P4-BNG: Central Office Network Functions on Programmable Packet Pipelines

Ralf Kundel[§*], Leonhard Nobach[†*], Jeremias Blendin[‡*],
Hans-Joerg Kolbe[†], Georg Schyguda[†], Vladimir Gurevich[‡], Boris Koldehofe[§], Ralf Steinmetz[§]

[§] Multimedia Communications Lab, Technische Universität Darmstadt, Germany
{*ralf.kundel, boris.koldehofe, ralf.steinmetz*}*@kom.tu-darmstadt.de*
[†]Deutsche Telekom Technik GmbH, Fixed Mobile Engineering Deutschland, Darmstadt, Germany
{*leonhard.nobach, hans-joerg.kolbe, g.schyguda*}*@telekom.de*
[‡]Barefoot Networks, Santa Clara, CA, USA
{*jblendin, vgurevich*}*@barefootnetworks.com*

***Abstract*—Large-scale telecommunications providers have to continuously challenge and evolve their network infrastructure to efficiently serve growing markets demands. They must increase performance, lower time-to-market, provide new services, and lower the cost of the infrastructure and its operation.**

Network Functions Virtualization (NFV) on commodity hardware offers an attractive, low-cost platform to establish innovations much faster than with purpose-built hardware products. Unfortunately, implementing NFV on commodity processors does not match the performance requirements of the high-throughput data plane components in large carrier access networks. In this article, we propose a way to offer residential network access with programmable packet processing architectures. Based on the highly flexible P4 programming language, we present a design and open source implementation of a BNG data plane that meets the challenging demands of Broadband Network Gateways in carrier-grade environments. The proposed evaluation results show the desired performance characteristics and our proposed design together with upcoming P4 hardware can offer a giant leap towards highest performance NFV network access.

***Index Terms*—NFV, P4, Access Networks, Network Functions, Hardware Acceleration, Computer Networks**

I. Introduction

Internet Service Providers (ISPs) are challenged by competition, regulators, and content providers to deliver high-performance services to residential subscribers at ever lower costs. Essential cost drivers are the high-performance network functions required at the access edge. From a functional perspective, besides offering Internet connectivity, the access edge has to provide authorization and implement the quality properties of Internet access contracts of residential subscribers. Traditionally, residential network access is provided by physical network functions called *Broadband Network Gateways (BNGs)*, which are purpose-built devices that provide high performance, but tend to be costly. One approach to reduce the costs for network operators is *Network Functions Virtualization (NFV)*. NFV aims to increase cost-efficiency and flexibility by implementing network functions on cost-efficient commodity servers, commonly based on the `x86` or `ARM` processing architecture. One way to realize NFV in ISP networks is *Central Office Re-architected as a Datacenter (CORD)* [1] which aims to transfer ideas from the data center world to ISP networks. However, for high-performance packet forwarding softwarized approaches have unfavorable performance characteristics compared to fixed hardware solutions. Our previous work investigated how commodity *bare-metal switches* could be adapted to provide the function of residential network access [2]. Commodity switch silicon in 2016 was sufficient to implement the essential properties of a small-scale BNG with a very high performance, but not with all features needed by a large-scale network operator, like *Point-to-Point Protocol over Ethernet (PPPoE)* encapsulation/decapsulation or the required scalability.

With newer hardware designs and the highly-flexible P4 pipeline description language [3], packet processing can be adapted to a variety of use cases. P4 shows the potential to fulfill all functional requirements that are desired for a BNG data plane. Furthermore, it is expected that numerous vendors of packet processing hardware will support the P4 programming language to configure their upcoming switching chips (ASICs) or network processors. ASICs or network processor generally provide much better performance than software-based NFV approaches for data plane tasks.

The goal of this paper is to present a P4-based design and implementation of a BNG data plane, which runs in a CORD environment and fulfills all requirements of a large-scale telecommunications provider. In detail, the contributions of our work are as follows:

- We analyze the functional requirements of a large-scale telecommunications provider's BNG in Section III.
- We present a design and open source implementation a BNG data plane network function in a CORD-based design for P4-targets, fulfilling these requirements.
- We provide an abstraction layer between a generic BNG and the runtime environment.

After providing an overview over state of the art and related approaches, we present the results of the requirements analysis in Section III. Finally, we discuss the implementation and

[*] The authors contributed equally to this paper

978-3-903176-24-9

present some evaluation results.

II. Background and Related Work

To benefit from fast, programmable packet processing hardware, our implementation of the BNG data plane is using the P4 pipeline description language and targets upcoming programmable hardware devices. Our implementation is design as a part of the CORD architecture which supports the execution of central-office functionality, using Virtualized Network Functions (VNFs) on commodity hardware. In the following, we provide an overview of the CORD project, the P4 programming language and related work.

A. Broadband Network Gateways

Central offices, the points connecting subscribers to the telecommunications infrastructure, have evolved in the last 100 years from analog, human-driven switching centers to digital packet gateways. Today, residential network access functions in the central office are integrated in the BNG, mostly implemented on purpose-built hardware. The BNG function has been specified by several technical reports of the Broadband Forum (TR-101 [4], TR-145 [5], TR-178 [6]). According to IETF internet drafts, a BNG is defined as a server which routes traffic to and from broadband access devices, e.g., DSLAMs, on an ISP's network [7]. Further terms are *Network Access Server (NAS)* [8], *Broadband Remote Access Server (BRAS)*. In this work we use the common term *BNG*.

