Received 09 February 2025; revised 07 March 2025; accepted 29 March 2025; Date of publication 02 April 2025; date of current version 30 April, 2025.

Digital Object Identifier 10.1109/10.1109/OJCOMS.2025.3557082

MPLS Network Actions: Technological Overview and P4-Based Implementation on a High-Speed Switching ASIC

FABIAN IHLE AND MICHAEL MENTH *(Senior Member, IEEE)*

Chair of Communication Networks, University of Tuebingen, 72076 Tuebingen, Germany

CORRESPONDING AUTHOR: M. Menth (e-mail: menth@uni-tuebingen.de).

This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG) under Grant ME2727/3-1, and in part by the Open Access Publishing Fund of the University of Tübingen.

ABSTRACT In MPLS, packets are encapsulated with labels that add domain-specific forwarding information. Special purpose labels were introduced to trigger special behavior in MPLS nodes but their number is limited. Therefore, the IETF proposed the MPLS Network Actions (MNA) framework. It extends MPLS with new features, some of which have already been defined to support relevant use cases. This paper provides a comprehensive technological overview of MNA concepts and use cases. It compares MNA to IPv6 extension headers (EHs) that serve a similar purpose, and argues that MNA can be better deployed than EHs. It then presents P4-MNA, a first hardware implementation running at 400 Gb/s per port. Scalability and performance of P4-MNA are evaluated, showing negligible impact on processing delay caused by network actions. Moreover, the applicability of MNA is demonstrated by implementing the use cases of link-specific packet loss measurement using the alternate-marking-method (AMM) and bandwidth reservation using network slicing. We identify header stacking constraints resulting from hardware resources and from the number of network actions that must be supported according to the MNA encoding. They make an implementation for hardware that can only parse a few MPLS headers infeasible. We propose to make the number of supported network actions a node parameter and signal this in the network. Then, an upgrade to MNA is also feasible for hardware with fewer available resources. We explain that for MNA with in-stack data (ISD), some header bits must remain unchanged during forwarding, and give an outlook on post-stack data (PSD).

INDEX TERMS Alternate-Marking Method, Data Plane Programming, IETF, MPLS Network Actions, Multiprotocol Label Switching, Network Slicing, P4, Wired Communications

I. Introduction

THE MULTIPROTOCOL Label Switching (MPLS) protocol defined in RFC 3031 [1] has become a predominant wide-area networking technology. In MPLS, packets are equipped with labels adding domain-specific forwarding information. Packets are switched to intermediate nodes based on the assigned label. Since the standardization in 2001, the protocol has continuously evolved. In the last years, new applications have emerged that require labels to not only contain forwarding information but also information on how to process a packet. Therefore, special purpose labels (SPLs) have been defined. Examples of applications using SPLs include entropy labels which support equal-cost multi-path (ECMP) [2], and labels for No Further Fast Reroute (NFFRR) [3] which avoid the looping of packets during network failures. Further, labels can embed service function chaining (SFC) information [4], [5], and data related to operations, administration and maintenance (OAM) [6], [7].

The IETF has proposed the MPLS Network Actions (MNA) framework [8] which is currently in the process of standardization to facilitate extensions in the MPLS protocol. The MNA framework provides an encoding for network actions and their data in the MPLS label stack. These network actions and their data can be either encoded as in-stack data (ISD) in the MPLS stack or as post-stack data (PSD) after the MPLS stack. Since the MNA framework serves as a foundation for future extensions of MPLS, it must be efficiently supported by hardware. Extension headers (EHs) in IPv6 are a similar concept to the network actions introduced in the MNA framework. However, various works [9]–[13] have shown that IPv6 EHs are not widely adopted because they are not efficiently implementable on hardware.

