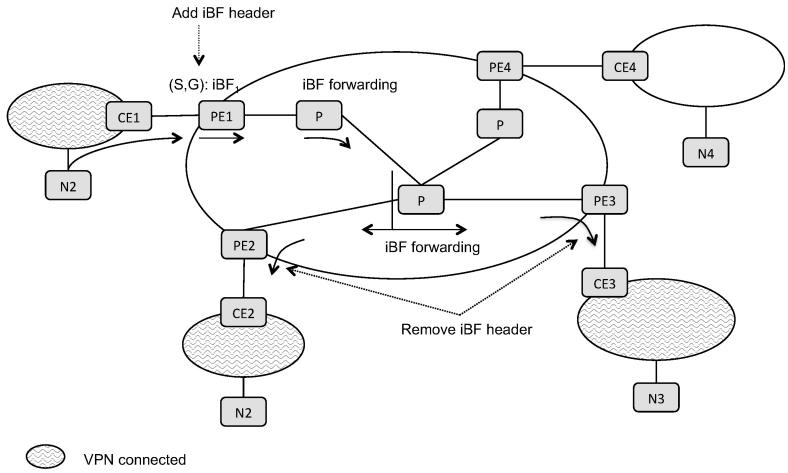



**Figure 4.14.** MPSS architecture

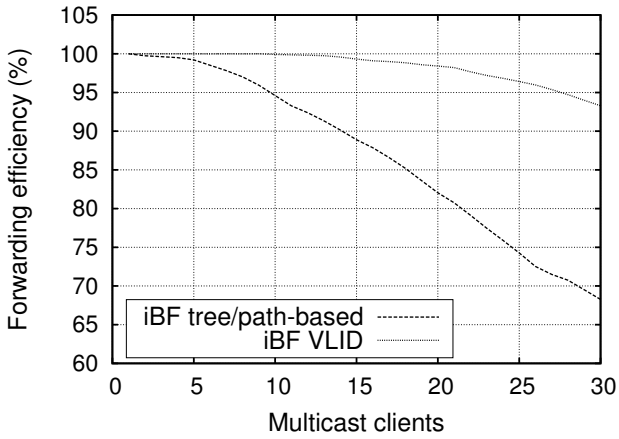



**Figure 4.15.** Forwarding efficiency with plain iBFs and using Virtual Trees (AS6461 topology)

service label, intact. The service label is used outside the PE nodes for mapping traffic to the correct customer networks.

When a data packet sent from a CE node arrives at the PE node, the PE uses the Virtual Routing and Forwarding (VRF) table to retrieve the MPLS label for the inner label and the corresponding iBF to deliver the packet to the destination PE nodes. It attaches the label and the iBF in the packet and sends it to the network.

MPLS multicast solutions have to make a trade-off between the state and bandwidth usage. Enabling a VPN-aware PE node allows the trees to be optimized. However, reducing the state in the network causes additional bandwidth usage because some data is delivered to PE nodes that do not actually need the data. These PE nodes can drop falsely routed

packet using the VPN information in the packet.

The main advantage of an MPSS-based L3VPN is the nearly stateless solution with simple multicast support. The routers in the network are quasi-stateless and the amount of state in the routers depends neither on the number of PE nodes nor on the number of (multicast) VPNs. As we show in our simulation work with the AS6461 topology, the forwarding efficiency with roughly fifteen PEs is around 90% without optimizations, 95% with fpr-optimization (Publication I and Figure 4.6), and roughly 98% when using simple virtual trees (Publication VI and Figure 4.15).

### 4.3.3 Migrating from IP Multicast to iBF

Over the years, different multicast technologies have been proposed to support data delivery from one source node to multiple destinations. The proposed solutions maintain multicast-related state in various locations in the network. For our evaluation, we selected IP multicast as the multicast solution for IP networks. IP multicast creates state on each of the routers through which the multicast channel is delivered. This means that the amount of state in the routers is directly dependent on the number of channels and the number of clients subscribing to the channels.

In Publication VI, we present a deployment scenario, where an operator can replace IP multicast with iBF in its own network. The replacement is invisible to the neighboring IP multicast networks. The solution maps traffic between IP multicast and iBF seamlessly. For evaluation purposes, we implemented the solution in Network Simulator 3 (ns-3) [8]. We evaluated the feasibility of replacing IP multicast with iBF by comparing the amount of state needed by them in the network.

This work further develops and evaluates the initial idea presented by the authors of “BloomCasting: Security in Bloom Filter Based Multicast” [110].

*Network Setup*

Figure 4.16 shows the basic scenario, where the iBF island connects to the neighboring IP networks with Destination Edge (DE) and Source Edge (SE) routers. DE and SE are actually roles in the routers and all edge routers implement both roles. SE routers handle the incoming IP multicast traffic from the multicast source and DE routers convert the traffic back to IP at the other edge of the iBF network. We assumed that Source Specific Multicast [52] with one source and a number of receivers is used

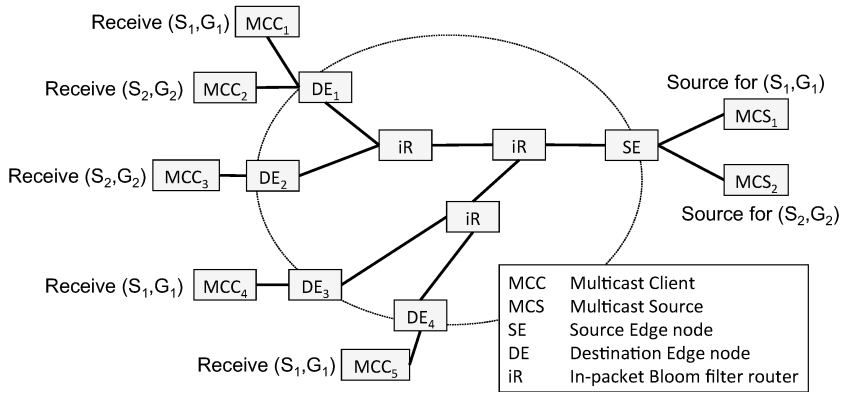



**Figure 4.16.** IP multicast with iBF domain

in the IP networks.

In the iBF network, each router has configured LIds on their interfaces as described in Section 4.1. In addition, IP forwarding is enabled within the iBF network in the simulator. As described in Section 4.1, the iBF for a connection has to be calculated before we can deliver packets through the network. In this work, we had two alternatives to implement the iBF creation. The first alternative was to implement an OSPF-based topology management system (see Section 4.2.1), where all the edge routers would know the whole topology of the network, including the LIds of the interfaces. The second alternative was to use IP-based routing for IP multicast messages through the iBF network and collect the iBF en-route for the data traffic as described in Section 4.1.1.

The purpose of this work was to get information about the required amount of state in the routers for different multicast solutions. The selected signaling message delivery solution does not affect the amount of state required for data delivery, thus we selected the IP-based mechanism because of the ease of implementation in the simulator.

