IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 4, 2021, accepted February 19, 2021, date of publication February 24, 2021, date of current version March 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3061763

Hardware-Based Evaluation of Scalable and Resilient Multicast With BIER in P4

DANIEL MERLING, STEFFEN LINDNER, AND MICHAEL MENTH, (Senior Member, IEEE)
Chair of Communication Networks, University of Tuebingen, 72076 Tübingen, Germany

Corresponding author: Daniel Merling (daniel.merling@uni-tuebingen.de)

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant ME2727/1-2. The authors alone are responsible for the content of this paper.

ABSTRACT Traditional IP multicast (IPMC) maintains state per IPMC group in core devices to distribute one-to-many traffic along tree-like structures through the network. This limits its scalability because whenever subscribers of IPMC groups change, forwarding state in the core network needs to be updated. Bit Index Explicit Replication (BIER) has been proposed by the IETF for efficient transport of IPMC traffic without the need of IPMC-group-dependent state in core devices. However, legacy devices do not offer the required features to implement BIER. P4 is a programming language which follows the software-defined networking (SDN) paradigm. It provides a programmable data plane by programming the packet processing pipeline of P4 devices. The contribution of this article is threefold. First, we provide a hardware-based prototype of BIER and BIER fast reroute (BIER-FRR) which leverages packet recirculation. Our target is the P4-programmable high-performance switching ASIC Tofino; the source code is publicly available. Second, we perform an experimental evaluation, with regard to failover time and throughput, which shows that up to 100 Gb/s throughput can be obtained and that failures affect BIER forwarding for less than 1 ms. However, throughput can decrease if switch-internal packet loss occurs due to missing recirculation capacity. As a remedy, we add more recirculation capacity by turning physical ports into loopback mode. To quantify the problem, we derive a prediction model for reduced throughput whose results are in good accordance with measured values. Third, we provide a provisioning rule for recirculation ports, that is applicable to general P4 programs, to avoid switch-internal packet loss due to packet recirculation. In a case study we show that BIER requires only a few such ports under realistic mixes of unicast and multicast traffic.

INDEX TERMS Software-defined networking, P4, bit index explicit replication, multicast, resilience, scalability.

I. INTRODUCTION

IP multicast (IPMC) has been proposed to efficiently distribute one-to-many traffic, e.g. for IPTV, multicast VPN, commercial stock exchange, video services, public surveillance data distribution, emergency services, telemetry, or content-delivery networks, by forwarding only one packet per link. IPMC traffic is organized in IPMC groups which are subscribed by hosts. Figure 1 shows the concept of IPMC. IPMC traffic is forwarded on IPMC-group-specific distribution trees from the source to all subscribed hosts. To that end, core routers maintain forwarding state for each IPMC group to determine the next-hops (NHs) of an IPMC packet. Scalability issues are threefold. First, a significant amount of storage is required to keep extensive forwarding state. Second, when subscribers of an IPMC group change, the distribution tree needs to be updated by signaling the changes to core devices. Third, the distribution trees have to

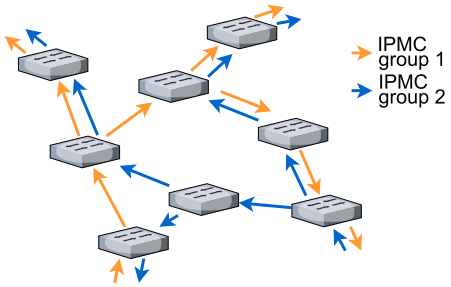

Figure 1. Two multicast distribution trees.

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

be updated when the topology changes or a failure is detected. Therefore, traditional IPMC comes with significant management and state overhead. As a result, traditional IPMC is often avoided and multicast is implemented on the application layer. Thereby, one-to-many traffic is carried via network layer unicast, which is not efficient.

The IETF proposed Bit Index Explicit Replication (BIER) [1] for efficient transport of IPMC traffic. BIER introduces a BIER domain where core routers do not need to maintain IPMC-group-dependent state. Upon entering the BIER domain, IPMC packets are equipped with a BIER header which specifies all destinations of the packet within the BIER domain. The BIER packets are forwarded through the BIER domain towards their destinations on paths from the Interior Gateway Protocol (IGP), which we call 'routing underlay' in the following. Thereby, only one packet is forwarded per link. When the BIER packets leave the BIER domain, the BIER header is removed.

Unicast and BIER traffic may be affected by failures. IP-Unicast traffic is often protected by fast reroute (FRR) mechanisms for IP (IP-FRR). IP-FRR leverages precomputed backup entries to quickly reroute a packet on a backup path when the primary NH is unreachable. Tunnel-based BIER-FRR [2] is used to protect BIER traffic by tunneling BIER packets through the routing underlay. The tunnel may be also affected by a failure, but FRR or timely updates of the forwarding information base (FIB) in the routing underlay quickly restore connectivity. However, BIER is not supported by legacy devices and there is no dedicated BIER hardware available. P4 [3] is a programming language that follows the software-defined networking (SDN) paradigm for programming protocol-independent packet processors. P4 allows developers to write high-level programs to define the packet processing pipeline of programmable network devices. A target-specific compiler translates the P4 program for execution on a particular device. With the P4-programmable data plane new protocols can be implemented and deployed in short time.

In previous work [2], [4] we implemented BIER and tunnel-based BIER-FRR for the P4 software switch bmv2 [5]. However, the developers of the bmv2 clarify that the 'BMv2 is not meant to be a production-grade software switch' [5] and is, therefore, only a 'tool for developing, testing and debugging P4 data planes' [5]. Thus, it remains unclear whether BIER and BIER-FRR forwarding is simple enough to be implemented also on P4-capable hardware platforms which entail functional and runtime constraints to achieve high-speed forwarding.

The contribution of this article is threefold. First, we provide a new prototype for BIER and BIER-FRR on the P4-programmable switching ASIC Tofino [6] which is used in the Edgecore Wedge 100BF-32X [7], a 32 100 Gb/s port high-performance P4 switch, and make our code publicly available.

Second, we conduct an experimental performance study with regard to failover time and throughput. The evaluations show that connectivity can be restored within less than 1 ms and that a throughput of up to 100 Gb/s can be obtained. However, we observe reduced throughput under certain conditions and conjecture that this results from switch-internal packet loss due to missing recirculation capacity. We add more recirculation capacity by turning physical ports into loopback mode to avoid switch-internal packet loss in case of recirculation. To quantify the problem, we derive a prediction model for BIER throughput whose results are in good accordance with measured values.

Third, we propose a provisioning rule for recirculation ports to avoid switch-internal packet loss due to packet recirculation. It is applicable to general P4 programs and helps to avoid throughput reduction on outgoing links. Finally, we utilize the provisioning model to show in a case study that only a few ports in loopback mode suffice to avoid internal packet loss with BIER under realistic mixes of unicast and multicast traffic.

The paper is structured as follows. In Section II we describe related work. Section III contains a primer on BIER and tunnel-based BIER-FRR. Afterwards, we give an overview on P4 in Section IV and explain important properties. In Section V, we briefly describe the P4 implementation of BIER and tunnel-based BIER-FRR for the Tofino. Section VI contains our evaluation and the model for throughput prediction of BIER. In Section VII we present a model to provision recirculation ports. We conclude the paper in Section VIII.

II. RELATED WORK

First, we describe related work for SDN-based multicast in general. Then, we review work for BIER-based multicast. Finally, we present P4 projects that are based on packet recirculation.

A. SDN-BASED MULTICAST

Elmo [8] increases scalability of traditional IPMC in data center environments by leveraging characteristics of data center networks, in particular symmetric topologies and short paths. By encoding multicast group information in the packet header, this information is no longer stored in forwarding devices. This significantly reduces the dynamic state that needs to be maintained by core nodes.

Two surveys [9], [10] provide a comprehensive overview of SDN-based multicast. They review the development of traditional multicast and different aspects of SDN-based multicast, e.g., building of distribution trees, group management, and approaches to improve the efficiency of multicast. Most of the papers in the surveys discuss multicast mechanisms that are based on explicit IPMC-group-dependent state in core devices. The downsides of those traditional IPMC approaches have been discussed in Section I. We still discuss some papers on IPMC due to their efforts to make traditional IPMC more efficient. The papers often focus on intelligent tree building mechanisms that reduce the state, or efficient signaling techniques when IPMC groups or the topology changes. The surveys also consider works that utilize SDN to

improve multicast. They are related as our approach also takes an SDN approach. Therefore, we present some representative examples from that area.

1) OPTIMIZATION OF MULTICAST TREES

Rückert *et al.* propose Software-Defined Multicast (SDM) [11]. SDM is an OpenFlow-based platform that provides well-managed multicast for over-the-top and overlay-based live streaming services tailored for P2P-based video stream delivery. The authors extend SDM in [12] with traffic engineering capabilities. In [13] the authors propose address translation from the multicast address to the unicast address of receivers at the last multicast hop in OpenFlow switches. This reduces the number of IPMC-group-dependent forwarding entries in some nodes.

Steiner trees are often used to build multicast distribution trees [14]. Several papers modify the original Steiner-tree problem to build distribution trees with minimal cost [15], number of edges [16], number of branch nodes [17], delay [18], or for optimal position of the multicast source [19].