B. SDN for telecommunication networks

A major contribution of software-defined networking (SDN) architectures is the separation of the control plane and the data plane [9]. The usability of SDN for telecommunications providers has been investigated within the Project SPARC [10]. Focus of the latter research project was the disaggregation of the data plane, and multiple, hierarchical arranged, control planes. However, the project found the existing OpenFlow [11] protocol insufficient for the task and, therefore, proposed numerous improvements to it.

Recently, the *Open Networking Foundation (ONF)* conceived the "Central Office Re-Architected as a Data Center" (CORD) project [1] to overcome the dependency on vendors and proprietary hardware in carrier networks. Based on a combination of capabilities provided by SDN and *Network Functions Virtualization (NFV)* [12], the CORD project opens up new possibilities to bring "'economies of scale" and agility of data centers into central offices. With CORD, network functionality of central offices is implemented as VNFs, based on open-source software components, such as Docker [13], OpenStack [14] and ONOS [15]. The economies of scale of commodity hard- and software promises reduced costs and vendor lock-ins.

CORD focusses on *Gigabit Passive Optical Networks (GPON)* as the access technology. Nevertheless, other technologies like XGS-PON, G.fast [16] or VDSL fit into the system model as well. The distinction of the access lines is ensured through VLAN tags, which are added or removed by

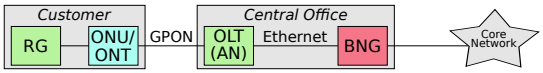

Fig. 1: Access topology assumption of the CORD project [1].

the Access Node (AN),in the concrete case of GPON this is an Optical Line Termination (OLT), shown in Figure 1. For each subscriber, a dedicated VNF instance, a virtual Subscriber Gateway (vSG), provides traffic processing. CORD claims that up to 1000 subscribers per server can be handled and a round-trip latency through the POD below $1ms$ is possible. Although this very promising approach achieves flexibility and vendor-independence, the performance of the vSG docker containers suffers from the software implementation of the data path, compared to using a hardware implementation [17].

C. OpenFlow

Implementing complex network functions with commodity OpenFlow hardware is not possible for numerous reasons. First, OpenFlow [11] does not support the use PPPoE or other special protocols [2]. Second, not all functionality that is available in the OpenFlow protocol, is available in actual OpenFlow switches.Thus, even a newer version of the protocol, e.g. supporting PPPoE, would be constraint by hardware limitations.

D. P4 and Reconfigurable Hardware

The majority of currently available devices, including OpenFlow-enabled switches, have fixed functionality that can only be modified within narrow limits. Furthermore, the design and production process for new ASICs is very time- and cost-intensive. The flexibility of software for VNFs is branded by limited bandwidth (PCIe, NIC, CPU) and its higher latency. The P4 language [3], introduced in 2014, is designed to describe the pipeline behavior of reconfigurable network devices, and combines the benefits of flexible software and high-performance hardware. This language has four main goals: (1) reconfiguration of deployed hardware, (2) independence of any network protocols, (3) independence of the specific target hardware and (4) the ability to generate control plane APIs for a given data plane program. P4 can be used to describe any network protocol headers and processing algorithms which can be expressed by a Direct Acyclic Graph (DAG). Thereby, new protocols can be added to the target pipeline without buying new hardware – only by updating the configuration.
Different vendors started introducing programmable hardware for packet processing in the last years, for example *Barefoot Networks* and *Netronome*. Furthermore, FPGAs can be used to implement pipelines described in P4 or similar languages, using high-level synthesis tools [18]. In this work, we show how this language can be used to describe a high performance BNG data plane.

E. Programmable Hardware for building VNFs

When implementing a high-performance VNF, *offloading* to hardware accelerators should be considered, discussed

in [19] [20]. These approaches assume, as the traditional VNF approach [21], a "off the shelf" server hardware which is extended by hardware accelerators like FPGAs or GPUs.

III. Requirements Analysis for a Carrier-Grade BNG

Implementing a fully-fledged P4-BNG requires detailed, in-depth knowledge of all functional requirements for BNGs in a DSL/GPON carrier environment. This section gives an insight into these functional requirements, which we have obtained by an requirements survey, based on [22], [23], [24] and own requirements.

A. Network Access Lines and Nodes

Access lines are a fundamental part of the BNG model. They are typically the limiting factor for the bandwidth that can be offered to a customer. Therefore, a subscriber is effectively sold a bundle of an access line and an Internet access service, established and enforced by a PPPoE session.
An access node terminates the customers' access line and provides corresponding virtual wires that connect the access lines to the BNG. Virtual wires are implemented using per-access line VLAN tags unique to each access node.

B. Functional BNG Requirements

A BNG is a system composed of one or multiple devices, which jointly implement residential and business access services. Based on specific configuration parameters this system assigns network resources including policy restrictions, usually defined by contractual constraints. In general, the *Access service provisioning* process is determined according to four general phases: (1) User discovery and authentication, (2) parameters assignment, (3) access control and features enforcement and (4) connection monitoring [24], depicted in Figure 2.

Starting from the top and moving down, the table illustrates different functional services that the BNG system must provide, leading to several tasks of the BNG implementation:

1) PPPoE: For each access line a single PPPoE session is used to authorize the subscriber and enforce restrictions as described in the subscribers contract. The discovery and session setup phase are conducted by the control plane, the data planes task is to forward packets from and to the control plane. After the session setup, the data plane handles PPPoE encapsulation and decapsulation.

2) Number of Active Subscribers per BNG: Due to the economies of scale, many subscribers should be treated by a BNG. According to an Intel blog post [25], the current number of subscribers per central office is around 5,000 but up to 35,000 are expected in the future.