*The Basic Concept*

When an IP multicast client subscribes to a multicast channel, it sends a join message towards the multicast source. In the IP network, the routers on the path establish multicast state for each of the subscribed channel. When the join message arrives at the DE node in the iBF network (see Figure 4.16), the DE router sets an iBF collecting field in the packet and sends it further using IP forwarding inside the simulated iBF network.

Each router on the path in the iBF network adds its incoming interface's LId in the collecting field using the bitwise logical OR operation. Once

the join message arrives at the SE node, the iBF collecting field contains an iBF that can be used to deliver packets from the SE to the DE. The SE router removes the collected iBF and sends the subscription further towards the multicast source. It verifies if there is already a mapping from the channel identifier $(S, G)$ to an iBF. If a mapping exists, a subscription for the same channel has arrived earlier from some other DE node. Using the existing and newly collected iBF, a full iBF can be created by merging them using a logical OR operation. If a mapping is not found, a new mapping is created from the values received in the join message.

When the channel source sends data, the traffic will follow the path established in the routers through the IP network. When the data packet arrives at the SE node, the node will verify its mapping tables. From the tables, the router retrieves the iBF that is mapped to the multicast channel identifier. The IP multicast packet is encapsulated with the iBF and sent to the iBF network which delivers the packet to all the DEs from where the channel has been subscribed.

IP multicast sends periodic messages to verify the status of subscriptions in the routers [29]. These messages do not affect the necessary state, but they add some IP multicast traffic. However, these messages can be delivered between DE and SE nodes in a similar way as other IP multicast messages.

*Storing iBFs at the SE Router*

In our work, we defined three different variants to create and maintain the $(S, G)$ to iBF mapping at the SE router:

- Tree-based iBF

  Only one iBF for each of the multicast channels is maintained at the SE. It does not keep any individual iBFs towards different DEs.

- Tree-based iBF with Counting Bloom filters

  This is similar to the Tree-based iBF, but the Bloom filter is a Counting Bloom filter, enabling removal of individual iBFs towards DEs from the iBF used for a multicast channel.

- Path-based iBF

  The SE maintains an iBF from it to each DE nodes separately and uses a channel to DE nodes mapping to create the full iBF for a certain chan-

nel.

The *tree-based iBF* solution maintains a single iBF entry at the SE router for each of the multicast channels passing through the iBF domain. When a join message arrives at the SE node, the node creates a new mapping for that $(S, G)$ multicast tree if it does not yet exist. The join message has also the iBF collecting field which is filled when the message comes from the DE router to the SE router. Using a logical OR operation, the SE router adds the collected iBF into the iBF field mapped to that specific channel $(S, G)$. It does not store the collected iBF as a separate entry, but it discards it. Thus, each IP multicast channel has a single iBF entry that contains the full iBF tree at the SE. When multicast data packets arrive at the SE node, this iBF is used to deliver the data to the correct egress DE routers. In Figure 4.17 a), we can see the required mappings for the multicast channel at the SE node.

Bloom filters do not allow removal of items. Thus, when a DE node sends a prune message to the SE node indicating unsubscribing from a multicast channel, the iBF for the channel must be recalculated using the individual iBFs towards the active DE nodes. We have two alternatives with the tree-based iBF solution: the SE node can explicitly request a new join message from the DE nodes that are active or the SE can wait for the DE nodes to respond to the periodical IP multicast queries sent from the SE to the DE nodes. In both cases, the iBF for the delivery tree is recalculated with the received individual iBFs from the active DE nodes.

The *tree-based iBF with Counting Bloom filters* assigns a counter for each Bloom filter bit at the $SE$ node. This solution requires more memory than the pure tree-based iBF solution but reduces network traffic. When an SE router receives a request to unsubscribe from a channel, it retrieves the collected iBF in the prune message. The DE can now be removed from the multicast channel's iBF by subtracting the corresponding counters in the iBF that were set in the collected iBF field.

The *path-based iBF* maintains iBF information for each separate point-to-point path from the SE to all the DE routers that have sent a join to this SE. Thus, the amount of iBFs stored at a SE is at most the number of DE nodes in the network. In addition, each multicast channel at the SE is mapped to a set of DE routers defining the egress nodes for that particular multicast channel. These mappings are presented in Figure 4.17 b).

These solutions do not maintain any multicast-related state in the core

a)

| IP Multicast group | iBF of the tree |
|---|---|
| $(S_1, G_1)$ | $iBF_{S1,G1} = iBF_1 \mid iBF_3 \mid iBF_4$ |
| $(S_2, G_2)$ | $iBF_{S2,G2} = iBF_1 \mid iBF_2$ |

b)

| IP Multicast group | Destination Edges | iBF of the tree |
|---|---|---|
| $(S_1, G_1)$ | $DE_1, DE_3, DE_4$ | $iBF_1 \mid iBF_3 \mid iBF_4$ |
| $(S_2, G_2)$ | $DE_1, DE_2$ | |

| Destination Edge | iBF to the DE |
|---|---|
| $DE_1$ | $iBF_1$ |
| $DE_2$ | $iBF_2$ |
| $DE_3$ | $iBF_3$ |
| $DE_4$ | $iBF_4$ |

**Figure 4.17.** Mapping IP Multicast channel $(S, G)$ to an iBF: a) tree-based iBFs, b) path-based iBFs

routers and all the required state is in the ingress SE router. To improve the performance, we used virtual trees that were presented in Section 4.1.4. This solution reduces the number of set bits in the iBFs with the cost of added state in the forwarding nodes.

In Publication VI, virtual trees are used in a simplified way. The trees are only paths between two nodes and we call them as *virtual paths*. In the presented solution the virtual path can be set up automatically without the need of a separate topology entity to create the paths. The simulated solution is also simple where virtual paths are established between all the edge-to-edge connections. The performance results are compared with the corresponding results from IP multicast and pure iBF based forwarding.

*Results*

The purpose of the simulation work was to compare the required amount of state in the network nodes. In the Figure 4.17 we can see the theoretical requirements for state with tree-based and path-based solutions at the SE node. For counting Bloom filters, the amount of state is the same as for tree-based solution, but the required amount of memory is larger due to the counters in each bit position. These solutions do not establish state in the routers between SEs and DEs. For virtual paths, we have to calculate

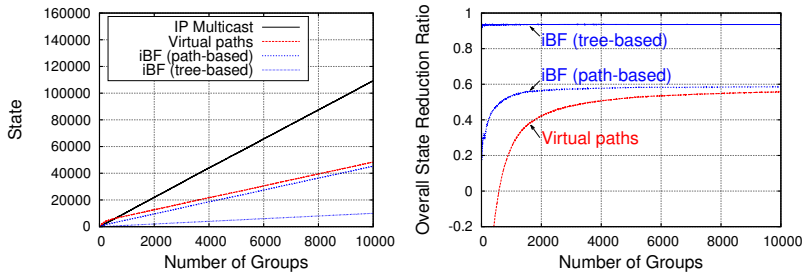
also the amount of state that has to be established in those routers.