The authors of [20] implement a multicast platform in OpenFlow with a reduced number of forwarding entries. It is based on multiple shared trees between different IPMC groups. The Avalanche Routing Algorithm (AvRA) [21] considers properties of the topology of data center networks to build trees with optimal utilization of network links. Dual-Structure Multicast (DuSM) [22] leverages different forwarding structures for high-bandwidth and low-bandwidth flows. This improves scalability and link utilization of SDN-based data centers. Jia *et al.* [23] present a way to efficiently organize forwarding entries based on prime numbers and the Chinese remainder theorem. This reduces the required state in forwarding devices and allows more efficient implementation. In [24] the authors propose a SDN-based multicast switching system that leverages bloom filters to reduce the number of TCAM-entries.

2) RESILIENCE FOR TRADITIONAL MULTICAST

Shen *et al.* [25] modify Steiner trees to include recovery nodes in the multicast distribution tree. The recovery nodes cache IPMC traffic temporarily and resend it after reconvergence when the destination notified the recovery point because it did not get all packets due to a failure. The authors of [26] evaluate several algorithms that generate node-redundant multicast distribution trees. They analyse the number of forwarding entries and the effect of node failures. In [27] the authors propose to deploy primary and backup multicast trees in SDN networks. The header of multicast packets contains an ID that identifies the distribution tree on which the packet is forwarded. When a failure is detected, the controller reconfigures affected sources to send packets along a working backup tree. Pfeiffenberger *et al.* [28] propose a similar method. Each node that is part of a distribution tree is the root of a backup tree that does not contain the unreachable NH but all downstream destinations of the primary distribution tree. When a node cannot forward a packet, it reroutes the packet on a backup tree by switching an VLAN tag in the packet header.

B. BIER-BASED MULTICAST

In this subsection we discuss work directly related to BIER. First, we define our work in contrast to other implementations. Then, we describe evaluations and extensions for BIER.

1) IMPLEMENTATIONS

We started with an implementation of BIER for the software switch bmv2 using $P4_{14}$. The protoype was documented at high level in a 2-page demo paper [4]. We then developed BIER-FRR and implemented a prototype for BIER and BIER-FRR on the software switch bmv2 using the newer variant $P4_{16}$ in [2]. That work demonstrated that the P4 language is expressive enough to implement also complex forwarding mechanisms and introduced a hierarchical controller hierarchy to quickly trigger FRR actions. The study compared restoration times for various failure cases and protection schemes at light load conditions of a few packets per second. Throughput measurements were not conducted as the bmv2 software switch is only a 'tool for developing, testing and debugging P4 data planes' [5] with low throughput (900 Mb/s) [29] and not for application in real networks. In contrast, this paper shows that BIER and BIER FRR can be implemented also on high-performance P4-programmable hardware, i.e., the switching ASIC Tofino, which entails additional functional and runtime constraints for implementations to achieve high throughput. Experimental measurement studies in a 100 Gb/s hardware testbed reveal performance challenges due to recirculations. As this is a general problem for some P4 programs, we derive recommendations to cope with them and validate them in our hardware testbed.

We know only a single BIER implementation by other authors which is based on OpenFlow and presented in [30], [31]. Their approach suffers from two major shortcomings. First, the BIER bit string is encoded in a MPLS header which is the only way to encode arbitrary bit strings in OpenFlow. This limits the bit string length, and thus the number of receivers, to 20 which is the length of an MPLS label. Second, the implementation performs an exact match on the bitstring. If a subscriber changes, the match does not work anymore and a local BIER agent that is not part of the OpenFlow protocol needs to process the packet. Therefore, we consider this project only as an early BIER-based prototype for OpenFlow and not as a production-ready BIER implementation.

2) EVALUATIONS AND EXTENSIONS OF BIER-BASED MULTICAST

The authors of [32] perform a simulation-based evaluation of BIER. They find that on metrics like delivery ratios and retransmissions BIER performs as well as traditional IPMC but has better link usage and no per-flow or per-group state in core devices.

Eckert *et al.* [33] propose an extension for BIER that allows for traffic engineering (BIER-TE). In addition to the egress nodes, the BIER header encodes the distribution tree of a packet. In [34] the authors propose 1 + 1 protection for BIER-TE. The traffic is transported on two disjoint distribution trees, which delivers the traffic even if one tree is interrupted by a failure.

C. PACKET RECIRCULATION IN P4

Hauser *et al.* [35] show in their P4 survey that packet recirculation is not used only in this BIER implementation but also in other P4 projects. In [36] the authors implement a congestion control mechanism in P4 and leverage packet recirculation to create notification packets, update their header fields, and send them to appropriate monitoring nodes. The authors of [37] present a content-based publish/subscribe mechanism in P4 where they introduce a new header stack that requires packet recirculation for processing. Uddin *et al.* [38] implement multi-protocol edge switching for IoT based on P4. Packet recirculation is used to process packets a second time after they have been decrypted.

III. BIT INDEX EXPLICIT REPLICATION (BIER)

In this Section we explain BIER. First, we give an overview. Then we describe the BIER forwarding table and how BIER packets are processed. Afterwards, we show a forwarding example. Finally, we review tunnel-based BIER-FRR.

A. BIER OVERVIEW

First, we introduce the BIER domain. Then, we present the layered BIER architecture followed by the BIER header. Finally, we describe BIER forwarding.

1) BIER DOMAIN

Figure 2 shows the concept of the BIER domain. When bit-forwarding ingress routers (BFIRs) receive an IPMC packet they push a BIER header onto it and forward the packet into the BIER domain. The BIER header identifies all destinations of the BIER packet within the BIER domain, i.e., bit-forwarding egress routers (BFERs). Bit-forwarding routers (BFRs) forward the BIER packets to all BFERs indicated in its BIER header. Thereby, packets are replicated and forwarded to multiple next-hops (NHs) but only one packet is sent over any involved link. The paths towards the destinations are provided by the Interior Gateway Protocol (IGP), i.e., the routing underlay. Therefore, from a specific BFIR to a specific BFER, the BIER packet follows the same path as unicast traffic. Finally, BFERs remove the BIER header.

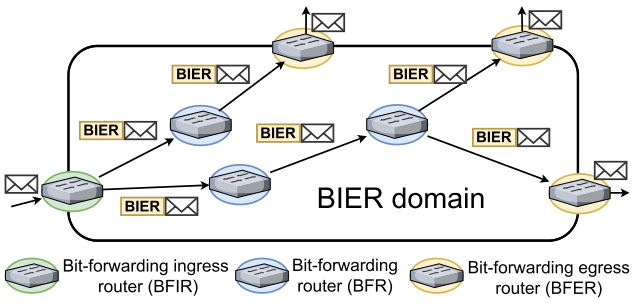

Figure 2. The concept of the BIER domain [39].

2) THE LAYERED BIER ARCHITECTURE

The BIER architecture consists of three components. The IPMC layer, the BIER layer and the routing underlay. Figure 3 shows the three layers, their composition, and interaction. The IPMC layer contains the sources and subscribers of IPMC traffic. The BIER layer acts as a transport layer for IPMC traffic. It consists of the BIER domain which is connected to the IPMC layer at the BFIRs, and BFERs. Therefore, the BIER layer acts as a point-to-multipoint tunnel from an IPMC source to multiple subscribers. The routing underlay refers to the IGP which provides the paths to all destinations within the network.

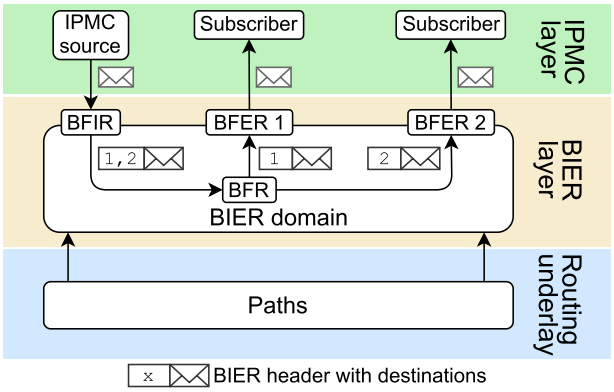

Figure 3. IPMC packets are transmitted over a layered BIER architecture; the paths are defined by the information from the routing underlay [39].

3) BIER HEADER

The BIER header contains a bit string to indicate the destinations of a BIER packet. To that end, each BFER is assigned an unique number that corresponds to a bit position in that bit string, starting by 1 for the least-significant bit. If a BFER should receive a copy of the IPMC packet, its bit is activated in the bit string in the BIER header of the packet. To facilitate readability we refer to the bit string in the BIER header of a BIER packet with the term 'BitString'.

4) BIER FORWARDING

A BFR forwards a packet copy to any neighbor over which at least one destination of the packet indicated by its BitString is reached according to the paths from the routing underlay. Before a packet is forwarded to a specific NH, the BFR clears all bits that correspond to BFERs that are reached via other NHs from the BitString of that packet. This prevents duplicates at the BFERs.