The contribution of this paper is manifold. We provide an introduction to the current state of the MNA framework and the first use cases identified by the working group. We compare the concept of the MNA framework to IPv6 EH and explore the feasibility for hardware implementations. Then, we implement the MNA framework based on the IETF MPLS working group (WG) proposal [14] in P4 on the Intel Tofino™ 2 switching ASIC using ISD. Our implementation is the first and to date the only implementation of the MNA framework. We evaluate the scalability and performance of the P4-MNA implementation. Further, we implement and evaluate two example network actions, namely the alternate-marking method (AMM) for performance measurement, and bandwidth reservation using network slicing. Based on our implementation experience, we propose to extend the signaling of hardware capabilities within an MNA-capable MPLS domain so that MPLS nodes with less capabilities can be integrated. Finally, we identify challenges arising from using ISD, propose solutions to address them in future work, and summarize security considerations of the MNA framework.

The rest of the paper is structured as follows. In Section II, we provide a primer on the MNA framework, including information on MPLS, SR-MPLS, SPLs, and MNA. An overview of identified use cases by the IETF MPLS WG and their mechanisms is given in Section III and in Section IV, we compare IPv6 EH with MNA and review related work. In Section V, we explain the concept of the programming language P4. The implementation of the MNA framework on the Intel Tofino™ 2 is described in Section VI. In Section VII we evaluate the P4-MNA implementation and the exemplary network actions. In Section VIII, we describe constraints for header stacking and propose a signaling extension, and in Section IX, we identify challenges with mutable data. In Section X, we describe security risks currently considered in the MNA framework. Finally, we conclude the paper in Section XI.

II. A Primer on MPLS Network Actions (MNA)

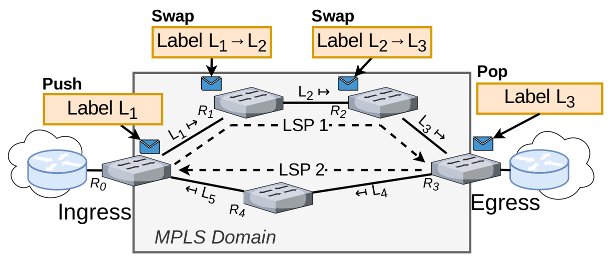
In this section, we give a brief overview of traditional MPLS networks including forwarding in MPLS and special-purpose labels. Then, we explain the concept of the MPLS Network Actions (MNA) framework as being standardized by the IETF MPLS WG [15]. This includes the proposed MNA header encoding, and the placement of network actions in the MPLS label stack.

A. Traditional MPLS Networks

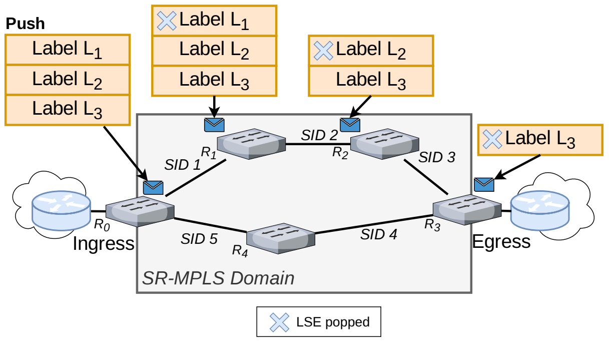
This section briefly summarizes the current state of the art for forwarding and extensions in MPLS networks.

1) MPLS Forwarding

Nodes in an MPLS network are called Label Switching Routers (LSRs). Initially, virtual circuits have been set up in MPLS along a path of LSRs. These virtual circuits are called Label Switched Paths (LSPs) and the virtual connection identifier is called a label. The ingress and egress of an LSP are called ingress Label Edge Router (LER) and egress LER. LSPs are established through signaling prior to communication. Packets are forwarded by LSRs using label switching according to the entries of a forwarding table that contains information about incoming interface and label, and outgoing interface and label. At the egress LER, the MPLS label of the LSP is popped and the packet is delivered to its destination using the underlying Layer 3 protocol. An exemplary network using virtual circuits in MPLS is shown in Figure 1(a).