In Figures 4.18 and 4.19, the amount of state required by different solutions is presented. Two different network setups were used in the simulations. In the first setup, only 26 of the AS6461 topology routers were selected as potential edge routers. The rest of the nodes were acting as core routers and were only forwarding traffic. In the second setup, all 138 nodes were potential edge routers. In the latter alternative, there are more potential paths through the network and with virtual paths this means that there will be more VLIds installed on the network nodes. This affects also the forwarding efficiency, while the number of VLIds on the interfaces increase.

Figures 4.18 (a) and 4.19 (a) show the amount of state that the network has to maintain. In both scenarios, the iBF tree-based solution has the lowest amount of state; there is only one mapping at the SE router between the incoming IP multicast channel and the iBF to the DE routers. The path-based solution uses more state because each of the SE routers must maintain one mapping for every SE-DE connection. Virtual paths install the VLIds in the network nodes as long as there are new connections established between SE and DE nodes. Once all the SE-DE pairs have their own virtual paths, the amount of state does not grow any more in the network nodes. This can be seen clearly in Figure 4.18 where the number of potential connections between edges is much lower than is the case shown in Figure 4.19. In the former case, we reach quickly the situation where all the potential virtual paths have been established. In the latter case, this has not happened yet with 10.000 multicast channels.

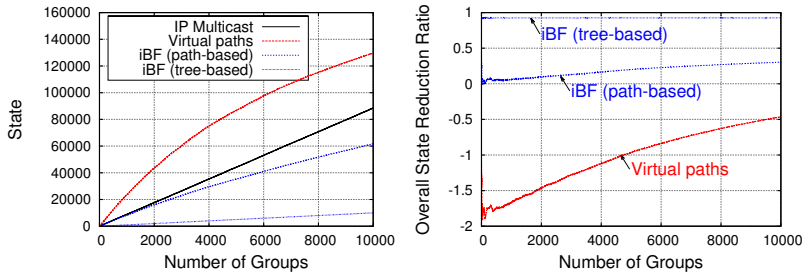
Figures 4.18 (b) and 4.19 (b) show the overall state reduction when compared to IP multicast. With virtual paths, the amount of state with a low number of channels is greater than with IP multicast (negative value in Overall state reduction). This happens because we set up virtual paths immediately in the beginning, creating state both in the network and in the SE nodes. When there are multiple channels using the same connections (and the same virtual paths) the situation changes. We do not need to create more state in the network routers and the amount of state grows with the same speed as with the path-based solution.

With IP multicast the forwarding efficiency is obviously 100% because there are no false routing decisions in the network if everything is configured correctly. With the presented tree-based, path-based, and virtual path iBFs, we get different results for the forwarding efficiency. Tree-




(a) State Count (b) Overall state reduction

**Figure 4.18.** State comparison with 26 edge routers.

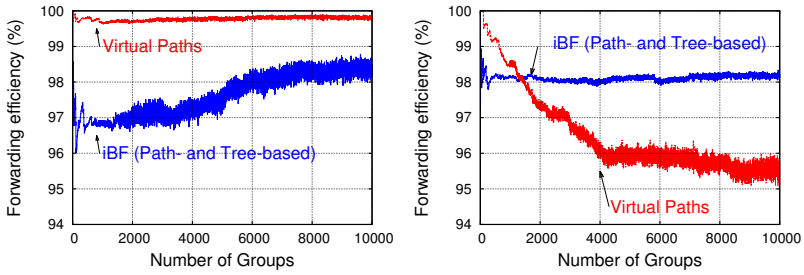



(a) State Count (b) Overall state reduction

**Figure 4.19.** State comparison with 138 edge routers.

based and path-based solutions have the same forwarding efficiency because the difference of the solutions is only in storing the iBFs at the SE. iBFs in both solutions are the same for data traffic. In Figure 4.20, we see the forwarding efficiency for the two different network setups.

When the number of edge nodes is low (see Figure 4.20(a)), the virtual path solution provides nearly 100% forwarding efficiency. The number of VLIds in the network nodes is relatively low and the forwarding clearly gains from the fact that the amount of bits set to 1 in the iBF is low. However, in Figure 4.20(b), the number of VLIds in the network increases when more multicast channels are added to the network. Once there are more connections the number of VLIds is not increasing so rapidly because most of the potential SE - DE connections have already been established and the corresponding VLIds inserted in the forwarding nodes. As a result, the forwarding efficiency stabilizes to slightly above 95%.




(a) 26 edge nodes

(b) 138 edge nodes

**Figure 4.20.** Forwarding efficiency comparison.

## 4.4 Security

The probabilistic nature of iBFs provide some security features by default. The encoded path is specific for a certain source location and the same iBF cannot be used from any other location. It is also hard to extract any route or end-host information from the iBF because there is no information in the bits that would specify to which LId they would belong. If an attacker decides to generate a random iBF, it should match all the LIds on the path from the source to any destination. When we limit the number of inserted 1s in the iBF to a certain percentage (e.g. 50%), it is nearly impossible to send traffic to some specific destination node multiple hops away.

### 4.4.1 Threats

It is clear that certain attacks are possible against a simple source routing system. In Publication I and Publication III a set of potential attacks have been identified. The presented list of attacks is not necessarily a complete list and new attacks can be identified in the future. At the time of writing the following are the obvious attacks that need attention.

1. iBF contamination attack

   The attacker fills the iBF with 1s resulting in data broadcasting. A simple solution is to limit the number of 1s in a iBF. Typically this fill factor is 50-70% and the verification of the number of bits can be implemented in hardware without any additional delay [Publication II].

2. iBF learning and re-use attack

The attacker tries to figure out the link identifiers close to it by luring nodes into subscribing traffic from it, collecting the iBFs, and using AND operations between the received iBFs to guess some LIds close to the attacker. This attack consumes a lot of resources from the attacker.

3. iBF computational attack

The attacker collects valid iBFs. Knowing the source, sink(s) and iBF triplets, it can try to analyze and extract potentially valid iBFs for certain parts of the paths. If there are similar patterns, the attacker may be able to find out some, at least partial, delivery trees that it can use for sending data.

4. iBF replay attack

When a node receives an iBF to send requested data to the destination nodes, it can try to reuse the same iBF also for other data and deliver unrequested information to the same destinations. This is possible with the basic iBF because the iBF is not tied to the stream or content (see Section 4.1).

5. iBF injection attack

If an attacker can find out an iBF that goes to a potential victim node and it can get to the path from the middle, the attacker can inject traffic to the victim from the middle of the path.

In general, changing the LIds periodically in the network leads to better security because the learning of existing LIds gets more difficult. The change period could be for example ten minutes. This limits the effects of some of the attacks, but requires refreshing the generated iBFs when one connection lasts over the LId change time.

The described attacks are possible because the LIds and the iBF are not tied to any information related to the connection. The identifiers are basically fixed and the same iBF can be used for any data delivered via the path described in the iBF. In the next section, we describe a forwarding mechanism that addresses this problem.