B. BIFT STRUCTURE

BFRs use the Bit Index Forwarding Table (BIFT) to determine the NHs of a BIER packet. Table 1 shows the BIFT of BFR 1 from Figure 4. For each BFER there is one entry in the BIFT. Entries of the BIFT consist of a NH, and a so-called F-BM. The F-BM is a bit string similar to the BitString. It records which BFERs have the same NH. In the F-BM of an BIFT entry the bits of BFERs are activated which are reached over the NH of that entry. Therefore, BFERs with the same NH have the same F-BM. BFRs use the F-BM to clear bits from the BitString of a packet before it is forwarded to a NH.

Table 1. BIFT of BFR 1 in the example of Figure 4 [39].

BFER	NH	F-BM
1	-	-
2	2	1010
3	3	0100
4	2	1010

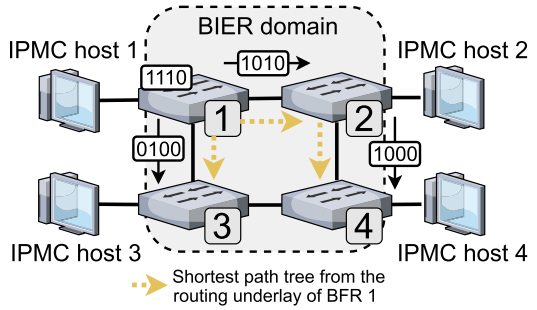

Figure 4. Example of a BIER topology and BitStrings of forwarded BIER packets [39].

C. BIER PACKET PROCESSING

When a BFR receives a BIER packet, it first stores the BitString of the packet in a separate bit string to account to which BFERs a packet has to be sent. In the following, we refer to that bit string with the term 'remaining bits'. The following procedure is repeated, until the remaining bits contain no activated bits anymore [1].

The BFR determines the least-significant activated bit in the remaining bits. The BFER that corresponds to that bit is used for a lookup in the BIFT. If a matching entry is found, it results in a NH *nh* and the F-BM *fbm* and the BFR creates a copy of the BIER packet. The BFR uses *fbm* to clear bits from the BitString of the packet copy. To that end, the BFR performs a bitwise AND operation of *fbm* and the BitString of the packet copy and writes the result into the BitString of the packet copy. This procedure is called applying the F-BM. It leaves only bits of BFERs in the BitString active that are reached over *nh*. The packet copy is then forwarded to *nh*. Afterwards, the bits of BFERs to which a packets has just been sent are cleared from the remaining bits. To that end, the BFR performs a bitwise AND operation of the bitwise complement of *fbm* with the remaining bits. The result is then stored in the remaining bits.

D. BIER FORWARDING EXAMPLE

Figure 4 shows a topology with four BIER devices where each is BFIR, BFR, and BFER. Table 1 shows the BIFT of BFR 1.

BFR 1 receives an IPMC packet from IPMC host 1 which should be distributed to all other IPMC hosts. Therefore, BFIR 1 pushes a BIER header with the BitString `1110` to the IPMC packet.

Then, BFR 1 determines the least-significant activated bit in the BIER header which corresponds to BFER 2. This BFER is used for lookup in the BIFT, which results in the F-BM `1010` and the NH BFR 2. BFR 1 creates a packet copy and applies the F-BM to its BitString. Then, the packet copy with the BitString `1010` is forwarded to BFR 2. Finally, the activated bits of the F-BM are cleared from the remaining bits which leaves the bit string `0100`.

This leaves only one bit active which identifies BFER 3. After the F-BM `0100` is applied to the BitString of a packet copy, it is forwarded to BFR 3 with the BitString `0100`. After clearing the bits of the F-BM from the remaining bits, processing stops because no active bits remain.

E. TUNNEL-BASED BIER-FRR

Tunnel-based BIER-FRR is used to deliver BIER traffic even when NHs are unreachable due to link or node failures. When a BFR detects that a NH is unreachable, e.g., by loss-of-carrier, loss-of-light, or a bidirectional forwarding detection (BFD[1]) [40] for BIER [41], it becomes the point of local repair (PLR) by tunneling the BIER packet through the routing underlay to nodes downstream in the BIER distribution tree. The tunnel may be affected by the failure, too. However, FRR mechanisms or timely updates of the FIB in the routing underlay restore connectivity for unicast traffic faster than for BIER traffic because recomputation of BIER entries can start only after the FIB of the routing underlay has been updated. Tunnel-based BIER-FRR can be configured either for link protection or node protection. BIER-FRR with link protection tunnels the BIER packet to the NH where the tunnel header is removed and the BIER header is processed again. BIER-FRR with node protection tunnels copies of the BIER packets to all next-next-hops (NNHs) in the distribution tree.

IV. INTRODUCTION TO P4

In this section we briefly review fundamentals of P4 [3]. First, we give an short overview of the P4 processing pipeline. Afterwards, we explain packet cloning and packet recirculation and point out important properties.

A. P4 PIPELINE

In this subsection we review the P4 processing pipeline. We explain its composition, transient and persistent memory, match + action tables, control blocks, packet cloning

[1]When a BFR is established between two nodes, they periodically exchange notifications about their status.

and packet recirculation. Figure 5 shows the concept of the P4 processing pipeline.

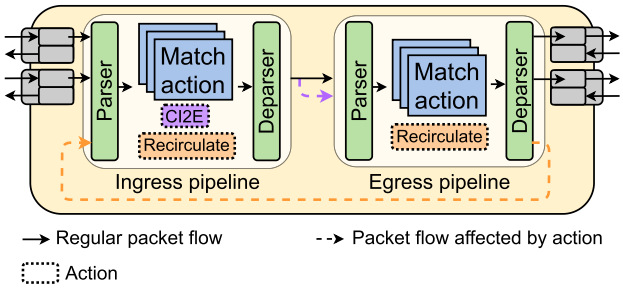

Figure 5. P4 processing pipeline.

1) COMPOSITION

The P4 pipeline consists of an ingress pipeline and an egress pipeline. They process packets in a similar fashion, i.e., both contain a parser, a match + action pipeline, and a deparser. When a packet arrives at the switch, it is first processed by the ingress pipeline. The header fields of the packet are parsed and carried along with the packet through the ingress pipeline. The parser is followed by a match + action pipeline which consists of a sequence of conditional statements, table matches, and primitive operations. Afterwards, the packet is deparsed and sent to the egress pipeline for further processing. Finally, the packet is sent through the specified egress port which has to be set in the ingress pipeline and cannot be changed in the egress pipeline.

The P4 program defines the parser and the deparser, which allows the use of custom packet headers. In addition, the P4 program describes the control flow of the match + action pipeline in the ingress pipeline and egress pipeline, respectively.

2) CONTROL BLOCKS

Both the ingress and egress pipeline can be divided into so-called control blocks for structuring. Control blocks are used to clearly separate functionality for different protocols like IP, BIER, and Ethernet, i.e., the IP control block contains Match + Action Tables (MATs) and operations that are applied only to IP packets, etc. In this paper we focus only on the BIER control block.

3) Match+Action TABLES (MATs)

MATs execute packet-dependent actions by matching packet header fields against MAT entries. To that end, an entry contains one or more match fields, and an action set. When a packet is matched against a MAT, the match fields of the entries are compared with specified header fields of the packet. An action set consists of one or more actions, e.g., reading or writing a header field, mathematical operations, setting the egress port of the packet, etc. It is not possible to match a packet on the same MAT multiple times.

B. PACKET CLONING

The operation clone-ingress-to-egress (CI2E) allows packet replication in P4. It can be called only in the ingress pipeline. At the end of the ingress pipeline, a copy of the packet is created. However, the packet copy resembles the packet that has been parsed in the beginning of the ingress pipeline, i.e., the header changes performed during processing in the ingress pipeline are reverted. This is illustrated in Figure 6.

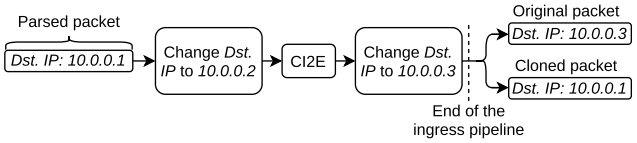

Figure 6. An example of the clone-ingress-to-egress (CI2E) operation [39].

If an egress port has been provided as a parameter, the egress port of the clone is set to that port. Both the original and cloned packet are processed independently in the egress pipeline. The cloned packet carries a flag to identify it as a clone.

C. PACKET RECIRCULATION

In this subsectin we explain the packet recirculation operation. First, we explain its working. Afterwards, we introduce the term recirculation capacity.

1) FUNCTIONALITY

P4 allows to recirculate a packet for processing it by the pipeline a second time. We use this feature to implement the iterative packet processing of BIER as described in Section III-C as P4 offers no other possibility to implement processing loops.

P4 leverages a switch-intern recirculation port for packet recirculation. When a packet should be recirculated, its egress port has to be set to the recirculation port during processing in the ingress pipeline. The flow of a packet through the pipeline when it is recirculated is shown in Figure 7. The packet is still processed by the entire processing pipeline, i.e., the ingress pipeline and egress pipeline. However, after the packet has been deparsed, it is not sent through a regular physical egress port but pushed back into the switch-intern recirculation port. The packet is then processed as if it has been received on a physical port. The recirculation port has the same capacity

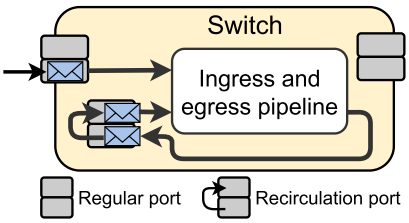

Figure 7. A packet is recirculated to a recirculation port and traverses the ingress and egress pipeline for a second time.

as the physical ports. For example, when two physical ports receive traffic at line rate and each packet is recirculated once, the recirculation port receives recirculated packets at double line rate, which causes packet loss.