3) IP Address Assignment to Subscribers and Reverse Path Forwarding: IP addresses must be assigned to new subscribers from the address pool by the control plane and installed in the data plane. The data plane has to ensure that packets to subscribers are forwarded using the correct PPPoE tunnel information. Furthermore, packets sent from subscribers have to be filtered by the IP addresses assigned to the subscriber; known as *reverse path forwarding (RFC 3704)*.

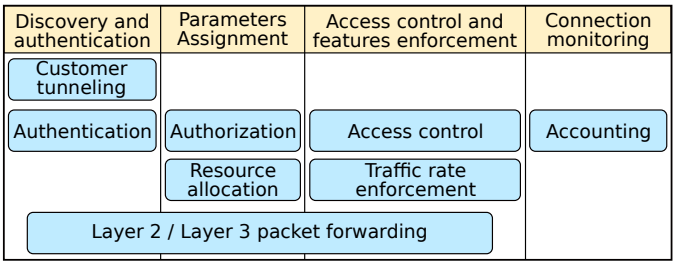

Fig. 2: Access service creation phases of a BNG system.

4) QoS Control: QoS is in use to separate traffic of different precedence and acuteness, e.g. normal Internet traffic, assured forwarding for services like IPTV and expedited forwarding for real-time services like Voice over IP (VoIP). As the corresponding QoS header fields of incoming packets are not trusted they should be overwritten by the BNG in order to prevent abuse of traffic classes by identifying QoS-classes based on L3-addresses for traffic from and to subscribers.

5) Rate Limiting and Prioritization: Rate limiting and prioritization for upstream traffic should be performed by the residential gateway and only upstream metering is required at the BNG, preventing abuse. The downstream traffic is limited and shaped at the BNG to ensure that the access network is not overloaded and a constant bandwidth can be guaranteed, based on customer contract and access network limitations.

6) Time to Live: The time to live (TTL) fields of forwarded packets are reduced in the forwarding process. If the TTL reaches zero the packet must be dropped and an ICMP notification has to be forwarded to the sender. Since this only happens very seldom, the processing can be done by the control plane.

7) MTU and Fragmentation: Adding the PPPoE header stack to traffic from the Internet to subscribers requires checking the packet size for violations of the maximum transmission unit (MTU). The residential gateway MTU is expected to be 1500 bytes for IP packets, the standard for Ethernet links. Packets transmitted on this link include additional PPPoE and VLAN headers as depicted in Figure 4. While the VLAN header is part of the Ethernet header, the 8 bytes for the PPPoE header decreases the MTU to 1492 bytes (RFC 4638). In case of larger packets the BNG has two options: (1) fragment and forward or (2) discard the packet and signal the event to the sender by an ICMP message. In case of IPv6 or IPv4 "don't fragment" packets fragmentation is not possible and therefore Path Maximum Transmission Unit discovery (PMTU) or TCP MSS-determination is usually conducted (RFC 1191).

8) Multicast: IP Multicast is useful for IPTV products of the ISP in order to reduce the total traffic by duplicating packets at the BNG. Establishing the multicast streams from upstream sources as well as processing IGMP messages from subscribers is done by the control plane.

C. Fundamental functional tasks of a virtual BNG network function

Based on these requirements the functional components are listed in Table I. The functionalities and their descriptions correspond directly with one of the access service creation phases from Figure 2. The signs can be mapped later with the design in Figure 4.

Function	Sign	Description
Customer tunneling		Encapsulate customer traffic with PPPoE
Authentication, authorization and accounting		IP-Address assignment, route establishment and traffic counting
Traffic rate enforcement		rate control, rate limiting and traffic shaping
Traffic access control		Provide processing of packets belonging to session authorized subscribers.
Traffic separation		Split control and data plane traffic, f.e. TTL=0 packets
Quality of service		Provide techniques that allow prioritization of traffic
Service aggregation		Allow aggregation of the circuits from one or more access link platforms; virtual wire start point
Security assurance		Provide protection of the system against misbehavior like address spoofing, e.g. RPF

TABLE I: Functional components of BNG systems.

IV. CORD-based BNG Architecture for Programmable Data Planes

In this section we focus on the high-speed packet-header processing part, the data plane; the control plane part is beyond the scope of this paper, however, we discuss the interfaces to that later. Our BNG design supports implementations for multiple P4 targets: (1) Barefoot Tofino, (2) Netronome SmartNIC, (3) P4-NetFPGA and the (4) P4 behavioral model (bmv2). In the following, we propose an integration of programmable data planes into the CORD framework, fulfilling the functional requirements of BNGs by disaggregation the BNG functionality:

A. Software Architecture Overview

One advantage over today's BNGs is the split of the functionality over three functional blocks, as depicted in Figure 3: a programmable data plane, the data plane controller, providing a control plane interface, and the BNG control plane itself. Furthermore, an access technology specific Access Node (AN), e.g. an OLT for GPON, is required. The functional requirements for the data plane are implemented in the *BNG Header Processing* component of the data plane, depicted in light green. The heterogeneity of different P4 data plane hardware implementations and their control plane interfaces are hidden by the hardware-specific data plane controller which provides a uniform interface towards the control plane. The BNG data plane controller is running on the CPU of the programmable data plane device. The BNG control plane is

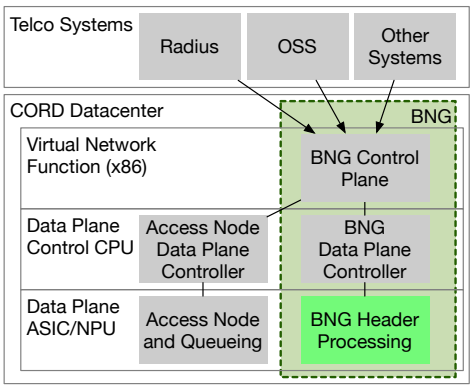

Fig. 3: CORD-Service Edge component overview.

operating as a x86-based VNF somewhere in the CORD data center. Besides the programmable data plane the control plane also controls the attached access nodes. External components, such as the Radius server or the Operations Support System (OSS), are attached on the northbound interface of the BNG control plane.