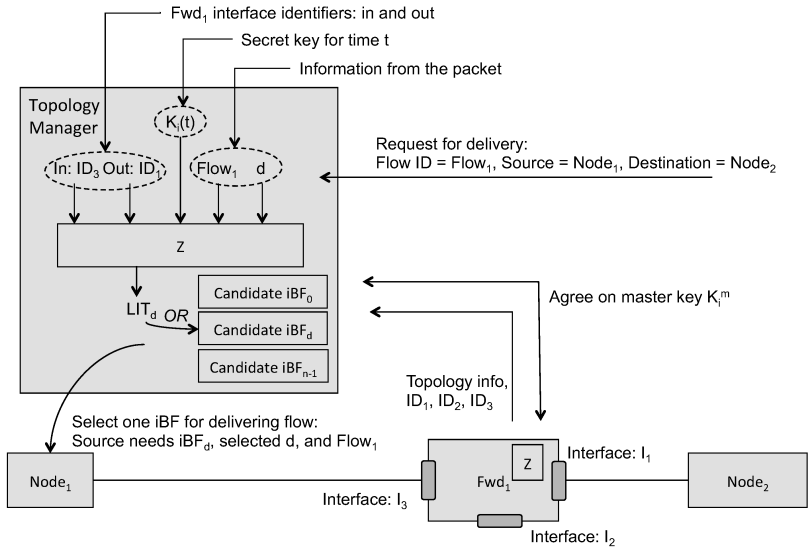



**Figure 4.21.** Creating an iBF with z-Formation

### 4.4.2 z-Formation

So far, we have described only fixed or periodically changing LIds and LITs on interfaces. They are the simplest way of deploying iBF forwarding in cases where there are no threats in the network. However, this is not a typical situation in the Internet so it is necessary to be prepared also for different kinds of attacks that utilize the weak points in the forwarding system.

To prevent attacks, forwarding nodes should be able to distinguish between authorized and unauthorized traffic. It is not possible to store all flow-based information in the forwarding nodes to determine if a packet belongs to a valid flow or not because the number of flows passing a single router may be huge. In Publication VI, we created a mechanism that ties the LIT to a flow identifier, time, and path. We call this mechanism *z-Formation*.

z-Formation does not use fixed LITs for interfaces at the forwarding nodes. Instead, LITs are dynamically calculated using flow identifier, time-dependent information, and path information. A TM and a forwarding node share the same information and a TM can calculate the correct LITs for each of the forwarding nodes on the path during the iBF creation phase. A forwarding node can repeat the same calculation for every incoming packet and compare the result with the iBF in the packet header to make a forwarding decision.

Figure 4.21 shows the z-Formation mechanism. The TM collects the topology and interface information from the forwarding nodes. It also negotiates a shared master key $K^m_i$ with each of the forwarding nodes $i$. With the master key, the topology manager and a forwarding node can calculate a time-dependent shared secret $K_i(t) = F(K^m, t)$, where $F$ is a cryptographically secure pseudo-random function.

In the example case in Figure 4.21, the TM receives a request to create an iBF for a flow identified with ID $Flow_1$ from the source, $Node_1$ to the destination, $Node_2$. The TM creates a delivery tree for the flow which goes via interfaces $I_3$ and $I_1$ of the forwarding node $Fwd_1$. The TM calculates an outgoing LIT using the Z-function. For each LIT that the Z-function calculates, it requires both the incoming and outgoing interfaces of the forwarding node binding the LIT to the path, the shared key binding the LIT to time, and the flow identifier binding the LIT to this flow. Using the logical OR operation, the TM adds the calculated LITs to the corresponding candidate iBF. The $n$ candidate iBFs are calculated with $d$ values ranging from zero to n-1.

Once all the candidate iBFs are ready, the TM selects one using the same mechanisms presented in Section 4.1.4. The TM sends this iBF, together with the selected d value and the flow identifier, to the source node that can start sending data using this information.

When a packet arrives at the forwarding node (see Figure 4.22), the forwarding node reads the flow identifier and d from the packet. It uses the Z-function to calculate LITs for all the potential outgoing interfaces. It inserts the flow ID and d value, incoming interface's ID, K(t), and the outgoing interface's ID as parameters to the Z-function. The forwarding node compares the calculated LIT values with the iBF in the packet (see Section 4.1.2) and if there is a match, forwards the packet out on that link.

The Z-function is the core of the z-Formation mechanism. In Publication VI, we describe that it can be implemented in a stream-cipher-like construction. The presented mechanism makes it harder for an attacker to guess any valid iBFs, even partial ones. If some path information is compromised, the iBF can be used for attacks only during the time period t. After that, new keys are generated and the old iBF is no longer valid.

Ghani and Nikander [47] implemented z-Formation on a NetFPGA router based on our original implementation presented in Publication II. Using this implementation, they estimated the delay caused by the z-Formation calculations. The results show that the z-Formation processing adds roughly

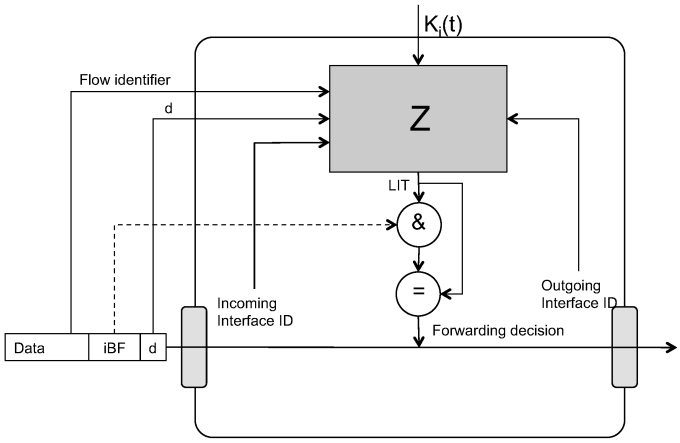



**Figure 4.22.** Z-Formation

half a microsecond (approximately 3%) to the processing latency of a packet when compared to the plain LId based iBF forwarding in the same NetFPGA router. This can be considered to be negligible.

## 4.5 Future Directions and Related Work

The iBF forwarding has inspired further research around the topic. There are proposals e.g. to improve scalability or to support mobility using iBFs.

### 4.5.1 Scalability Improvements

In addition to the BRA architecture presented in Section 4.2.2, some alternative solutions have also been developed for better scalability.

The performance of iBF forwarding can be enhanced by splitting the delivery tree at the source node [101]. When the iBF is created, the number of 1s in the iBF is calculated and if a defined threshold value is exceeded, e.g. 50%, the tree is split. This means that the source node has more than one iBFs for the same delivered data. Each of the iBFs serve a different set of subscribers. The splitting procedure continues every time the threshold value is exceeded for a single iBF. This solution increases the traffic in the network and causes some duplicates of the same data to be delivered over the same links close to the source node. E.g. if the tree is split into two, but both trees leaving the source node using the same link, the packet is delivered twice over the first link. However, at the same time the solution reduces the number of false positives in the network and allows the usage of iBFs also for a larger number of multicast

receivers.