2) RECIRCULATION CAPACITY

To discuss the effect of packet loss due to many recirculations we introduce the term 'recirculation capacity'. It is the available capacity to process recirculation traffic. Additional recirculation capacity is provided by using physical ports in loopback mode. When the forwarding device switches a packet to an egress port that is configured as a loopback port, the packet is immediately placed in the ingress of that port, instead. The packet is then processed as if it has been received on that port as usual, i.e., by the parser, the ingress and egress pipeline, and the deparser. Only traffic that has to be recirculated is switched to recirculation ports. In the following the term 'recirculation port' refers to a physical port in loopback mode, or the switch-intern recirculation port. When recirculation ports are required, the switch-intern recircution port should be used first, before any physical ports are configured as loopback ports. Only packets that are recirculated require recirculation capacity, i.e., common unicast traffic, e.g., as in regular IP unicast forwarding, is not recirculated, and therefore, does not occupy any recirculation capacity.

When multiple recirculation ports are deployed to increase the recirculation capacity, packets that should be recirculated need to be distributed over these ports. There are different distribution strategies. We developed a round-robin-based distribution approach for recirculation traffic to distribute the load equally over all recirculation ports. We store in a register which recirculation port receives the next packet which should be recirculated. When a packet has to be sent to a recirculation port, that register is accessed and updated in one atomic operation. This prevents any race conditions when traffic is distributed. Thus, this distribution strategy has two advantages. First, if n recirculation ports are used, the available recirculation capacity is increased to $n \cdot linerate$. Second, the equal distribution of recirculation traffic over all recirculation ports guarantees the full utilization of available recirculation capacities before packet loss occurs.

V. P4 IMPLEMENTATION OF BIER AND BIER-FRR FOR TOFINO

In this section we give an overview of the P4 implementation of BIER and tunnel-based BIER-FRR. First, we discuss the implementation basis. Afterwards, we give an overview of the processing of BIER packets, in particular we discuss packet recirculation.

A. CODEBASE

In [2] we presented a software-based prototype of a $P4_{16}$ implementation of BIER and tunnel-based BIER-FRR for the P4 software switch bmv2. We provided a very detailed description of the P4 programs including MATs with match fields and action parameters, control blocks, and applied operations. The prototype and the controller are publicly available on GitHub.[2]

In this paper we refrain from including a detailed technical description of the implementation for the Tofino. However, the source code[3] can be accessed by anyone on GitHub. In the following, we only explain important aspects of the hardware-based implementation to facilitate the understanding of the evaluation in Section VI and the model derivations in Section VII.

B. BIER PROCESSING

First, we describe the implementation of regular BIER forwarding on the Tofino. Afterwards, we explain operation of tunnel-based BIER-FRR.

1) BIER FORWARDING

Figure 8 shows how a BIER packet is processed once in the packet processing pipeline.

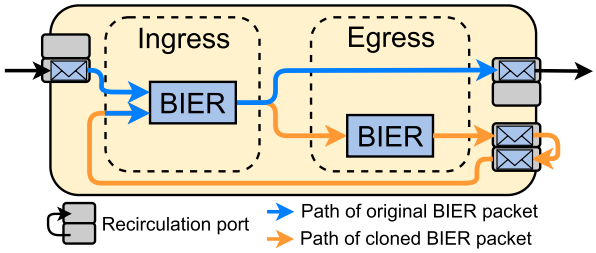

Figure 8. Paket flow of a BIER packet in the processing pipeline.

When the switch receives a BIER packet it is processed by the BIER control block. First, the BitString of the packet is matched against the BIFT which determines the egress port and the F-BM. The F-BM is applied to the BitString of the packet and cleared from the remaining bits. If the remaining bits still contain activated bits, CI2E is called and the egress port is set to a recirculation port so that the packet will be processed again. After the ingress pipeline, the copy is created and both packet instances enter the egress pipeline independently of each other. The original packet is sent through an egress port towards its NH. The packet clone is processed by a second BIER control block in the egress pipeline which sets the BitString of the packet copy to the remaining bits. Since the egress port of the packet clone is a recirculation port, the packet is recirculated, i.e., it is processed by the ingress pipeline again.

BIER forwarding removes BIER headers from packets that leave the BIER domain, and adds IP headers for tunneling through the routing underlay by tunnel-based BIER-FRR. Whenever a header is added or removed, the packet is recirculated for further processing.

When a BIER packet has more than one NH, two challenges appear. First, the BitString of a BIER packet has to be

[2]https://github.com/uni-tue-kn/p4-bier
[3]https://github.com/uni-tue-kn/p4-bier-tofino

matched several times against the BIFT to determine all NHs. However, matching a packet multiple times against the same MAT is not possible in P4. Second, multiple packet copies have to be created for forwarding. However, P4 does not allow to dynamically generate more than one copy of a packet. Therefore, we implemented a packet processing behavior where in each pipeline iteration one packet is forwarded to a NH and a copy of the packet is recirculated for further processing. This is repeated until all NHs receive a packet over which at least one destination of the BIER packet is reached. Figure 9 shows the processing of a BIER packet which has to be forwarded to three neighbors. In the first and second pipeline iteration the original BIER packet is sent through a physical egress port towards a NH and the copied BIER packet is recirculated by sending the packet copy to a recirculation port. In the last iteration when the remaining bits contain no activated bits anymore, no further packet copy is required and only the original BIER packet is sent through the egress port. In total, the packet needs to be recirculated two times to forward it to all three NHs. Therefore, in general, a BIER packet with n NHs, has to be recirculated $n-1$ times and the first NH can be served without packet recirculation.

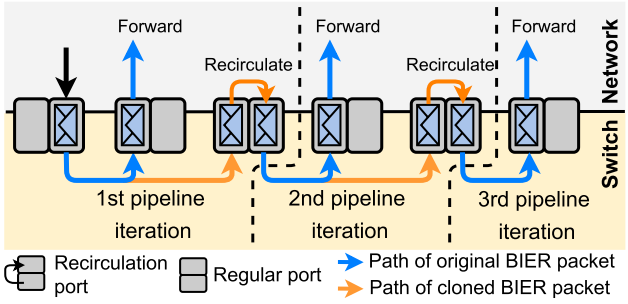

Figure 9. BIER processing over multiple pipeline iterations.

2) FORWARDING WITH TUNNEL-BASED BIER-FRR

The concept of tunnel-based BIER-FRR has been proposed in [2]. We implement it for the Tofino as follows.

The switch monitors the status of its ports as described in Section. When the match on the BIFT results in a NH which is reached by a port that is currently down, the processing of the BIER packet differs in the following way from the BIER processing described above. An IP header is added to the original BIER packet to tunnel the packet through the routing underlay towards an appropriate node in the BIER distribution tree. The egress port of the original packet is set to a recirculation port to process the IP header in another pipeline iteration, i.e., forward the IP packet to the right NH.

VI. PERFORMANCE EVALUATION OF THE P4-BASED HARDWARE PROTOTYPE

In this section we perform experiments to evaluate the performance of the P4-based hardware prototype for BIER regarding Layer-2 throughput and failover time, i.e., the time until BIER traffic is successfully delivered after a network failure.

A. FAILOVER TIME FOR BIER TRAFFIC

Here we evaluate the restoration time after a failure in three scenarios and vary the protection properties of IP and BIER. First, only the IP FIB and BIER FIB are updated by the controller, respectively, and no FRR mechanisms are activated. This process is triggered by a device that detects a failure. It notifies the controller which computes new forwarding rules and updates the IP and BIER FIB of affected devices. This scenario measures the time until the BIER FIB is updated after a failure, which is our baseline restoration time. The control plane, i.e., the controller, is directly connected to the P4 switch, which keeps the delay to a minimum in comparison to networks where the controller is several hops away.

Second, only BIER-FRR is deployed. In this scenario BIER is able to utilize tunnel-based BIER-FRR in case of a failure. However, FRR for IP traffic remains deactivated. Thus, IP traffic can be forwarded only after the IP FIB is updated.

Third, both IP-FRR and BIER-FRR are deployed. This scenario evaluates how quickly the P4 switch can react to network failures and restore connectivity of BIER and IP forwarding.

In the following, we first explain the setup and the metric. Then, we present our results. Finally, we discuss the influence of the setup on the results.