B. BNG Pipeline Design

Based on the CORD data center architecture and on current and future national (legal) requirements we propose a protocol stack and design architecture as depicted in Figure 4 (see Table I for the meaning of the symbols). The end-to-end header processing works as follows: The residential gateway (RG) sets up one PPPoE connection with the BNG. Different services are identified by multiple VLAN tags. The access line of a customer is expanded to the BNG by a VLAN-based virtual wire. In addition, the packets are encapsulated in two MPLS segment routing headers and a new outer Ethernet header, as intended by the CORD fabric which forwards the packets from the access node to the BNG. At session setup, the PPPoE control messages arriving at the BNG are forwarded to the control plane component. After authenticating the subscriber, a PPPoE session is created and all required flow rules, enabling packet forwarding from and to the Internet, are installed in the BNG data plane. For normal traffic, the BNG removes the access header stack as well as the CORD MPLS headers, after applying reverse-path filtering and ACLs, forwards the packet to the uplink interface with two new MPLS headers. Downstream traffic is processed accordingly in the opposite order. In today's design uplink rate limiting and traffic shaping is implemented in the residential gateway. Therefore, policing at the access node is sufficient for avoiding bandwidth violation. In contrast to this, downstream traffic requires policing, traffic shaping and prioritization at the central office in or close to the BNG. The queueing requirements of ISPs in access networks, e.g. huge buffers and hierarchical queues in large quantities, are not met by today's commodity packet switching ASICs. Therefore, his functionality is implemented on the access node. Access nodes typically include a non-programmable switching chip which includes sufficient queueing capabilities for their own

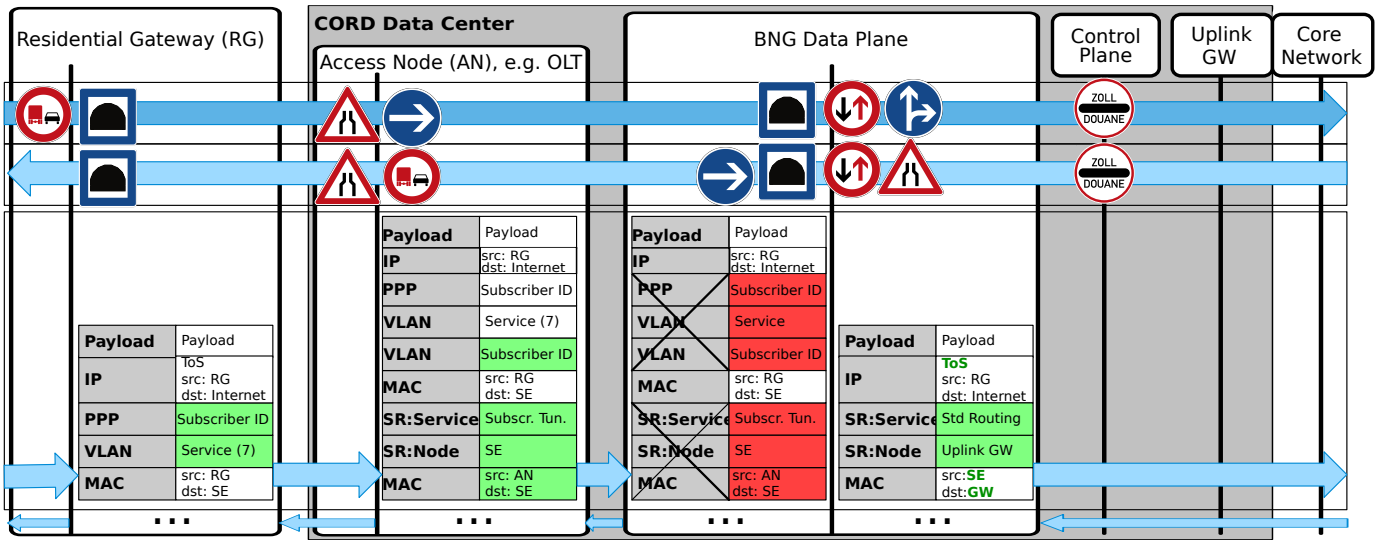

Fig. 4: Design for implementing BNG functionality in a CORD site.

subscribers. As multiple of them are used in a CORD data center, we assume the queueing functionality to be placed there, as the resources are available anyway. We do not focus on this further, but other solutions are possible as well.

V. P4-based Pipeline Implementation

In this section, we describe our implemented P4 pipeline of a BNG data plane based on the requirements obtained in Section II and IV wich is OpenSource available as part of the openCORD project on Github[1]. Figure 5 provides an overview over the BNG pipeline modeled in P4. Note that the pipeline of a BNG differs significantly between upstream and downstream traffic processing. Thus, the depicted flow diagram can be separated into an upstream (light blue) and downstream (light-red) part.

The control traffic is forwarded to the control plane and identified in the table `t_cptap_outer_ethernet`. Then, a subsequent table `t_usds` decides based on the combination of physical port and *MPLS0* label, whether a packet comes from a core or an access port and is destined to the Upstream or the Downstream pipeline.

A. Upstream Pipeline

The Upstream pipeline (Figure 5 light-blue) first applies the table `t_line_map`. This table maps each combination of a physical port, *MPLS0*/*MPLS1* label and a subscriber VLAN ID to a unique *line ID*.