The Filter switching proposal [123] takes a different approach for dividing the multicast tree. The solution uses mapping points inside the network, where the iBF header is changed to another one. This approach has some similarities to the BRA system described in 4.2.2. The solution calculates the subtrees in a slightly different way, but the result is approximately the same. The iBF in the packet header is changed to a new one at certain branching points. This adds some state in the network router.

### 4.5.2 Using iBFs in Different Types of Forwarding Nodes

In optical routers, the problem is to make the forwarding decision purely on the optical level and to avoid the O-E-O conversion that is required when packets are forwarded using IP. Theoretical work has been done to verify the suitability of iBFs for enabling all-optical packet forwarding [11]. The authors propose a Serial Processing Unit (SPU), that is capable of making packet forwarding decision at the optical level using a small number of photonic logic gates. The all-optical processing makes the operation of the router faster than converting the incoming packet header first to electronic format for making the forwarding decision. The proposed solution does not need to use any memory units. The SPU supports also wavelength multicasting providing the possibility to forward a single incoming packet out from multiple outgoing interfaces.

### 4.5.3 Mobility Management with iBFs

Not all the network nodes are fixed elements and when the nodes move, the forwarding mechanism must support mobility for uninterrupted data transfer. In IPv4 networks, this can be handled using e.g. Mobile IPv4 [84].

The $\Psi$ architecture proposes to use iBFs for mobility management [45]. $\Psi$ is designed for Information Centric Networks, where the communication is utilizing the publish/subscribe paradigm. However, the mobility management scheme that the authors present for iBF, is not tied to the ICN principles and it was originally introduced in “Fast Inter-domain Mobility with In-packet Bloom filters” [108]. When the Mobile Node (MN) initiates a connection to the Correspondent Node (CN), it receives the initial iBF from the topology function. The reverse path iBF is collected enroute. The initial packet contains also authentication information based on hash chains to support future location update messages when the MN

has moved. After initial message exchange, both nodes have iBFs to send data to the peer node.

After the MN moves, it requests a new iBF towards the CN from the topology function. The MN sends a location update packet using this new iBF towards the CN, including the required authentication information. The location update packet collects the reverse path iBF and the CN can start using the new iBF once it has authenticated the MN.

As always in mobility management, the problem is making a seamless handover. If a mobile device disconnects from its current network before it has attached to a new network, seamless handover is not possible. In the presented solution, bicast can be supported and the MN can send the information about its new location before the old connection is broken. The CN creates a temporary iBF by merging the old and new iBFs together using a logical OR operation. Using this temporary iBF, the CN can send data to both locations to allow the MN to accomplish its move to the new location.

### 4.5.4 Energy Efficiency

Zheng and Wang showed in their work [129] that iBF forwarding provides 3.5% - 41.6% saving in energy consumption when compared to IP forwarding. The authors used a NetFPGA router for calculating the required energy in various conditions. These figures show that alternatives for the IP forwarding can be considered to reduce some of the network-wide energy consumption. iBFs provide one potential solution for this problem.

## 4.6 Summary

In this chapter, we presented a new kind of packet forwarding mechanism based on Bloom filters. In our work we showed that an in-packet Bloom filter can be used to store source route information in a small and fixed size packet header. With our prototype on a NetFPGA router, we showed that packet forwarding with iBFs is faster than using traditional IP-based forwarding. We proved that iBF is deployable gradually and it provides savings in network resources. iBFs have also other advantages, such as easier multicast management and easier setup of MPLS-style networks.

The limitations of iBF forwarding are related to the false routing decisions due to false positives during the Bloom filter verification. This may

cause problems when the amount of added links in the filter increases. Especially with inter-domain routing, enhancements are needed to reduce the amount of set bits in a single iBF. To mitigate these problems, we have proposed various enhancements for iBF, e.g. z-Formation to enhance security and virtual trees to achieve better performance in terms of false positives.

Other research has shown potential advantages using iBF forwarding. For example, energy efficiency has been investigated [129] and the research has shown the potential of iBFs in saving energy when compared to IP-based forwarding. They can probably also be used to simplify forwarding implementations e.g. in all-optical routers [11], providing potentially savings in the implementation technology.

With these advantages in mind, it may be worthwhile for an operator to consider switching to a new forwarding mechanism in some certain network environments. There is always an investment when a new technology is deployed, but the potential savings described can provide profit for the operator on the long run.

# 5. Conclusions

IP-based routing and forwarding has been used for decades in the Internet. The growth of the Internet during these years has not come without a cost. IPv4 address exhaustion, efficiency problems due to changing user behavior, and attacks against hosts and servers have initiated research on different areas to propose solutions for the problems. Our work started in a project designing a completely new architecture for future Internet, based on information centric approach. To better support the transport in the Information Centric Networks and to fix some of the shortcomings of IP-based transmission, we introduced a new forwarding mechanism, in-packet Bloom filters (iBF). During the work, it became obvious that this forwarding solution can also be used in the current Internet to support or replace IP in some areas.

The solution that we created, provides a so-called quasi-stateless forwarding that is based on strict source routing and compresses the forwarding information in a compact, fixed length Bloom filter. Forwarding nodes in the network do not need to maintain any routing table, but forwarding is based on verifying if any of the link identifiers of the router have been inserted in the filter in the packet header. In addition, we created an enhanced virtual tree solution that provides better performance than the basic iBF in some circumstances and incurs only a cost of a small amount of additional state in the forwarding nodes.

As we showed, iBF forwarding provides faster forwarding than IP. It also supports multicast with considerably less state in network routers compared to IP multicast. These advantages suggest that the solution can, under suitable circumstances, provide more effective and scalable forwarding than IP.

To avoid potential attacks against the new forwarding system, we introduced a z-Formation mechanism that generates the required link identi-

fiers on the fly when a packet arrives at the forwarding node. The generated identifier can be bound to a flow identifier in the packet, making it very hard to try for an attacker to guess arbitrary iBFs to forward packets through the network. This solution was proven to be fast and simple to implement. The solution also provides additional advantages because the forwarding can be controlled in multiple ways.

The deployment scenarios that we presented for replacing MPLS label switching and IP multicast show that we can benefit from deploying iBF in small, internal networks while maintaining compatibility with external networks. The potential gains in these areas are significant and can provide savings for the deploying operator in terms of simpler hardware and more efficient throughput.

As we have shown, iBFs can provide many advantages over IP. However, some trade-offs have to be taken into account. The main drawback of iBF is the amount of false positive forwarding decisions during packet forwarding because they create unnecessary traffic in the network when a packet is sent over an incorrect link. Hence, false positives become more probable the longer forwarding paths are included in the filter. The performance can be improved with careful network design or by using e.g. virtual trees as we have suggested.