1) EXPERIMENT SETUP

Figure 10 shows the testbed. The Tofino [6], a P4-programmable switching ASIC, is at the core of the hardware testbed. We utilize a Tofino based Edgecore Wedge 100BF-32X [7] switch with 32 100 Gb/s ports. An EXFO FTB-1 Pro [42] 100 Gb/s traffic generator is connected to the Tofino to generate a data stream that is as precise as possible. Furthermore, we deploy two bmv2s that act as BFRs and BFERs. The traffic generator, the controller and two bmv2s are connected to the Tofino. The traffic generator sends IPMC traffic to the Tofino. The IPMC traffic has been subscribed only by bmv2-1. As long as the link between the Tofino and bmv2-1 works, the BIER packets are forwarded on the primary path. When the Tofino detects a failure, it notifies the

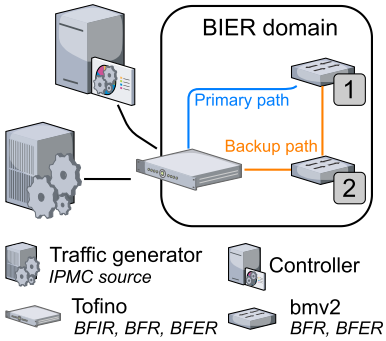

Figure 10. Experimental setup for evaluation of restoration time.

controller which computes new rules and updates forwarding entries of affected devices. In the meantime, the Tofino uses BIER-FRR to protect BIER traffic, and IP-FRR to protect IP traffic if enabled. This causes the Tofino to forward traffic on the backup path via bmv2-2 towards bmv2-1.

2) METRIC

We disable the link between the Tofino and bmv2-1 and measure the time until bmv2-1 receives BIER traffic again. We evaluate different combinations with and without IP-FRR and with and without BIER-FRR. To avoid congestion on the bmv2 and the VMs, the traffic generator sends only with 100 Mb/s, which has no impact on the results.

Figure 11 shows the average restoration time for the different deployed protection scenarios based on 10 runs which we discuss in the following. Confidence intervals are given on the base of a confidence level of 95%.

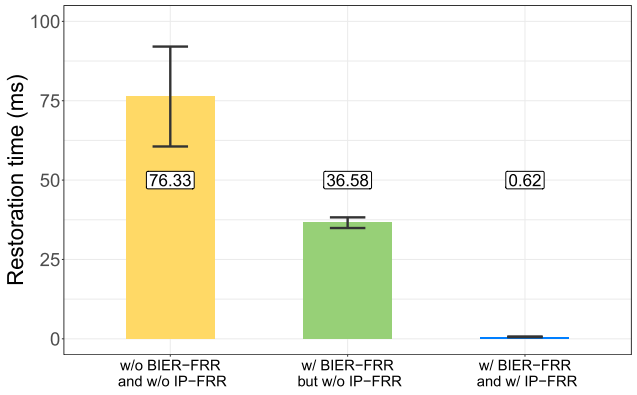

Figure 11. Restoration time for BIER with different FRR strategies.

3) FAILOVER TIME W/O BIER-FRR AND W/O IP-FRR

When no FRR mechanism is activated, multicast traffic arrives at the host only after the IP and BIER forwarding rules have been updated, which takes about 76 ms. The controller is directly connected to the Tofino. In a real deployment the controller may be multiple hops away, which would increase the restoration time significantly.

The same failover time is achieved without BIER-FRR but with IP-FRR, for which we do not present separate results. As BIER forwarding entries are updated only after IP forwarding entries have been updated, the use of IP-FRR in the network does not shorten the failover time for BIER traffic.

4) FAILOVER TIME W/BIER-FRR BUT W/O IP-FRR

When tunnel-based BIER-FRR but not IP-FRR is activated, bmv2-1 receives multicast traffic after 36 ms. In case of a failure, BIER-FRR tunnels the BIER traffic through the routing underlay. As soon as IP forwarding rules are updated, multicast traffic arrives at the host again. Since IP rules are updated faster than BIER rules, BIER-FRR decreases the restoration time for multicast traffic even if no IP-FRR mechanism is deployed.

5) FAILOVER TIME W/BIER-FRR AND W/IP-FRR

In the fastest and most resilient deployment both BIER-FRR and IP-FRR are activated. Then, multicast packets arrive at the host with virtually no delay after only 0.6 ms. In contrast to the previous scenario, unicast traffic is rerouted by IP-FRR which immediately restores connectivity for IP traffic.

6) INFLUENCE OF EXPERIMENTAL SETUP

The experimental setup (see Figure 10) features two BFERs on the base of bmv2 software switches with rather low performance compared to the Tofino-based hardware switch. However, we designed the experiment such that the low performance of these BFERs has no impact on results. bmv2 software switches can forward traffic with a rate up to 900 Mb/s [29]. By limiting the generated traffic rate to 100 Mb/s, the bmv2 switches forwarding and receiving BIER traffic are not overloaded so that bmv2-1 is able to measure correct restoration times. Furthermore, failure detection and protection switching are only carried out by the Tofino-based switch in the setup.

We now consider the impact of the hardware hosting the controller. When the controller is notified about a failure, it recomputes entries for IP and BIER forwarding tables. The computation time depends on the performance of the host and the size of the network in terms of number of nodes. Thus, the recomputation time may be significantly larger in larger networks, which increases the restoration time for BIER without any fast-reroute and for BIER with BIER-FRR but without IP-FRR. In contrast, the restoration time for BIER with BIER-FRR and IP-FRR is not impacted by the controller hardware or network size.

We discuss the impact of the signalling delay between the failure-detecting node and the controller. This delay was very low in our setup while it may be significantly larger in networks with large geographic extension or slow links. Such signalling delay adds to the restoration time for BIER without any fast-reroute and for BIER with BIER-FRR but without IP-FRR. The restoration time for BIER with BIER-FRR and IP-FRR is not impacted by that delay.

Finally, controller overload may occur when the controller needs to process too many messages, e.g., in case of a failure. This again has no impact on the restoration time for BIER with BIER-FRR and IP-FRR while it has significant impact on the restoration time for the other two settings.

B. THROUGHPUT FOR BIER TRAFFIC

The P4-based implementation of BIER described in Section V-B requires recirculation and is limited by the amount of recirculation capacity. The PSA defines a virtual port for this purpose. In this section we show the impact of insufficient recirculation capacity on throughput and the effect when additional physical recirculation ports, i.e., ports in loopback mode, are used for recirculation. We validate our experimental results in Section VI-C based on a theoretical model.

1) EXPERIMENTAL SETUP

The experimental setup is illustrated in Figure 12. A source node sends IPMC traffic to a BFIR. The BFIR encapsulates that traffic and sends it to a BFR. The BFR forwards the traffic to n BFERs which decapsulate the BIER traffic and send it as normal IPMC traffic to connected subscribers.

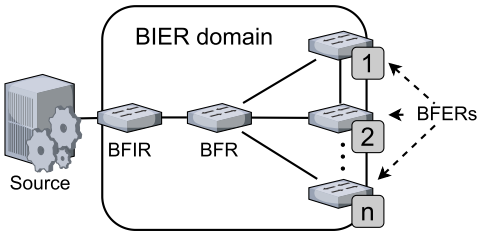

Figure 12. Theoretical setup for evaluation of BIER throughput.

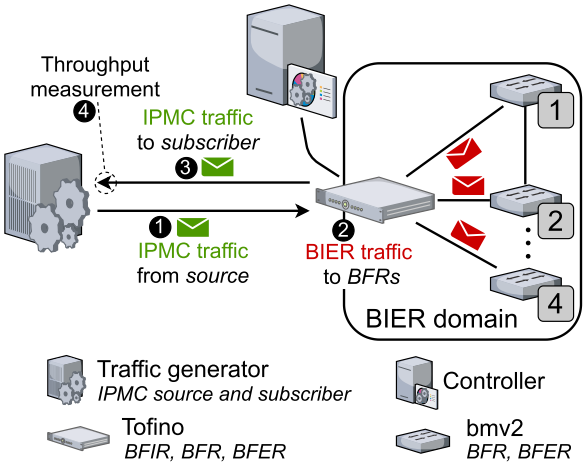

Figure 13. Hardware setup for evaluation of BIER throughput.

The goal of the experiment is to evaluate the forwarding performance of the BFR depending on the number of NHs. With n NHs, BIER packets have to be recirculated $n-1$ times, and internal packet loss occurs if recirculation capacity does not suffice. The objective of the experiment is to measure the BIER throughput depending on the number of recirculation ports for which only physical loopback ports are utilized in the experiment. However, the n subscribers may see different throughput. The first BFER does not see any packet loss while the last BFER sees most packet loss. Therefore, we measure the rate of IPMC traffic received on Layer 2 at the last subscriber.

2) HARDWARE SETUP AND CONFIGURATION

Due to hardware restrictions in our lab, we utilize one traffic generator, one P4-capable hardware switch, and one server running multiple P4 software switches to build the logical setup sketched above. The hardware setup is shown in Figure 13. The traffic generator is the source of IPMC traffic and sends traffic to the BFIR. The traffic generator is also the subscriber of BFER n and measures the throughput of received IPMC traffic on Layer 2. The hardware switch acts as BFIR, BFR, and BFER n while BFERs 1 to $n-1$ are deployed as P4 software switches on the server. In addition, we collapse the BFIR and the BFR in the hardware switch so that packet forwarding from the BFIR to the BFR is not needed. Therefore, the traffic generator is the last NH of the BIER packet when it is processed by the BFR.

Packet recirculation is required after (1) encapsulation to enable further BIER processing, (2) decapsulation to enable further IP forwarding, and (3) BIER packet replication to enable BIER forwarding to additional NHs. We set up the hardware switch so that all recirculation operations in connection with encapsulation and decapsulation are supported by two dedicated ports in loopback mode and spend another k ports in loopback mode to support packet recirculation after packet replication. This models the competition for recirculation ports on a mere BFR as in the theoretical model.