If the line is legitimate, the table `t_pppoe_cpdp` determines whether a packet is a PPPoE control plane packet or a data plane packet for further processing; all unknown and illegitimate packets are dropped for security reasons. Typical packets destined to the controller, are PPPoE discovery, PPPoE LCP protocol, or keep-alive packets. The decision, if a packet is control traffic, can be done based on the inner L2 destination address, the service VLAN Ethertype, and the *PPPoE Protocol* type. This table is preconfigured at startup, only requires a few entries (< 16), and does not grow with the number of subscribers or networks.

After authenticating the subscriber via PPP, the residential gateway's MAC address, the negotiated PPPoE session ID, the service VLAN ID, and the line ID are used as input for the table `t_antispoof_mac`, which write the *subscriber ID* to the packet metadata. As a subscriber ID is unique per line, only the combination of line ID and subscriber ID can identify a subscriber.

After ensuring that only authorized subscribers can use the data plane, IP source address spoofing is prevented. To this end, we have to distinguish between IPv4 and IPv6 traffic. For every authenticated subscriber, networks must be added to the `t_antispoof_ipv4` and `t_antispoof_ipv6` tables which ensures that the source address matches the allowed address range. Packets without a valid source address are dropped.

The last ingress table, `t_us_routev4/6`, decides to which core interface and MPLS destination a packet should be sent towards its destination address. Then, the original header stack becomes replaced by the MPLS-SR/IP stack which is used in the core network. Finally, an egress table applies the correct outer destination and source MAC addresses, before the packet is sent out.

B. Downstream Pipeline

Compared to the upstream, the downstream direction is much simpler and requires fewer tables. If a packet enters the BNG, we determine the authenticated subscriber (line ID and subscriber ID) based on the destination IP address in table `t_ds_routev4/6` which is filled with all networks of currently authenticated subscribers. If a match occurs, the corresponding action attributes contain all information, required to construct the MPLS/VLAN/PPPoE header stack and the

[1]https://github.com/opencord/p4se

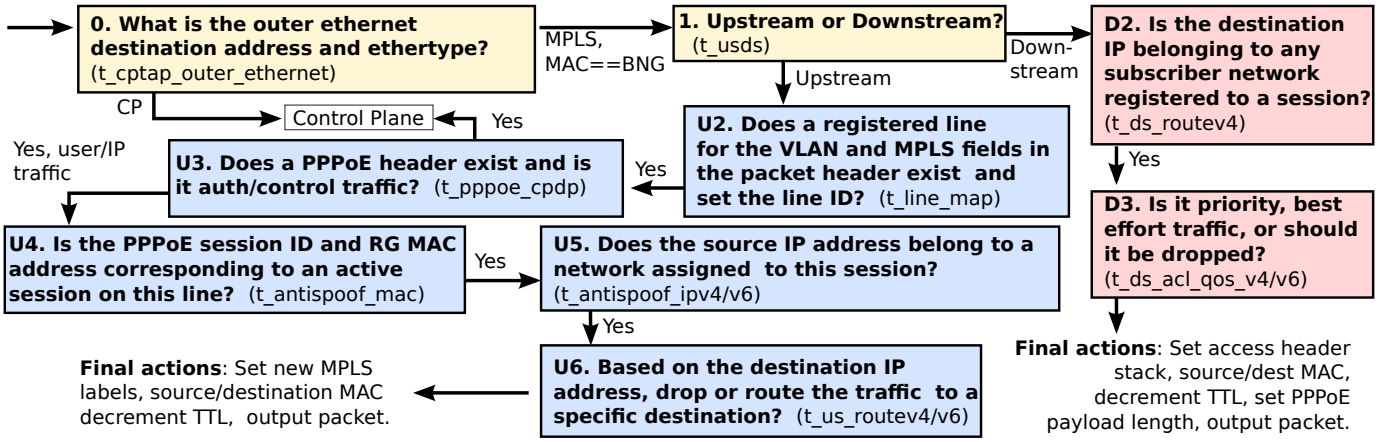

Fig. 5: Overview over the end-to-end packet pipeline.

output port. Furthermore, a per-subscriber meter is applied to ensure not exceeding downstream bandwidths. Finally, in the egress pipeline the PPPoE payload length is calculated based on the IP header length field.

C. State Configuration Model

In the following, we suggest a higher-level control/data plane interface for a BNG providing access to the functionality of our implementation described before. We also sketch the P4 tables required to make the given modifications. The implementation of the control interface is provided in *p4_runtime_mgr.py*. Besides this generic implementation, a specific implementation or each target, which is called by this generic interface, is needed, e.g. for the P4-NetFPGA, SmartNICs or the BMv2.

Just as one example, `enableLabelPortAccess(port, mpls0_label, ourOuterMAC, peerOuterMAC)` sets a combination of a physical port and outer MPLS label to be considered for receiving upstream traffic from and transmitting downstream traffic to the access network. Furthermore, the function sets the expected outer MAC address of the BNG and the communication partner's expected outer MAC address to the corresponding value and it adds entries to the tables `t_usds` and table `t_ds_srcmac`. The insertion of the table entries is performed by the corresponding BNG data plane controller which has a unified interface to the control plane.

Table `addUpstreamRouteV4/V6(..)` is another example which sets a route for authenticated, reverse-path-filtered subscriber traffic to next hops in the core network. For a network prefix and a service ID, the next hop's MAC address and the MPLS segment routing labels can be defined. It is possible to specify NULL for the upstream route to deny traffic to a certain network, thus to implement an IP-based ACL. It adds an entry to table `t_us_routev4/v6` for the downstream direction.

All other API methods, which are implemented and documented in the Open Source code as well, work very similar to those.