False forwarding decisions can also lead to loops in the network as we showed in our simulations. We provided some solutions for preventing loops, but these solutions require some state either in the network or in the packet. Still, the amount of state is relatively low.

False positives can be problematic in large networks. Thus, at this phase iBF provides best results when adopted for specific use scenarios, such as replacing IP multicast inside an operator network or replacing MPLS label switching with iBFs. Other researchers have shown that data centers can benefit from iBF. However, the deployment decision must be based on careful evaluation of the use scenario.

Although proposals to enhance scalability exists, such solutions require more state information either in some of the routers or in the packet itself. If the iBF in the packet header is switched to a new one in a designated router, the router must maintain flow-based information about the next iBF to be used when the packet leaves the router. This affects the robustness of the forwarding system, e.g. when a router fails and routing information must be updated to reroute traffic. It is also possible to stack the iBFs corresponding to different parts of the delivery tree in the packet

header and certain branching nodes can remove the outermost iBF from the header. This increases the initial size of the packet, by adding e.g. multiple 512-bit iBFs in the header.

An insight from our research is that independent of the purity of the initial idea, additional requirements in a complex environment can create requirements that disrupt the original simplicity of the design. Including security to our solution incurs complexity when we introduced the z-Formation. However, the z-Formation was still a useful addition to the original design because it helped us to solve some of the identified problems with abusing the weaknesses of the basic iBF forwarding. The increasing amount of false positives during the forwarding process concerned us. From the purity point of view, this concern was justified: the Bloom filter-based Relay Architecture (BRA) solution that we presented make the iBF forwarding scalable, but the beauty and simplicity of the solution was, to some extent, lost.

The work we have done, as well as the related work by others, shows that iBF-based forwarding can be deployed even in larger networks. We see that this area is a source for more ideas that can in long run make packet forwarding in the network more efficient. It requires deeper analysis of topology management, nature of iBFs, and IBF management in different kinds of network setups. As an example, one potential solution for scalability - and an area that has not been deeply studied - is the usage of virtual trees. Our work only proved that they can provide better performance under certain conditions. However, the partial lack of solutions in some areas does not impede deploying iBF forwarding in the areas discussed in the author's publications.

In the thesis, we shortly discussed using iBFs in optical routers to enable efficient optical domain forwarding. Other researchers have shown that such deployment may be possible and the author sees this as one important area for further studies to design faster and simpler optical routers.

Current and future needs for energy efficiency together with the increasing amount of traffic in the network are changing how networks are designed. Efficient packet forwarding is one of the key solutions to reduce energy consumption in the routers. The possibility to deploy gradually new, simpler and potentially more energy efficient mechanisms without affecting negatively to the quality of experience of the end-user, can pave the way for iBF usage in the Internet.

# References


[1] CIDR Report. http://www.cidr-report.org/as2.0 (accessed: August 27, 2014).

[2] Content Centric Networking project CCNx. http://www.ccnx.org (accessed: August 27, 2014).

[3] EU FP7 project PSIRP (Publish-Subscribe Internet Routing Paradigm). http://www.psirp.org (accessed: August 27, 2014).

[4] EU FP7 project PURSUIT (Publish Subscribe Internet Technology). http://www.fp7-pursuit.eu (accessed: August 27, 2014).

[5] EU FP7 project SAIL (Scalable and Adaptive Internet Solutions (SAIL)). http://www.sail-project.eu (accessed: August 27, 2014).

[6] IEEE 802.3 Ethernet Working Group. www.ieee802.org/3/ (accessed: August 27, 2014).

[7] NetFPGA Programmable Router. http://www.netfpga.org (accessed: August 27, 2014).

[8] Network simulator 3. http://www.nsnam.org (accessed: September 2, 2014).

[9] Rocketfuel ISP topology data. http://www.cs.washington.edu/research/networking/rocketfuel/maps/weights-dist.tar.gz (accessed: August 27, 2014).

[10] Allot MobileTrends, Global Mobile Broadband Traffic Report. http://www.allot.com, February 2011.

[11] M. F. Al-Naday, R. C. Almeida Jr., K. M. Guild, and M. J. Reed. Design Proposal of a Photonic Multicast Bloom Filter Node. *Photonic Network Communications, Vol 24, Issue 2, pp. 132-137*, October 2012.

[12] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. Scalable Bloom Filters. *Information Processing Letters, Vol 101, Issue 6, pp. 255-261*, March 2007.

[13] David G Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. Accountable internet protocol (AIP). *ACM SIGCOMM Computer Communication Review, Vol 38, Issue 4, pp. 339-350*, August 2008.

[14] L. Andersson, I. Minei, and B. Thomas. LDP Specification. RFC 5036 (Draft Standard), October 2007. Updated by RFC 6720.

[15] L. Andersson and E. Rosen. Framework for Layer 2 Virtual Private Networks (L2VPNs). RFC 4664 (Informational), September 2006.

[16] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks, Vol 54, Issue 15, pp. 2787-2805*, October 2010.

[17] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209 (Proposed Standard), December 2001. Updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711, 6780.

[18] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702 (Informational), September 1999.

[19] Louise Barkhuus. Television on the Internet: New Practices, New Viewers. *CHI '09 Extended Abstracts on Human Factors in Computing Systems, ACM, pp. 2479-2488*, April 2009.

[20] T. Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456 (Draft Standard), April 2006.

[21] Alex Bikfalvi, Jaime Garcia-Reinoso, Ivan Vidal, Francisco Valera, and Arturo Azcorra. P2P vs. IP multicast: Comparing approaches to IPTV streaming based on TV channel popularity. *Computer Networks, Vol 55, Issue 6, pp. 1310-1325*, April 2011.

[22] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM, Vol 13, Issue 7, pp. 422-426*, July 1970.

[23] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms. Explicit Multicast (Xcast) Concepts and Options. RFC 5058 (Experimental), November 2007.

[24] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936, 4495, 5946, 6437, 6780.

[25] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics, Vol 1, Issue 4, pp. 485-509*, 2002.

[26] Tian Bu, Lixin Gao, and Donald F. Towsley. On routing table growth. *Computer Communication Review, Vol 32, Issue 1, pp. 77-77*, January 2002.

[27] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439 (Informational), December 2002.

[28] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. ROFL: Routing on Flat Labels. *SIGCOMM Computer Communications Review, Vol 36, Issue 4, pp. 363-374*, August 2006.

[29] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), October 2002. Updated by RFC 4604.

[30] R. Callon and M. Suzuki. A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs). RFC 4110 (Informational), July 2005.

[31] Brian E. Carpenter. Observed Relationships between Size Measures of the Internet. *SIGCOMM Computer Communications Review, Vol 39, Issue 2, pp. 5-12*, March 2009.

[32] D. R. Cheriton and M. Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. *Stanford Computer Science Technical Report*, January 2000.

[33] William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[34] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. OSPF for IPv6. RFC 5340 (Proposed Standard), July 2008.