The P4 software switches are bmv2s that run alongside our controller on VMs on a server with an Intel Xeon Scalable Gold 6134 (8x 3.2 GHz) and 4 x 32 GB RAM. The P4 hardware switch is a Tofino [6] inside an Edgecore Wedge 100BF-32X [7] which is a 100 Gb/s P4-programmable switch with 32 ports. The traffic generator is an EXFO FTB-1 Pro [42] which generates up to 100 Gb/s. All devices are connected with QSFP28 cables which transmit up to 100 Gb/s.

3) INFLUENCE OF EXPERIMENTAL SETUP

The presented setup contains only a single Tofino-based switch which is partitioned and utilized as a single BFIR/BFR and a single BFER. All other BFERs in this setting are software switches that support only significantly lower bit rates (900 Mb/s [29]) than the Tofino-based switch (100 Gb/s). However, this has no impact on results because we measure the rate received by the single BFER implemented on the Tofino-based hardware. Furthermore, packet loss by the low-performance software switches does not reduce the generated traffic rate as this is configured as a constant rate on the generator.

4) BIER THROUGHPUT MEASUREMENTS DEPENDING ON RECIRCULATION PORTS

The traffic generator sends IPMC traffic at a rate of 100 Gb/s to the hardware switch, the hardware switch encapsulates the IPMC traffic, forwards BIER traffic iteratively n-1 times to bmv2s, recirculates the BIER packet to process the last activated header bit, decapsulates the traffic as BFER n, and returns it back to the traffic generator, which measures the received IPMC rate on Layer-2. We start measuring only after a 30 seconds initialization phase to avoid any influences from the startup phase. After 30 seconds, the traffic generator measures for 60 seconds the traffic arriving from the Tofino and reports the average Layer-2 throughput. We repeated experiments 10 times and computed confidence intervals with a confidence level of 95%. Their width was less than 0.5% of the measured average and, therefore, invisible.

Table 2. Model predictions $T(i)$ for BIER throughput and measured values $M(i)$ (Gb/s); the latter are the same values as presented in Figure 14.

Number of NHs:	1 NH		2 NHs		3 NHs		4 NHs	
Recirculation ports	$T(1)$	$M(1)$	$T(2)$	$M(2)$	$T(3)$	$M(3)$	$T(4)$	$M(4)$
1	100	99.32	100	99.32	38.2	43.3	15.74	19
2	100	99.32	100	99.32	100	99.32	53.14	50.5
3	100	99.32	100	99.32	100	99.32	100	99.32

Therefore, we omit them in future figures and tables for better readability.

In our experiments, we consider 1, 2, 3, and 4 NHs and utilize 1, 2, and 3 ports in loopback mode to support recirculation for BIER forwarding. The results are compiled in Figure 14.

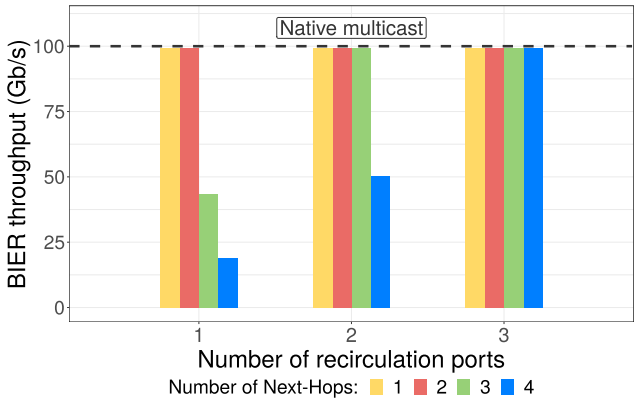

Figure 14. Measured throughput of BIER and traditional IPMC on the 100 Gb/s Tofino-based switch for different numbers of NHs and recirculation ports.

The left-most bar shows that with a single recirculation port, the last NH receives the full IPMC rate of 100 Gb/s if 1 NH is connected. The second bar from the left shows that the last NH still receives the full IPMC rate of 100 Gb/s if 2 NHs are connected. For 3 or 4 NHs, i.e., the third and fourth bar from the left, the IPMC traffic rate received by the last NH is reduced to 43 and 19 Gb/s, respectively.

With 2 recirculation ports, the last NH does not perceive a throughput degradation if at most 3 NHs, i.e., fifth to seventh bar from the left, are connected. For 4 NHs, i.e., eighth bar from the left, the IPMC traffic rate received by the last NH is reduced to 50 Gb/s.

And with 3 recirculation ports, even up to 4 NHs, i.e., ninth to twelfth bar from the left, can be supported without throughput degradation for the last NH.

Thus our experiments confirm that when multicast traffic arrives with 100 Gb/s at the Tofino, n-1 recirculation ports are needed to forward BIER traffic to n NHs without packet loss. This is different for a realistic multicast portion in the traffic mix, i.e., a minor fraction instead of 100%.

The hardware switch also supports traditional multicast in P4. With traditional multicast forwarding, all NHs receive 100 Gb/s regardless of the number of NHs. However, this comes with all the disadvantages of traditional IPMC we have discussed earlier.

C. THROUGHPUT MODEL FOR BIER FORWARDING WITH INSUFFICIENT RECIRCULATION CAPACITY

We model the throughput of BIER forwarding with insufficient recirculation capacity and validate the results with the experimentally measured values.

To forward a BIER packet to n NHs, it has to be recirculated $n-1$ times (see Section V-B). Any time a packet is sent to a recirculation, the packet is dropped with a certain probability if insufficient recirculation capacity is available. Due to the implemented round robin approach (see Section IV-C), the drop probability p is equal for all recirculation ports. The drop probability p in a system can be determined by comparing the available recirculation capacity and the sustainable recirculation load. The latter results from recirculations after BIER packet replication and takes packet loss into account. It is shown in the following formula.

$$C \cdot \sum_{m=1}^{n-1} (1-p)^m = k \cdot C \tag{1}$$

The available recirculation capacity is $k \cdot C$ where k is the number of recirculation ports and C is line capacity. The sustainable recirculation load is the sum of the successfully recirculated traffic rates after any number of recirculations. The traffic amount that has been successfully recirculated once is $C \cdot (1-p)$. The traffic amount that has been recirculated twice is $C \cdot (1-p)^2$, and so on. Therefore, the total amount is $C \cdot \sum_{m=1}^{n-1} (1-p)^m$.

We calculate the BIER throughput at any NH, i.e., after any number of recirculations. At the first NH, the throughput of the BIER traffic is C because the BIER packet is forwarded to the first NH before the packet is recirculated the first time. At the second NH, the BIER throughput is $C \cdot (1-p)$, at the third NH its $C \cdot (1-p)^2$, and so on. Therefore, the BIER throughput $T(i)$ at NH $1 \leq i \leq n$ is:

$$T(i) = C \cdot (1-p)^{i-1} \tag{2}$$

Table 2 shows the throughput predictions $T(i)$. We make predictions for the same scenarios as we evaluated in the performance evaluation in Section VI-B4 and compare them to the measured values $M(i)$.

The comparison shows that the model provides reasonable predictions for the BIER throughput.

VII. PROVISIONING RULE FOR RECIRCULATION PORTS

In this section we propose a provisioning rule for recirculation ports. It may be used for general P4-based applications requiring packet recirculation, not just for BIER forwarding. We first point out the importance for sufficient recirculation capacity. Then, we derive a general provisioning rule for recirculation ports and illustrate how their number depends on other factors. Finally, we apply that rule to provision the number of loopback ports for BFRs in the presence of traffic mixes.

A. IMPACT OF PACKET LOSS DUE TO MISSING RECIRCULATION CAPACITY

In Section IV-C we briefly discussed projects that leverage packet recirculation in P4. However, if recirculation capacity does not suffice and packets need to be recirculated several times, packet loss observed at the last stage may be quite high. We first illustrate this effect. If the packet loss probability due to missing recirculation capacity is p, then the overall packet loss probability after n recirculations is $p(n) = 1-(1-p)^n$. We illustrate this connection in Figure 15, which utilizes logarithmic scales to better view several orders of magnitude in packet loss. With only one recirculation, we obtain a diagonal for the overall packet loss. A fixed number of recirculations shifts the entire curve upwards, and with several recirculations like $n = 6$ or $n = 10$, the overall loss probability $p(6)$ or $p(10)$ is an order of magnitude larger than the packet loss probability p of a single recirculation step. Therefore, avoiding packet loss due to recirculations is important. Thus, sufficient recirculation capacity must be provisioned but overprovisioning is also costly since this means that entire ports at high speed cannot be utilized for operational traffic. Therefore, well-informed provisioning of recirculation ports is an important issue.