VI. Verification and Evaluation

The functionality of the BNG can be divided into two parts: (1) service creation and (2) packet forwarding from subscribers to the Internet described in the next two subsections. The first is executed only once per session and therefore its functional correctness is more important than its performance. The latter should be characterized by high throughout, low delay and minimum packet loss. The sources of our P4-based BNG implementation and the verification framework are available online[2].

A. Functional Verification

A BNG data plane implementation, targeted for productive usage, must be verified for correct operation in various ways. In particular, the following desired qualitative behavior has been considered for testing:

- The control plane can send arbitrary packets on any port.
- On registered lines, subscriber authentication packets (PPPoE) are forwarded to the control plane.
- Subscriber traffic is forwarded by the data plane to a core-facing port if a PPPoE session is installed.
- *Reverse path forwarding* mechanisms are in effect. Packets with source addresses, subnets (Antispoof) or destination subnets (ACL) are dropped.

To validate this behavior, we implemented a Python based framework based on the `Scapy` library for packet creation and sending. The verification framework runs on a separate computing node (Figure 6), sending packets and verifying the behavior. Although the maximum achievable bandwidth is low compared to high performance load generators, it is sufficient for functional testing. In addition, the Packet Testing Framework (PTF) of the P4 consortium turned out to be very helpful and increases productivity. The tests are written in a target-independent way by introducing two abstraction layers: First, we propose a **control plane abstraction layer** for a simple BNG network function, based on the interface suggested in Section V-C, which translates control plane commands into

[2]https://github.com/opencord/p4se

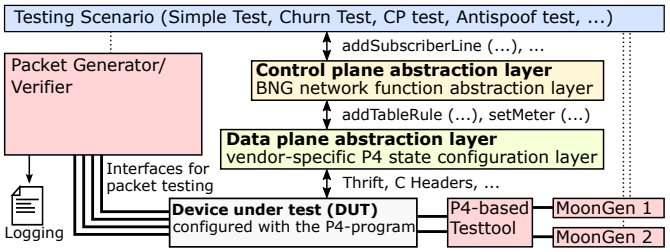

Fig. 6: Data Plane Verification Framework with the abstraction layer used for the evaluation. Either the "Packet Generator/Verifier" or "MoonGen + P4-Testtool" can be used.

the underlying P4 state configuration commands (e.g. add an entry to a table). Example for control plane commands are: adding subscriber circuits, activating/deactivating subscriber sessions after authentication/de-authentication, and adding network addresses, and subnets to the subscriber sessions at runtime. Secondly, the framework consists of a P4 **data plane abstraction layer**, which we had to introduce because the state configuration interfaces for the P4 pipeline might be vendor-specific. Unified control plane interfaces, e.g. P4-Runtime, will simplify this in future. All performed functional tests have shown the expected behavior, based on the capabilities of the target (see VI-D) and thus BNG network functionality can be implemented with P4 targets. Performance characteristics are discussed in the following.

B. Performance Characteristics

Our generic P4 Implementation can be executed by the software model *bmv2*, P4-NetFPGAs and Netronome P4-SmartNICs. As the *bmv2* is a software model for functional prototyping a performance comparison to real hardware is not useful. Thus, we compare the Agilio SmartNICs and the P4-NetFPGA design with x86-based server which performs simple packet forwarding (without PPPoE termination or any other BNG functionality) in the Linux-kernel by docker-containers. We assume that a docker based CORD [1] implementation (without kernel bypassing) of BNG functionality will perform, in the best case, similar to the Linux kernel in terms of throughput, packet loss and latency. In addition, some estimations for the P4 programmable Barefoot Tofino implementation, which as open source available as well, are given.

The following metrics will be considered in the following:

- Reliability/packet loss of authenticated sessions
- The number of simultaneous supported sessions
- The total data plane performance (*throughput*, *latency*)
- The traffic is correctly limited based on meter settings

To investigate the performance, we used a testbed setup as depicted in Figure 6. The device under test (DUT) is connected with two ports (core and subscriber side) to a P4-programmable switch ("P4-based Testtool"), stamping all packets before and after the DUT with a timestamp inside the packet in a 16 byte TCP option field of the payload. Load generation is performed by two x86 servers with DPDK capable NICs and the Moongen load generator [26], which create packets from the access nodes (*MoonGen 1*) and the core side (*MoonGen 2*), as depicted in Figure 4. By capturing the packets, latencies can be extracted from the corresponding header field. Due to link speed limitations of the NetFPGA Sume, all tests are performed with $10Gbit/s$ link speed.

As additional 30 bytes (inner Ethernet, 2xVLAN, PPPoE) are added to the packet in downstream direction, the downstream traffic must be shaped before the DUT in the programmable switch to $9.4Gbit/s$ in order to prevent packet loss. Upstream traffic is not affected by that. The packet size before entering the DUT is 532 bytes (including timestamp header).