(a) LSPs are set up prior to communication and labels are switched according to LSP-specific entries in the forwarding tables.



(b) With SR-MPLS, paths are encoded in packet headers as label stacks. During forwarding, LSRs pop the top-of-stack label.

FIGURE 1. **MPLS forwarding using virtual circuits and SR-MPLS.**

About 10 years ago, segment routing (SR) has been introduced which turned MPLS into a connection-less and source-routed packet switching technology. Labels are interpreted as segment identifiers (SIDs) which denote next intermediate nodes to where the packet is forwarded on the way to the egress LER. In SR, there are two types of segments. An adjacency segment describes a strict forwarding instruction over a specific link between two nodes while a prefix segment describes a loose forwarding instruction to a prefix over multiple hops [16]. For an adjacency segment, a node pops the top-of-stack label and forwards the packet to the next segment [17]. For a prefix segment, the top-of-stack label is not popped until the prefix is reached. By stacking multiple labels, a source route is defined. Thus, with SR, the ingress LER pushes a label stack and intermediate hops pop these labels. There is no connection concept with SR-MPLS and therefore no signaling protocol is required to set up connections. However, SIDs need to be signaled and label stacks need to be computed. An exemplary network using SR-MPLS with adjacency segments is shown in Figure 1(b). In both examples, penultimate hop popping is not applied, i.e., the egress node receives a label, pops it, and forwards the packet based on the underlying Layer 3 information.

The ingress LER pushes one or more MPLS labels onto the MPLS stack of a packet. An entry in this header stack is called a Label Stack Entry (LSE). In Figure 2, the MPLS header is visualized. It is 4 B large and consists of a 20 bit label, 3 bits traffic class (TC), a bottom-of-stack bit (S), and a 8 bit time-to-live (TTL).

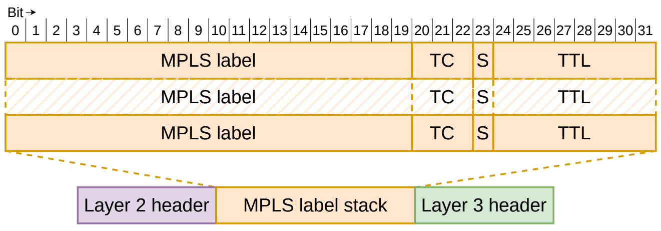


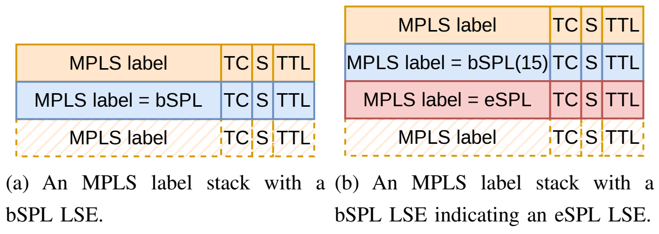
FIGURE 2. The MPLS label stack is located between the Layer 2 header and the Layer 3 header. An MPLS stack consists of one or more LSEs, each of which consists of a label value, a traffic class (TC), a bottom of stack bit (S), and a time-to-live (TTL) field.

2) MPLS Special Purpose Labels

MPLS labels do not only contain addressing information but also encapsulate information for special purposes. A special label value range is reserved for these purposes and must not be used for forwarding [18]. They are called base Special Purpose Labels (bSPLs) and indicate an operation to be performed by the LSR. An MPLS label stack containing a bSPL LSE is illustrated in Figure 3(a).

For bSPLs, only 16 reserved values are available. Therefore, the IETF defined the specific bSPL label with value 15, called an extension label, which indicates that the LSE following that bSPL contains extended Special Purpose Label (eSPL) values [18]. For eSPLs, more label values are reserved and available for extensions. An MPLS label stack containing an eSPL LSE is illustrated in Figure 3(b).