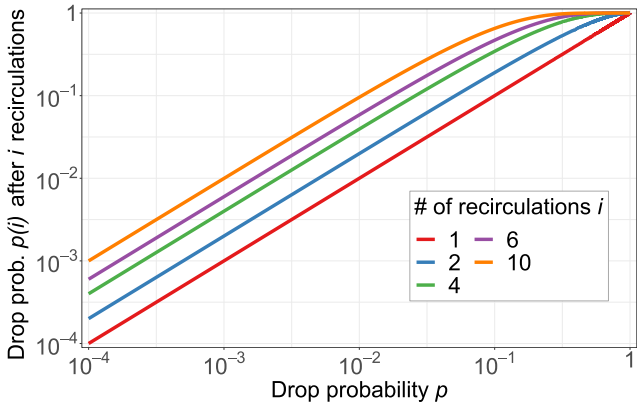

Figure 15. Loss probability after multiple recirculations.

B. DERIVATION OF A PROVISIONING RULE FOR RECIRCULATION PORTS

We first introduce the recirculation factor R and the utilization ratio U. Then, we use them to derive a provisioning rule for recirculation ports.

The recirculation factor R is the average number of recirculations per packet. Not all packets may be recirculated or the number how often a packet is recirculated depends on the particular packet.

The utilization ratio U describes the multiple by which a recirculation port can be higher utilized than a normal port. For example, if the average utilization of each normal port is 10%, then each recirculation port may be operated with a utilization of 40%, in particular if multiple of them are utilized. This corresponds to a utilization ratio of $U = 4$. We give some rationales for that idea. Normal ports at high speed are often underutilized in practice because bandwidths exist only in fixed granularities and usually link speeds are heavily overprovisioned to avoid upgrades in the near future. Furthermore, some links operate at lower utilization, others at higher utilization. Recirculation ports can be utilized to a higher degree. First, there is no need to keep the utilization of recirculation ports low for reasons like missing appropriate lower link speeds as it can be the case for normal ports. Second, recirculation ports are shared for all recirculation traffic of a switch so that resulting traffic fluctuations are lower and the utilization of the ports can be higher than the one of other ports.

If m incoming ports carry traffic with a recirculation factor R and a utilization ratio U can be used on the switch, then

$$m' = \left\lceil \frac{m \cdot R}{U} \right\rceil \tag{3}$$

describes the number of required recirculation ports.

C. ILLUSTRATION OF REQUIRED RECIRCULATION PORTS

For illustration purposes, we consider a P4 switch with 32 physical (external) ports and one virtual (internal) port in loopback mode for recirculations. If the capacity of that single virtual recirculation port does not suffice for recirculations, physical ports need to be turned into loopback mode as well and be used for recirculation. All recirculation ports are utilized in round-robin manner to ensure equal utilization among them.

Thus, the number of normal ports m plus the number of recirculation ports m' must be at most 33, i.e., 32 physical ports and 1 virtual port. Therefore, we find the smallest m' according to Equation 3, so that $m+m' \leq 33$ while maximizing m. The number of physical recirculation ports is $m'-1$ as the virtual port can also be used for recirculations. Figure 16 shows the number of physical recirculation ports depending on the recirculation factor R and the utilization ratio U. Since U depends on the specific use case and traffic mix, we present results for different values of U. Thereby, R and U are fractional numbers. While the number of recirculations for each packet is an integral number, the average number of recirculations per packet R is fractional. The number of physical recirculation ports increases with the recirculation factor R. Due to the fact that both m and m' are integers, the number of physical recirculation ports $(m-1)$ is not monotonously increasing because for some R and U the sum

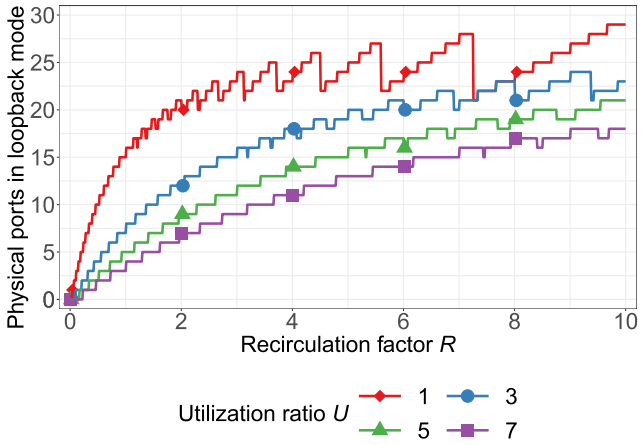

Figure 16. Number of physical ports in loopback mode.

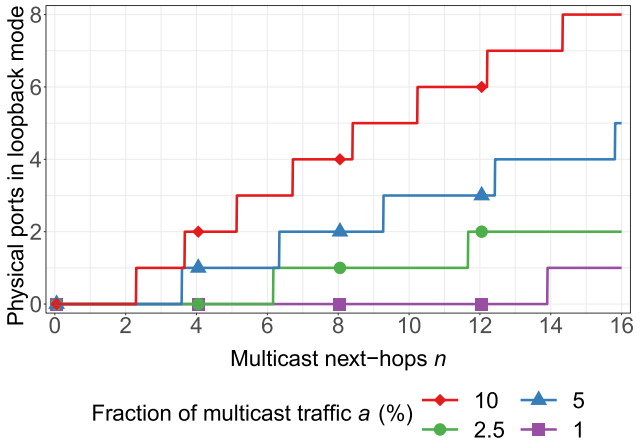

Figure 17. Physical ports in loopback mode for traffic mixes with realistic multicast portions.

$m + m'$ amounts to the maximum 33, and to lower values for other R and U.

The various curves show that the number of required physical recirculation ports decreases with increasing utilization ratio U. With a large recirculation factor $R \geq 3$ and a low utilization Ratio $U \leq 3$, half of the ports of the 32 port switch or even more need to be used for recirculation, which is expensive. However, with small $R < 1$ and large $U > 3$ the number of required physical recirculation ports is low because most of the traffic does not require packet recirculation, and due to the large utilization ratio U, the recirculation ports can cover significantly more traffic than normal ports. It is even possible that no physical recirculation port is needed if the recirculation capacity of the internal recirculation port can cover the recirculation load.

D. APPLICATION OF THE PROVISIONING METHOD TO TRAFFIC MIXES WITH BIER

In this section we make predictions for m', the number of recirculation ports, for traffic mixes with typical multicast portions. We assume different portions of multicast traffic $a \in \{0.01, 0.025, 0.05, 0.1\}$ and different average numbers of BIER NHs $n \in \{0, 2, 4, \ldots, 16\}$, i.e., each BIER packet is recirculated $n - 1$ times on average. Since unicast traffic is normally processed without recirculation, it does not need any recirculation capacity, i.e., its amount has no influence on the number of required recirculation ports and is, therefore, not considered in this analysis. Then, we calculate $R = a \cdot (n - 1)$, and assume $U = 4$. Again, we calculate the smallest m', i.e., like in Equation 3, so that $m + m' \leq 33$ while maximizing m. Figure 17 shows the number of physical recirculation ports depending on the average number of multicast NHs n and the fraction of multicast traffic a. If the fraction of multicast traffic is low like 1%, the capacity of the internal port suffices to serve up to 13 NHs on average. Moderate fractions of 2.5% multicast traffic require no physical recirculation port for up to 5 NHs, 1 physical recirculation port for up to 11 NHs, and 2 physical recirculation ports for 12 and more NHs. With 5% multicast traffic, the number of required physical recirculation ports increases almost linearly from zero to 5 with an increasing number of NHs. Large fractions of multicast traffic, like 10%, require up to 8 recirculation ports if the number of NHs is also large like 16. Under such conditions, 25% of the physical ports cannot be used for normal traffic forwarding as they are turned into loopback mode. However, the assumptions seem rather unlikely as multicast traffic typically makes up only a small proportion of the traffic.

VIII. CONCLUSION

The scalability of traditional IPMC is limited because core devices need to maintain IPMC group-dependent forwarding state and process lots of control traffic whenever topology or subscriptions change. Therefore, BIER has been introduced by the IETF as an efficient transport mechanism for IPMC traffic. State in BIER core devices does not depend on IPMC groups, and control traffic is only sent to border nodes, which increases scalability in comparison to traditional IPMC significantly. In addition, there are fast-reroute (FRR) mechanisms for BIER to minimize the effect of network failures. However, BIER cannot be configured on legacy devices as it implements a new protocol with a complex forwarding behavior.

In this paper we demonstrated a P4-based implementation of BIER with tunnel-based BIER-FRR, IP unicast with FRR, IP multicast, and Ethernet forwarding. The target platform is the P4-programmable switching ASIC Tofino which is used in the Edgecore Wedge 100BF-32X, a 32 100 Gb/s port high-performance P4 switch.

In an experimental study, we showed that BIER-FRR significantly reduces the restoration time after a failure, and in combination with IP-FRR, the restoration time is reduced to less than 1 ms. We confirmed that the prototype is able to forward traffic at a speed up to 100 Gb/s. However,

under some conditions, less throughput is achieved when switch-internal recirculation ports are overloaded. As a remedy, we added more recirculation capacity by turning physical ports into loopback mode. We modelled BIER forwarding, predicted limited throughput due to missing recirculation capacity, and validated the results by measured values. Furthermore, we proposed a simple method for provisioning of physical recirculation ports. The approach was motivated by BIER, but holds for general P4 programs requiring recirculations. In a case study, we applied it to BIER with different mixes of unicast and multicast traffic and showed that only a few physical recirculation ports suffice under realistic conditions.