Figure 7 shows the average latency of the different target platforms depending on the throughput on a logarithmic scale for 512 subscribers. The latency of the P4-NetFPGA is very constant around $5\mu s$ whereas the P4-SmartNIC, which processes the packets in many cores of the NPU, has a latency of $10\mu s$ for low bandwidths and increases up to $22\mu s$ for $10Gbit/s$ which is still quite well. In case of Linux kernel based packet forwarding the latency for $10Mbit/s$ is $53\mu s$ on average and goes down to $33\mu s$ for $100Mbit/s$. After reaching a bandwidth of $1700Mbit/s$ the Linux kernel is not able to handle more packets which leads to a rising packet loss rate and an average latency of $2.2ms$ for $3000Mbit/s$ and above. Increasing the number of subscribers to 4000 has no measurable impact. For Barefoot Tofino we assume a latency below $1\mu s$ as long as no packets are queued due to egress port oversubscription. The standard deviation of the latency is given in Table II. The delay of a hardware pipeline, as the P4-NetFPGA or Barefoot Tofino, is very constant. In contrast to that, software approaches such as the Linux packet forwarding has a higher variability in processing times because of many influencing factors, but mainly the interrupt based system architecture. NPUs, such as the P4-SmartNICs, are in between. Table III lists the total latency (transceiver to transceiver) dependent on the number of subscribers for a constant downstream traffic rate of $9.4Gbit/s$ which is very

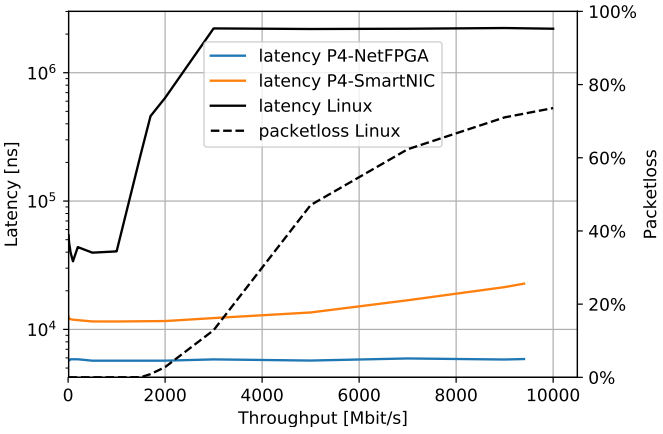

Fig. 7: Latency for Linux-kernel, P4-NetFPGA and P4-SmartNICs depending on throughput for downstream traffic (512 subscriber). Packet loss of all P4 targets is constant 0.

	Linux-Kernel	Netronome-SmartNIC	P4-NetFPGA
100 Mbit/s	$13.2\mu s$	$1.7\mu s$	$< 0.5\mu s$
1000 Mbit/s	$19.8\mu s$	$0.7\mu s$	$< 0.5\mu s$
90000 Mbit/s	$157.9\mu s$	$1.6\mu s$	$< 0.5\mu s$

TABLE II: Latency Standard Deviation of different targets.

#Subscribers	32	128	512	2048	4000
P4-NetFPGA	$5.08\mu s$	$5.08\mu s$	$5.08\mu s$	$5.08\mu s$	$5.08\mu s$
P4-SmartNIC	$22.12\mu s$	$22.13\mu s$	$22.15\mu s$	$22.03\mu s$	$22.06\mu s$

TABLE III: Latency depending on number of subscribers for $9.4Gbit/s$ downstream traffic. packet size: 532 byte.

constant and does not depend on the number of subscribers. The results for upstream traffic on SmartNICs and FPGA are slightly higher (hundreds of ns).We assume that this is caused by the more complex program (see Figure 5). As long as all flow tables are realized in SRAM and not in external DRAM we expect this behavior for all future programmable network devices as well. Rate limiting can be performed by Netronome SmartNICs and Barefoot Tofino, which have not shown any violations of the configured limits.

C. Resource Utilization

Although the different target platforms have different hardware architectures, we provide some values in order to point out the bottlenecks. The P4-NetFPGA project is based on a Xilinx Virtex-7 690T chip, introduced in 2010, which persists mainly on many reconfigurable lookup tables (LUT) and memory blocks (BRAM). The resource utilization of the P4-NetFPGA with 4096 subscribers is given in Table IV and shows the total consumption of the design and the P4 pipeline only. This difference is caused by four 10G Ethernet IP-Cores, a micro-processor and additional peripheral logic. The limiting resource is BRAM, which is only available in selected areas of the FPGA and thus a utilization of 100 % becomes very hard due to timing constraints. However, newer FPGA generations from Intel and Xilinx provide much more resources and by that even more subscribers could be realized. The resource utilization of our P4 service edge implementation for Barefoot Tofino, given in Table V, shows that the number of subscribers is not the only influencing factor of the resource consumption. Depending on further requirements and parameters, the previously named goal of 35,000 subscribers per BNG [25] can be achieved with additional optimization of the P4 program.

D. Language and Target Specific Limitations

Although P4 was designed as a target independent language we observed that modifications of the P4 code are needed. Netronome SmartNICs are able to execute P4_14 code, written for software reference switch BMv2, without modifications; all needed P4 functionality was supported. However, we observed

	total	P4-datapath	available
LUT	202155 (47%)	163013 (38%)	433200
BRAM	1074.5 (73%)	952 (65%)	1470

TABLE IV: P4-NetFPGA resource utilization.

	4096 Subscribers	8192 Subscribers
SRAM	12.81%	15.94%
TCAM	15.97%	15.97%
#Pipeline Stages	9	9

TABLE V: Resource utilization for Barefoot Tofino.

limitations, which can lead to dropped flow rules regardless to the configured table size under special circumstances if the number of installed flow rules becomes too high. Barefoot Tofino has also shown a very good support of the language and its features, however the P4 code requires minor modifications. The P4-NetFPGA project, based on the Xilinx SDNet toolchain, supporting only P4_16 , does currently not support P4 tables with longest prefix match or multiple match inputs and *parser_value_sets* are not supported. As this is no limitation of FPGAs itself, this might be supported with future compiler releases. One benefit of P4 on FPGAs are external functions which can be used to integrate logic not supported by the P4-compiler, e.g. metering and traffic shaping.
QoS-aware flow classification can easily be done in P4. Although describing scheduling behavior is currently target dependent and outside of the scope of P4, the language provides a mechanism (*intrinsic metadata*) allowing P4 programs to interface with any vendor-specific queueing and scheduling component. We observed a similar situation for multicast traffic, managed by vendor-specific logic which can be controlled by *intrinsic metadata* as well.