However, currently only two eSPL values are allocated which are used for SFC in RFC 8595 [4]. For these eSPL values, it is assumed that no other extension label is present in the MPLS stack [19]. This makes it difficult to combine



(a) An MPLS label stack with a bSPL LSE. (b) An MPLS label stack with a bSPL LSE indicating an eSPL LSE.

FIGURE 3. MPLS label stacks containing bSPL and eSPL LSEs [18].

SFC eSPLs with other extensions such as the alternate-marking method (AMM). Further, MPLS extensions often require modifications to existing specifications [8].

B. The MNA Framework

This section summarizes the concept of the MNA framework, including an overview, the header encoding for network actions, and scopes in MNA proposed by the IETF MPLS working group.

1) Overview

The MNA framework provides a general mechanism for the transmission and processing of predefined network actions and their required data to facilitate extensions to the MPLS protocol [8]. To that end, the working group proposed a new header encoding for network actions in the MNA framework [14]. Network actions are either located in-stack or post-stack. In this work, we consider in-stack network actions. Generally, a network action in MNA contains an opcode that identifies the operation that the LSR will perform on the packet. Those network actions are encoded as LSEs in the MPLS stack. A packet can contain multiple network actions.

A network action may require input parameters to process the indicated network action or may write the result from the processed network action into the packet header. This data is carried in the MPLS stack leveraging the MNA encoding and is called ancillary data (AD).

2) The MNA Header Encoding

In MNA, a stack of related LSEs in the MPLS stack containing network actions and AD is called a Network Action Sub-stack (NAS) [8], [14]. A NAS is inserted below a forwarding label in the MPLS stack. Multiple NAS can exist in an MPLS stack. Figure 4 shows an example of two NAS inserted into an MPLS stack with stacked forwarding labels.

A NAS is comprised of multiple LSEs with different purposes. Essentially, a NAS is a bSPL LSE followed by network action LSEs. For network action LSEs, the traditional MPLS LSE encoding is repurposed. A NAS consists of four differently encoded LSEs listed below [14].

- The NAS indicator LSE (Format A),

- the initial opcode LSE (Format B),
- the subsequent opcode LSE (Format C),
- the ancillary data (AD) LSE (Format D).

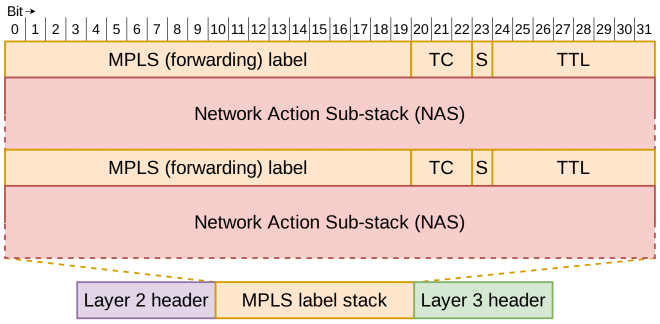


FIGURE 4. An example of two NAS inserted into an MPLS label stack. A NAS consists of at least two LSEs and up to 17 LSEs.

For network actions and AD LSEs, the encoding of an MPLS LSE is repurposed as shown in Figure 5.

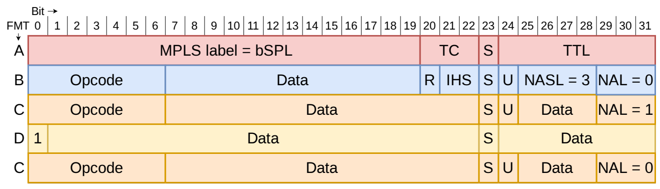


FIGURE 5. The four different LSE encodings of a Network Action Sub-stack (NAS) in the MNA framework.

In the following, the different encodings are explained. A NAS starts with a bSPL value (*Format A*) that indicates the beginning of the NAS. This LSE is referred to as NAS indicator. Following the NAS indicator, the mandatory initial opcode LSE (*Format B*) defines the first network action to be processed in this sub-stack. The repurposed fields in the initial opcode format are shown in Figure 5 and are briefly described below [14].