REFERENCES

[1] I. Wijnands. (Nov. 2017). *RFC 8279: Multicast Using Bit Index Explicit Replication (BIER)*. [Online]. Available: https://datatracker.ietf.org/doc/rfc8279/

[2] D. Merling, S. Lindner, and M. Menth, "P4-based implementation of BIER and BIER-FRR for scalable and resilient multicast," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102764.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, and J. Rexford, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[4] W. Braun, J. Hartmann, and M. Menth, "Scalable and reliable software-defined multicast with BIER and P4," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 905–906.

[5] P4lang. (2021). *behavioral-Model*. Accessed: Jan. 28, 2021. [Online]. Available: https://github.com/p4lang/behavioral-model

[6] Edge-Core Networks. (2017). *The World's Fastest & Most Programmable Networks*. [Online]. Available: https://barefootnetworks.com/resources/worlds-fastest-most-programmable-networks/

[7] Edge-Core Networks. (2019). *Wedge100BF-32X/65X Switch*. [Online]. Available: https://www.edge-core.com/_upload/images/Wedge100BF-32X_65X_DS_R05_2019%1210.pdf

[8] M. Shahbaz, L. Suresh, J. Rexford, N. Feamster, O. Rottenstreich, and M. Hira, "Elmo: Source routed multicast for public clouds," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 458–471.

[9] S. Islam, N. Muslim, and J. W. Atwood, "A survey on multicasting in software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 355–387, 1st Quart., 2018.

[10] Z. AlSaeed, I. Ahmad, and I. Hussain, "Multicasting in software defined networks: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 104, pp. 61–77, Feb. 2018.

[11] J. Ráckert, J. Blendin, and D. Hausheer, "Software-defined multicast for over-the-top and overlay-based live streaming in ISP networks," *J. Netw. Syst. Manage.*, vol. 23, no. 2, pp. 280–308, Apr. 2015.

[12] J. Ruckert, J. Blendin, R. Hark, and D. Hausheer, "Flexible, efficient, and scalable software-defined over-the-top multicast for ISP environments with DynSdm," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 4, pp. 754–767, Dec. 2016.

[13] T. Humernbrum, B. Hagedorn, and S. Gorlatch, "Towards efficient multicast communication in software-defined networks," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jun. 2016, pp. 106–113.

[14] C. A. S. Oliveira, "Steiner trees and multicast," *Math. Aspects Netw. Routing Optim.*, vol. 53, pp. 29–45, Dec. 2011.

[15] L.-H. Huang, H.-J. Hung, C.-C. Lin, and D.-N. Yang, "Scalable and bandwidth-efficient multicast for software-defined networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 1890–1896.

[16] Z. Hu, D. Guo, J. Xie, and B. Ren, "Multicast routing with uncertain sources in software-defined network," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Service (IWQoS)*, Jun. 2016, pp. 1–6.

[17] S. Zhou, H. Wang, S. Yi, and F. Zhu, "Cost-efficient and scalable multicast tree in software defined networking," in *Proc. Conf. Algorithms Archit. Parallel Process.*, 2015, pp. 562–605.

[18] J.-R. Jiang and S.-Y. Chen, "Constructing multiple Steiner trees for software-defined networking multicast," in *Proc. 11th Int. Conf. Future Internet Technol.*, Jun. 2016, pp. 1–6.

[19] B. Ren, D. Guo, J. Xie, W. Li, B. Yuan, and Y. Liu, "The packing problem of uncertain multicasts," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 16, p. e3985, Aug. 2017.

[20] Y.-D. Lin, Y.-C. Lai, H.-Y. Teng, C.-C. Liao, and Y.-C. Kao, "Scalable multicasting with multiple shared trees in software defined networking," *J. Netw. Comput. Appl.*, vol. 78, pp. 125–133, Jan. 2017.

[21] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *Proc. 6th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2014, pp. 1–8.

[22] W. Cui and C. Qian, "Scalable and load-balanced data center multicast," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2014, pp. 1–6.

[23] W.-K. Jia and L.-C. Wang, "A unified unicast and multicast routing and forwarding algorithm for software-defined datacenter networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2646–2657, Dec. 2013.

[24] M. J. Reed, M. Al-Naday, N. Thomos, D. Trossen, and G. Petropoulos, "Stateless multicast switching in software defined networks," in *Proc. IEEE Int. Conf. Commun.*, May 2016, pp. 1–7.

[25] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 181–189.

[26] M. Popovic, R. Khalili, and J.-Y. Le Boudec, "Performance comparison of node-redundant multicast distribution trees in SDN networks," in *Proc. Int. Conf. Networked Syst. (NetSys)*, Mar. 2017, pp. 1–8.

[27] D. Kotani, K. Suzuki, and H. Shimonishi, "A multicast tree management method supporting fast failure recovery and dynamic group membership changes in OpenFlow networks," *J. Inf. Process.*, vol. 24, no. 2, pp. 395–406, 2016.

[28] T. Pfeiffenberger, J. L. Du, P. B. Arruda, and A. Anzaloni, "Reliable and flexible communications for power systems: Fault-tolerant multicast with SDN/OpenFlow," in *Proc. 7th Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jul. 2015, pp. 1–6.

[29] A. Bas. (Jan. 2018). *BMv2 Throughput*. [Online]. Available: https://github.com/p4lang/behavioral-model/issues/537#issuecomment-360537441

[30] A. Giorgetti, A. Sgambelluri, F. Paolucci, P. Castoldi, and F. Cugini, "First demonstration of SDN-based bit index explicit replication (BIER) multicasting," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2017, pp. 1–6.

[31] A. Giorgetti, A. Sgambelluri, F. Paolucci, N. Sambo, P. Castoldi, and F. Cugini, "Bit index explicit replication (BIER) multicasting in transport networks," in *Proc. Int. Conf. Opt. Netw. Design Modeling (ONDM)*, May 2017, pp. 1–5.

[32] Y. Desmouceaux and T. Clausen, "Reliable multicast with BIER," *J. Commun. Netw.*, vol. 20, pp. 182–197, May 2018.

[33] T. Eckert. (Nov. 2017). *Traffic Engineering for Bit Index Explicit Replication BIER-TE*. [Online]. Available: http://tools.ietf.org/html/draft-eckert-bier-te-arch

[34] W. Braun, M. Albert, T. Eckert, and M. Menth, "Performance comparison of resilience mechanisms for stateless multicast using BIER," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 230–238.

[35] F. Hauser, M. Haeberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank, and M. Menth, "A survey on data plane programming with P4: Fundamentals, advances, and applied research," 2021, *arXiv:2101.10632*. [Online]. Available: https://arxiv.org/abs/2101.10632

[36] J. Geng, J. Yan, and Y. Zhang, "P4QCN: Congestion control using P4-capable device in data center networks," *Electronics*, vol. 8, p. 280, Mar. 2019.

[37] C. Wernecke, H. Parzyjegla, G. Muhl, P. Danielis, and D. Timmermann, "Realizing content-based publish/subscribe with P4," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2018, pp. 1–7.

[38] M. Uddin, S. Mukherjee, H. Chang, and T. V. Lakshman, "SDN-based multi-protocol edge switching for IoT service automation," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2775–2786, Dec. 2018.

[39] *Reprinted from Journal of Network and Computer Applications, vol. 169, Daniel Merling, Steffen Lindner, Michael Menth, P4-Based Implementation of BIER and BIER-FRR for Scalable and Resilient Multicast*, Elsevier, Amsterdam, The Netherlands, 2020.

[40] D. Katz. (Jul. 2004). *Bidirectional Forwarding Detection (BFD)*. [Online]. Available: https://datatracker.ietf.org/doc/rfc5880/
[41] Q. Xiong. (Oct. 2017). *BIER BFD*. [Online]. Available: https://datatracker.ietf.org/doc/draft-hu-bier-bfd/
[42] EXFO. (2019). *FTB-1v2/FTB-1 Pro Platform*. [Online]. Available: https://www.exfo.com/umbraco/surface/file/download/?ni=10900&cn=en-US&pi=5404

DANIEL MERLING received the master's degree from the Chair of Communication Networks of Prof. Dr. habil. Michael Menth, Eberhard Karls University, Tübingen, Germany, in 2017, where he is currently pursuing the Ph.D. degree. His research interests include software-defined networking, scalability, P4, routing and resilience issues, multicast, and congestion management.

STEFFEN LINDNER received the bachelor's and master's degrees from the Chair of Communication Networks of Prof. Dr. habil. Michael Menth. He is currently pursuing the Ph.D. degree with the Communication Networks Research Group, Eberhard Karls University, Tübingen, Germany. His research interests include software-defined networking, P4, and congestion management.

MICHAEL MENTH (Senior Member, IEEE) received the Diploma degree from The University of Texas at Austin, in 1998, the Ph.D. degree from the University of Ulm, Germany, in 2004, and the Habilitation degree from the University of Wuerzburg, Germany, in 2010. He is currently a Professor with the Department of Computer Science, University of Tuebingen, Germany, since 2010, and the Chair Holder of communication networks. His research interests include performance analysis and optimization of communication networks, resilience and routing issues, resource and congestion management, industrial networking and the Internet of Things, software-defined networking, and the Internet protocols.