[35] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 1883 (Proposed Standard), December 1995. Obsoleted by RFC 2460.

[36] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[37] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), August 1989. Updated by RFC 2236.

[38] S.E. Deering and D.R. Cheriton. Host groups: A multicast extension to the Internet Protocol. RFC 966, December 1985. Obsoleted by RFC 988.

[39] Sarang Dharmapurikar, Praveen Krishnamurthy, and David E. Taylor. Longest Prefix Matching using Bloom Filters. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 201-212*, August 2003.

[40] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik, Vol 1, Issue 1, pp. 269-271*, December 1959.

[41] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network, Vol 14, Issue 1, pp. 78-88*, January 2000.

[42] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking, Vol 8, Issue 3, pp. 281-293*, June 2000.

[43] A. Farrel, J.-P. Vasseur, and J. Ash. A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational), August 2006.

[44] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), May 2000. Updated by RFC 3704.

[45] Nikos Fotiou, Konstantinos V. Katsaros, George C. Polyzos, Mikko Särelä, Dirk Trossen, and George Xylomenos. Handling mobility in future publish-subscribe information-centric networks. *Telecommunication Systems, Vol 53, Issue 3, pp. 299-314*, July 2013.

[46] K. Fouli and M. Maier. The road to carrier-grade Ethernet. *IEEE Communications Magazine, Vol 47, Issue 3, pp. 30-38*, March 2009.

[47] Adnan Hassan Ghani and Pekka Nikander. Secure in-packet Bloom Filter forwarding on the NetFPGA. 1st European NetFPGA Developers Workshop, September 2010.

[48] G. Goth. The End of IPv4 is Nearly Here - Really. *IEEE Internet Computing, Vol 16, Issue 2, pp. 7-11*, March - April 2012.

[49] M. Handley, E. Rescorla, and IAB. Internet Denial-of-Service Considerations. RFC 4732 (Informational), December 2006.

[50] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930 (Best Current Practice), March 1996.

[51] C.L. Hedrick. Routing Information Protocol. RFC 1058 (Historic), June 1988. Updated by RFCs 1388, 1723.

[52] H. Holbrook and B. Cain. Source-Specific Multicast for IP. RFC 4607 (Proposed Standard), August 2006.

[53] H. Holbrook, B. Cain, and B. Haberman. Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard), August 2006.

[54] G Huston. Growth of the BGP Table - 1994 to Present. http://bgp.potaroo.net (accessed: August 27, 2014), 2014.

[55] International Telecommunication Union. The International Public Telecommunication Numbering Plan. ITU-T Recommendation E.164, May 1997.

[56] R. Jain. Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation. *IEEE Military Communications Conference, MILCOM 2006, pp. 1-9*, October 2006.

[57] P. Jokela, R. Moskowitz, and J. Melen. Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP). RFC 7402 (Proposed Standard), April 2015.

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

[59] Petri Jokela, Pekka Nikander, Jan Melen, Jukka Ylitalo, and Jorma Wall. Host identity protocol: Achieving ipv4 œ ipv6 handovers without tunneling. 2003.

[60] Petri Jokela, Pekka Nikander, Jan Melen, Jukka Ylitalo, and Jorma Wall. Host identity protocol-extended abstract. *Wireless World Research Forum*, 2004.

[61] Petri Jokela, Teemu Rinta-aho, Tony Jokikyyny, Jorma Wall, Martti Kuparinen, Heikki Mahkonen, Jan Melén, Tero Kauppinen, and Jouni Korhonen. Handover performance with hip and mipv6. In *Wireless Communication Systems, 2004, 1st International Symposium on*, pages 324–328. IEEE, 2004.

[62] D. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (Proposed Standard), September 2003. Updated by RFCs 4203, 5786.

[63] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303 (Proposed Standard), December 2005.

[64] Changhoon Kim, Matthew Caesar, and Jennifer Rexford. SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. *ACM Transactions on Computer Systems, Vol 29, Issue 1, pp. 1-35*, February 2011.

[65] Adam Kirsch and Michael Mitzenmacher. Less Hashing, Same Performance: Building a Better Bloom Filter. *Random Structures and Algorithms, Vol 33, Issue 2, pp. 187-218*, September 2008.

[66] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Network Architecture. *SIGCOMM Computer Communications Review, Vol 37, Issue 4, pp. 181-192*, October 2007.

[67] Franck Le, Geoffrey G. Xie, and Hui Zhang. On route aggregation. *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies, CoNEXT '11, Article 6, pp. 6:1-6:12*, December 2011.

[68] Dan Li, Jiangwei Yu, Junbiao Yu, and Jianping Wu. Exploring efficient and scalable multicast routing in future data center networks. *IEEE INFOCOM, pp. 1368-1376*, April 2011.

[69] T. Li and H. Smit. IS-IS Extensions for Traffic Engineering. RFC 5305 (Proposed Standard), October 2008. Updated by RFC 5307.

[70] G. Malkin and R. Minnear. RIPng for IPv6. RFC 2080 (Proposed Standard), January 1997.

[71] E. Mannie. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. RFC 3945 (Proposed Standard), October 2004. Updated by RFC 6002.

[72] M. Mealling and R. Denenberg. Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations. RFC 3305 (Informational), August 2002.

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

[74] D.L. Mills. Exterior Gateway Protocol formal specification. RFC 904 (Historic), April 1984.

[75] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.

[76] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson. Host Identity Protocol Version 2 (HIPv2). RFC 7401 (Proposed Standard), April 2015.

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

[78] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFCs 5709, 6549.

[79] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. PortLand: a scalable fault-tolerant layer 2 data center network fabric. *SIGCOMM Computer Communications Review, Vol 39, Issue 4, pp. 39-50*, October 2009.

[80] Jukka Nousiainen. Management of Carrier Grade Intra-Domain Ethernet. *Master's Thesis, Aalto University, Department of Communications and Networking*, March 2010.

[81] M. O'Dell. GSE: An Alternate Addressing Architecture for IPv6. Internet-Draft draft-ipng-gseaddr-00.txt, IETF Secretariat, February 1997.

[82] D. Oran. OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (Informational), February 1990.

[83] George Pallis and Athena Vakali. Insight and Perspectives for Content Delivery Networks. *Communications of the ACM, Vol 49, Issue 1, pp. 101-106*, January 2006.

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

[85] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

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

[87] S. Persia and D. Cassioli. IPv4 Wireless multimedia sensor networks. *Third International Workshop on Software Engineering for Sensor Network Applications (SESENA), pp. 58-63*, June 2012.

[88] L.L. Peterson and B.S. Davie. *Computer Networks: A Systems Approach, 3rd Edition*. The Morgan Kaufmann Series in Networking. Elsevier Science, 2003.

[89] Sébastien Pierrel, Petri Jokela, Jan Melén, and Kristian Slavov. A policy system for simultaneous multiaccess with host identity protocol. *Proc. ACNM*, pages 71–77, 2007.