- Opcode: the operation code that identifies this network action.
- Data: AD belonging to this network action, e.g., flags. 13 bits are available for AD in Format B.
- IHS: the scope of this sub-stack. This can be either ingress-to-egress (**I**2E), hop-by-hop (**H**BH), or **S**elect. More on this in Section II-B3.
- Network Action Sub-stack Length (NASL): the number of additional network actions and AD LSEs in this NAS excluding the Format B LSE, i.e., the length of this NAS.
- Network Action Length (NAL): the number of AD LSEs following and belonging to this network action.

The NASL and the IHS field defined in the Format B LSE are valid for the entire NAS. The R, S, and U fields are not relevant in this work. The bare minimum NAS consists of the NAS indicator (Format A) LSE and the initial opcode (Format B) LSE. Only one Format B LSE is allowed in an NAS.

Optionally, subsequent opcode LSEs (Format C) or AD LSEs (Format D) follow. The *Format C* LSE is a simplified version of the Format B encoding. The *Format D* LSE contains AD which relates to the preceding Format B, or Format C LSE. AD can be carried either as part of a Format B or Format C LSE, or as a separate AD LSE in Format D. In the Format C LSE, 20 bits are available for opcode-specific AD. Finally, the Format D LSE contains 30 bits for opcode-specific AD. The semantics of the AD are left to the predefined network action. A NAS can contain up to 16 network actions and AD LSEs. Together with the NAS indicator, a NAS therefore contains up to 17 LSEs. After exposing a NAS to the top, it must be popped.

3) MNA Scopes and MNA with SR-MPLS

Network actions in the MNA framework can be processed on selected nodes, on the egress node only, or on all nodes along a path. To that end, a NAS specifies the scope of the contained network actions in the IHS field of the Format B LSE. Available scopes in the proposed MNA header encoding are *select*, *ingress-to-egress (I2E)*, and *hop-by-hop (HBH)* [14]. In the following, the three scopes are described. Then, an example for the placement of differently scoped NAS in the MPLS stack is given in Figure 6.

A *select*-scoped NAS is processed by one specific node on the path. It is located below the forwarding label for the specific node. Only the node that exposes this NAS to the top of stack, i.e., that pops the preceding MPLS forwarding label, processes the select-scoped NAS. This NAS is popped after processing.

The *I2E*-scoped NAS is only processed by the egress LER. The I2E scope provides ingress to egress transport of network actions and is a separate scope to provide data from the ingress node specific to the egress node. It is placed at the bottom of the MPLS stack.

HBH-scoped network actions must be processed on each node along the path from source to destination. An HBH-scoped NAS is located deeper in the stack, i.e., below another forwarding label than the top-of-stack label. Therefore, an LSR must search deeper in the MPLS stack to find an HBH-scoped NAS. This problem is further described in Section VIII.

Figure 6 illustrates the differently scoped NAS and their placement in an exemplary MPLS label stack in an SR-MPLS environment of two LSRs R_1 and R_2, and one egress LER R_3. Further, Figure 6 shows which nodes process the differently scoped NAS.

The MPLS stack in Figure 6 encodes MPLS forwarding labels for adjacency segments. In this example, LSRs parse the entire MPLS stack. Therefore, in Figure 6, R_1 executes network actions in the select-scoped NAS located below the top-of-stack forwarding label, and the HBH-scoped NAS