VII. Conclusion

The P4 pipeline description language is a powerful instrument that allows network designers to create highly functional and versatile data plane programs. In this paper, we have proposed a P4-based implementation of a BNG network function data plane, which complies with all essential requirements of a large operator of a telecommunications network. Together with our implementation, we have also proposed a vendor- and hardware-independent runtime configuration interface for a BNG data plane, allowing control planes to be decoupled from the latter. The latest advances in reconfigurable hardware enable the execution of the P4 BNG implementation with highest performance, demonstrated by our evaluation results. Following the open-source idea, we have shared the code with the networking community.

Next steps will be an in-depth integration and operational testing – towards a flexible, programmable BNG network function with highest performance in productive use. The BNG use case would clearly benefit from any future enhancements of P4 in the area of queueing and QoS behavior description.

Acknowledgements

This work has been supported by Deutsche Telekom through the Dynamic Networks 6 project, and in parts by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Center (CRC) 1053 – MAKI. Furthermore, we thank our colleagues and reviewers for their valuable input and feedback.

REFERENCES

[1] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 96–101, October 2016.

[2] L. Nobach, J. Blendin, H.-J. Kolbe, G. Schyguda, and D. Hausheer, "Bare-metal switches and their customization and usability in a carrier-grade environment," in *Conference on Local Computer Networks (LCN)*. IEEE, Oct 2017, pp. 649–657.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656890

[4] T. Anschutz, "Migration to Ethernet-based Broadband Aggregation," Broadband Forum, Tech. Rep. 101 Issue 2, July 2011.

[5] A. Cui and Y. Hertoghs, "Multi-service Broadband Network Functional Modules and Architecture," Broadband Forum, Tech. Rep. 145 Issue 1, November 2012, https://www.broadband-forum.org/technical/download/TR-145.pdf. [Online]. Available: https://www.broadband-forum.org/technical/download/TR-145.pdf

[6] C. Alter, Y. Hertoghs, H. Li, and J. Rius i Riu, "Multi-service Broadband Network Architecture and Nodal Requirements," Broadband Forum, Tech. Rep. 178 Issue 1, September 2014, https://www.broadband-forum.org/technical/download/TR-178.pdf. [Online]. Available: https://www.broadband-forum.org/technical/download/TR-178.pdf

[7] fangwei hu, R. Hua, and S. Hu, "Yang data model for configuration interface of control-plane and user- plane separation bng," Working Draft, IETF Secretariat, Internet-Draft draft-hu-rtgwg-cu-separation-yang-model-01, January 2018, http://www.ietf.org/internet-drafts/draft-hu-rtgwg-cu-separation-yang-model-01.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-hu-rtgwg-cu-separation-yang-model-01.txt

[8] S. Ooghe, N. Voigt, M. Platnic, T. Haag, and S. Wadhwa, "Framework and requirements for an access node control mechanism in broadband multi-service networks," Internet Requests for Comments, RFC Editor, RFC 5851, May 2010.

[9] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology," Tech. Rep., 2015.

[10] W. John, A. Devlic, Z. Ding, D. Jocha, A. Kern, M. Kind, A. Köpsel, V. Nordell, S. Sharma, P. Sköldström *et al.*, "Split architecture for large scale wide area networks," *arXiv preprint arXiv:1402.2228*, 2014.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[12] M. Chiosi, D. Clarke, P. Willis, A. Reid, and et al., "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for actions," *SDN and OpenFlow World Congress*, 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[13] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2600239.2600241

[14] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, 2012.

[15] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620744

[16] M. Timmers, M. Guenach, C. Nuzman, and J. Maes, "G. fast: evolving the copper access network," *IEEE Communications Magazine*, vol. 51, no. 8, pp. 74–79, 2013.

[17] P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, and G. Carle, "Throughput and latency of virtual switching with open vswitch: A quantitative analysis," *J. Netw. Syst. Manage.*, vol. 26, no. 2, pp. 314–338, Apr. 2018. [Online]. Available: https://doi.org/10.1007/s10922-017-9417-0

[18] P. Benáček, V. Puš, H. Kubátová, and T. Čejka, "P4-to-vhdl: Automatic generation of high-speed input and output network blocks," *Microprocessors and Microsystems*, vol. 56, pp. 22 – 33, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933117304787

[19] L. Nobach and D. Hausheer, "Open, elastic provisioning of hardware acceleration in nfv environments," in *Networked Systems (NetSys), 2015 International Conference and Workshops on*. IEEE, 2015, pp. 1–5.

[20] Z. Bronstein, E. Roch, J. Xia, and A. Molkho, "Uniform handling and abstraction of nfv hardware accelerators," *IEEE Network*, vol. 29, no. 3, pp. 22–29, 2015.

[21] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[22] I. Agilent Technologies, "Understanding dslam and bras access devices (white paper)," 2006.

[23] R. Bifulco, T. Dietz, F. Huici, M. Ahmed, J. Martins, S. Niccolini, and H.-J. Kolbe, "Rethinking access networks with high performance virtual software brases," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. IEEE, 2013, pp. 7–12.

[24] T. S. Consortium, "Split architecture for large scale wide area networks - deliverable d3.3," 2012, accessed: 2017-07-03. [Online]. Available: http://www.fp7-sparc.eu/project/deliverables/

[25] D. Rodriguez, "Next generation central offices transform network edge with datacenter economics, cloud flexibility," 2018.

[26] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "Moongen: A scriptable high-speed packet generator," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: ACM, 2015, pp. 275–287. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815692