[90] David M. Piscitello and A. Lyman Chapin. *Open Systems Networking: TCP/IP and OSI*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1993.

[91] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[92] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFCs 1349, 2474.

[93] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[94] Jarno Rajahalme. Inter-Domain Incentives and Internet Architecture. *PhD thesis, Aalto University, Department of Computer Science and Engineering*, August 2012.

[95] R. Ramaswami, K. Sivarajan, and G. Sasaki. Optical networks: A practical perspective, 3rd edition. *Morgan Kaufmann Publishers Inc.*, 2009.

[96] Sylvia Ratnasamy, Andrey Ermolinskiy, and Scott Shenker. Revisiting IP Multicast. *SIGCOMM Computer Communications Review, Vol 36, Issue 4, pp. 15-26*, October 2006.

[97] Y. Rekhter and T. Li. An Architecture for IP Address Allocation with CIDR. RFC 1518 (Historic), September 1993.

[98] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. Updated by RFCs 6286, 6608.

[99] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996.

[100] Jennifer Rexford. Internet Topology. *Princeton University, Course COS461: Computer Networks*, Spring 2006.

[101] Sajjad Rizvi, András Zahemszky, and Tuomas Aura. Scaling Bloom filter based multicast with hierarchical tree splitting. *IEEE International Conference on Communications (ICC), pp. 2790-2795*, June 2012.

[102] E. Rosen and R. Aggarwal. Multicast in MPLS/BGP IP VPNs. RFC 6513 (Proposed Standard), February 2012.

[103] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364 (Proposed Standard), February 2006. Updated by RFCs 4577, 4684, 5462.

[104] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001. Updated by RFC 6178.

[105] Christian Esteve Rothenberg, Carlos Macapuna, Fábio Verdi, Maurício Magalhães, and András Zahemszky. Data center networking with in-packet Bloom filters. *28th Brazilian Symposium on Computer Networks (SBRC)*, May 2010.

[106] Matthew Roughan, Walter Willinger, Olaf Maennel, Debbie Perouli, and Randy Bush. 10 Lessons from 10 Years of Measuring and Modeling the Internet's Autonomous Systems. *IEEE Journal on Selected Areas in Communications, Vol 29, Issue 9, pp. 1810-1821*, October 2011.

[107] J. Saltzer. On the Naming and Binding of Network Destinations. RFC 1498 (Informational), August 1993.

[108] Mikko Särelä, Jörg Ott, and Jukka Ylitalo. Fast Inter-domain Mobility with In-packet Bloom Filters. *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture, MobiArch '10, pp. 9-14*, September 2010.

[109] Mikko Särelä, Christian Esteve Rothenberg, Tuomas Aura, András Zahemszky, Pekka Nikander, and Jörg Ott. Forwarding Anomalies in Bloom Filter-Based Multicast. *IEEE INFOCOM, pp. 2399-2407*, April 2011.

[110] Mikko Särelä, Christian Esteve Rothenberg, András Zahemszky, Pekka Nikander, and Jörg Ott. BloomCasting: Security in Bloom Filter Based Multicast. *15th Nordic Conference in Secure IT Systems (NordSec), pp. 1-16*, October 2010.

[111] John F. Shoch. Inter-Network Naming, Addressing, and Routing. *Internet Experiment Note no. 19*, January 1978.

[112] V. Sivaraman, A. Vishwanath, Zhi Zhao, and C. Russell. Profiling per-packet and per-byte energy consumption in the NetFPGA Gigabit router. *IEEE Conference on Computer Communications Workshops, pp. 331-336*, April 2011.

[113] F. Solano, R. Van Caenegem, D. Colle, J.L. Marzo, M. Pickavet, R. Fabregat, and P. Demeester. All-Optical Label Stacking: Easing the Trade-offs Between Routing and Architecture Cost in All-Optical Packet Switching. *The 27th Conference on Computer Communications (INFOCOM 2008), pp. 1328-1336*, April 2008.

[114] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001.

[115] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational), August 1999.

[116] W.R. Stevens, B. Fenner, and A.M. Rudoff. UNIX Network Programming. *Addison Wesley Professional*, 2004.

[117] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. *IEEE/ACM Transactions on Networking, Vol 12, Issue 2, pp. 205-218*, April 2004.

[118] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *SIGCOMM Computer Communications Review, Vol 31, Issue 4, pp. 149-160*, August 2001.

[119] T. Takeda. Framework and Requirements for Layer 1 Virtual Private Networks. RFC 4847 (Informational), April 2007.

[120] S. Tarkoma, C.E. Rothenberg, and E. Lagerspetz. Theory and Practice of Bloom Filters for Distributed year. *IEEE Communications Surveys Tutorials, Vol 14, Issue 1, pp. 131-155*, February 2012.

[121] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 5065 (Draft Standard), August 2007.

[122] D. Trossen, G. Parisis, B. Gajic, J. Riihijarvi P. Flegkas, P. Sarolahti, P. Jokela, X. Vasilakos, C. Tsilopoulos, S. Arianfar, and M. Reed. PURSUIT Deliverable 2.3: Architecture Definition, Components Descriptions and Requirements. *EU, PURSUIT-project, FP7-INFSO-ICT 257217*, October 2011.

[123] Christos Tsilopoulos and George Xylomenos. Scaling Bloom filter-based multicast via filter switching. *IEEE Symposium on Computers and Communications (ISCC), pp. 548-553*, July 2013.

[124] Rodney S. Tucker, J. Baliga, R.W.A. Ayre, K. Hinton, and W.V. Sorin. Energy consumption in IP networks. *34th European Conference on Optical Communication (ECOC 2008)*, September 2008.

[125] Weizhen Yang, Dirk Trossen, and Janos Tapolcai. Scalable forwarding for information-centric networks. *IEEE International Conference on Communications (ICC), pp. 3639-3644*, June 2013.

[126] Jukka Ylitalo, Petri Jokela, Jorma Wall, and Pekka Nikander. End-point identifiers in secure multi-homed mobility. In *IN PROCEEDINGS OF OPODIS 02*, pages 17–28, 2002.

[127] Minlan Yu, Alex Fabrikant, and Jennifer Rexford. BUFFALO: Bloom Filter Forwarding Architecture for Large Organizations. *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09), pp. 313-324*, December 2009.

[128] A. Zahemszky and S. Arianfar. Fast Reroute for Stateless Multicast. *International Conference on Ultra Modern Telecommunications Workshops (ICUMT '09), pp. 1-6*, October 2009.

[129] Xing Zheng and Xiaojun Wang. Comparative study of power consumption of a NetFPGA-based forwarding node in publish-subscribe Internet routing. *Computer Communications, Vol 44, pp. 36-43*, May 2014.

Aalto-DD 108/2017

ISBN 978-952-60-7470-2 (printed)
ISBN 978-952-60-7469-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

**Aalto University**
**School of Electrical Engineering**
**Department of Communications and Networking**
**www.aalto.fi**

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

**DOCTORAL
DISSERTATIONS**