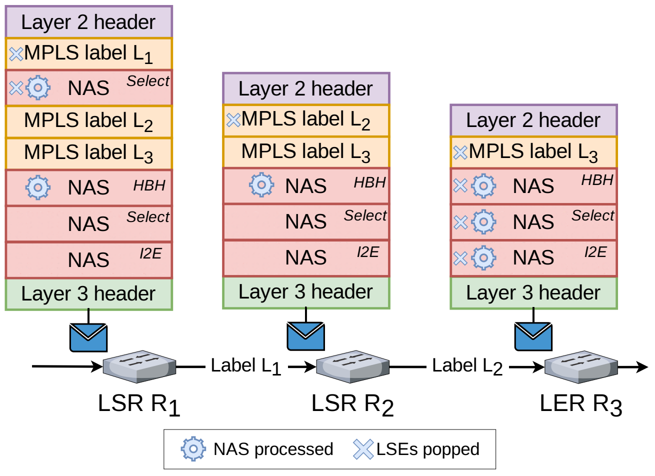


FIGURE 6. An example MPLS label stack in an SR-MPLS environment with adjacency segments. The HBH-scoped NAS is processed by all nodes, the select-scoped NAS are processed by nodes that pop the preceding forwarding label, and the I2E-scoped NAS is processed by the egress LER.

located below the forwarding label L_3. R_1 pops the select-scoped NAS after popping L_1. R_2 executes network actions in the HBH-scoped NAS. R_3 executes network actions in the HBH-scoped NAS, the select-scoped NAS, and the I2E-scoped NAS located below its forwarding label. In this example, penultimate hop popping is not applied. With penultimate hop popping, R_2 removes the last forwarding label but must not remove the NAS exposed to the top. Then, R_3 may receive a NAS at the top of stack [14].

III. Identified Use Cases For the MNA Framework

In this section, we describe five use cases identified by the IETF working group [20], namely no further fast reroute (NFFRR), in-situ OAM, SFC, AMM, and network slicing. Many of the identified use cases for MNA exist as a technology-agnostic mechanism and are implemented and adapted for the MPLS protocol. Table 1 shows a summary of the identified use cases for the MNA framework. For each use case, the technology-agnostic mechanism, the adaption to MPLS, and a proposal for the MNA framework are shown. In the following sections, the use cases are described in more detail.

A. No Further Fast Reroute (NFFRR)

The MPLS Fast Reroute (FRR) mechanism defined in RFC 4090 [37] is a well-established mechanism to counter link and node failures in an MPLS network. In FRR, backup tunnels are established to bypass a section of an LSP in case of a failure. Kompella *et al.* [3] identified the problem of looping packets in a network with multiple failures where FRR is applied multiple times, eventually leading to congestion and packet loss. An example is illustrated in Figure 7 and explained below.

TABLE 1. Overview of identified use cases for MNA.

Use case	Technology-agnostic mechanism	Adaption to MPLS	Proposal for MNA
NFFRR	-	Kompella *et al.* [3]	Saad *et al.* [21]
IOAM	RFC 9197 [22], RFC 9326 [23]	RFC 5586 [24], RFC 6669 [25]	Gandhi *et al.* [6], Mirsky *et al.* [26]
SFC	RFC 7665 [27], RFC 8300 [28]	RFC 8595 [4]	MNA use cases draft [20]
AMM	RFC 9341 [29], RFC 9342 [30]	RFC 8372 [31], Cheng *et al.* [32]	Cheng *et al.* [33]
Network slicing	RFC 9543 [34]	Saad *et al.* [35]	Li *et al.* [36]

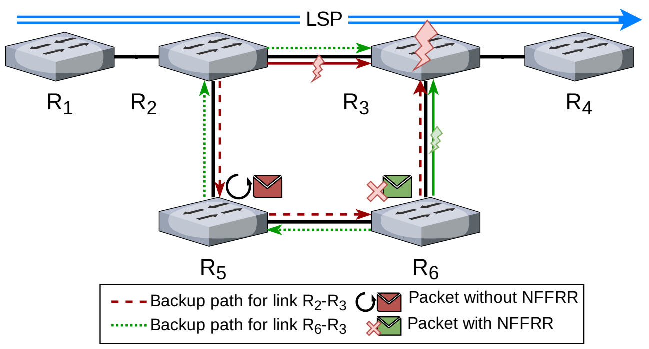


FIGURE 7. In the network topology, an LSP R_1–R_2–R_3–R_4 is established. The link R_2–R_3 is protected by the path R_2–R_5–R_6, and the link R_6–R_3 is protected by the path R_6–R_5–R_2. If the link R_2–R_3 and R_6–R_3 fail at the same time, packets loop between the backup paths. With NFFRR, packets are marked on the first reroute and dropped on a second reroute [3].

In Figure 7, an LSP from node R_1 to node R_4 over R_2 and R_3 is established. If the links R_2–R_3 and R_6–R_3 fail simultaneously, e.g., by a node failure of node R_3, a packet from R_1 arriving at R_2 is rerouted using the backup tunnel R_2–R_5–R_6. Once the packet arrives at node R_6, the packet is again rerouted using the backup tunnel R_6–R_5–R_2 because the link R_6–R_3 is down. The packet is rerouted to node R_2 where it is again rerouted to node R_6. Packets loop between R_2, R_5, and R_6 until the TTL expires. This leads to congestion of the links R_2–R_5 and R_5–R_6.

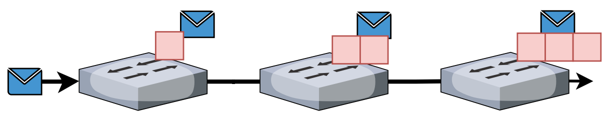
As a solution, Kompella *et al.* propose that an LSR adds a bSPL MPLS label if a packet was rerouted via a bypass FRR tunnel and if another bypass via a FRR tunnel is not desired. A packet carrying such a bSPL allows an LSR to distinguish fast-rerouted packets from regular packets. The LSR that encounters such an bSPL MPLS label must not send the packet over a FRR tunnel. With this approach, node R_2 marks the packet in Figure 7 with the NFFRR bSPL. Node R_6 does not reroute the packet back to node R_2 but drops it because the NFFRR bSPL label is present.

However, the number of available bSPLs is limited. Therefore, Saad *et al.* [21] propose a network action to leverage the MNA framework for NFFRR. They propose to indicate whether a packet was rerouted with a single bit in the AD of a network action.

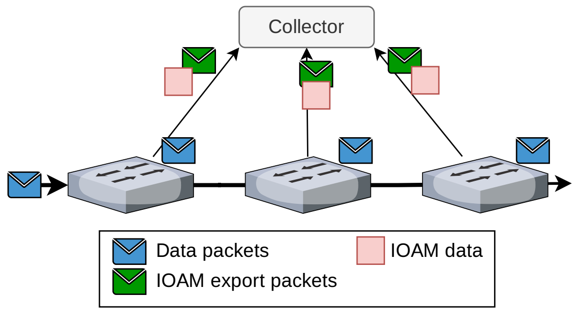
B. In-Situ OAM

In-situ OAM (IOAM) collects operational and telemetry information as packets traverse a specific path. This information can be leveraged for traffic engineering and monitoring. IOAM does not send probe packets to collect metrics. Instead, the information is added to or the collection is triggered by existing data packets.

IOAM data can be collected in two ways. First, by adding information to each packet and evaluating the information at the tail-end. Second, by exporting information from packets received by the LSR to a collector in the network, e.g., the control plane. The first method is called the passport mode and the second is called the postcard mode. The methods are illustrated in Figure 8 [23].



(a) Passport mode.



(b) Postcard mode.

FIGURE 8. Operation modes in IOAM.

In the passport mode in Figure 8(a), each LSR places a “stamp”, i.e., the IOAM data, on the “passport”, i.e., the header, of a packet. The passport mode is used in RFC 9197 [22], e.g., for path tracing. RFC 9326 [23] introduces the direct export option which leverages the postcard mode for data collection. In the postcard mode in Figure 8(b), each LSR “writes a postcard”, i.e., sends a message containing the IOAM data to the collector. The information is either exported as raw data per packet or aggregated, processed, and exported using protocols such as IPFIX. No information is added to the data packet with the postcard mode. Which mode to use and what data to collect is indicated in the packet, e.g., with bSPL labels in MPLS [24].

Gandhi *et al.* [6] propose an approach to encapsulate IOAM data in MPLS data planes. They leverage the MNA framework for this approach. Their draft encapsulates the IOAM data from RFC 9197 and RFC 9326 in a post-stack network action. Mirsky *et al.* [26] propose an alternative approach that encapsulates the direct export option from RFC 9326 in an in-stack network action.

C. Service Function Chaining (SFC)

SFC steers packets through a defined set of network functions, such as firewalls, IDS, and NAT. Packets are classified at the network edge and the classification decision is stored in an SFC header. This header is leveraged to direct packets to the desired network functions for processing [27]. To that end, SR-MPLS or the network service header (NSH) can be used [5]. An example of a SFC using SR-MPLS is given in Figure 9.

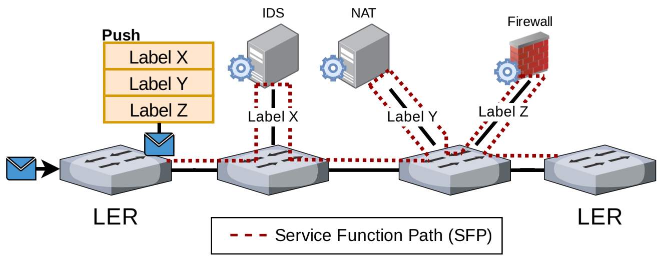


FIGURE 9. A packet is classified at the ingress LER. Based on the classification, an SR-MPLS stack is pushed onto the packet that encodes the service function path (SFP). This path directs the traffic to various service functions.

In Figure 9, the ingress LER classifies a packet and adds an SR-MPLS stack which encodes the service function path (SFP). The SFP steers the traffic to different service functions. Alternatively to SR-MPLS, the NSH can be used to direct traffic to the service function. The NSH defined in RFC 8300 [28] describes an encapsulation for SFCs.

RFC 8595 [4] introduces an MPLS-based forwarding plane for SFC leveraging eSPL values. Two consecutively stacked MPLS labels encode the NSH in an MPLS label stack. The fields of an LSE are repurposed to reflect the data encoded in an NSH. However, this imposes some limitations on the NSH. Because the 24-bit wide service path identifier of an NSH is mapped to the 20-bit wide MPLS label field, the MPLS representation of NSH must not assign values that exceed the size of an MPLS label. Furthermore, the eight-bit wide service index, which indicates the location of the service function within the path, is mapped to the second MPLS label, leaving 12 bits unused.

The proposed MNA framework is a generalization of the concept described in RFC 8595. In MNA, the NSH can be encoded more efficiently in the MPLS label stack without imposing restrictions. To that end, the eSPL values can be introduced as network action opcodes. The service-path identifier, the service index, and metadata can be encoded as AD in a NAS.

D. Performance Measurement with Alternate-Marking Method (AMM)

The alternate-marking method (AMM) defined in RFC 9341 [29] and RFC 9342 [30] is a mechanism to measure packet loss and delay. In general, packet loss can be detected using sequence numbers in packets. However, this requires the insertion of sequence numbers into packets. Further, devices must be able to extract and verify the sequence number [29]. AMM is an alternative approach to measure the packet loss and delay between two points. For packet loss measurement, the number of sent packets is counted on the sending end and the number of received packets is verified on the receiving end. If the number of sent packets is not equal to the number of received packets, packets are lost. AMM requires synchronization of both sides, i.e., they must be referring to the same set of packets. For this purpose, a flow is divided into batches by marking packets of the same batch and the same flow with the same color. For each consecutive batch, the color is alternated. On changing the color, the number of sent packets in the previous batch is exported via an out-of-band channel, e.g., to the control plane. This allows the control plane to verify the number of packets of the same color. The number of packets in a batch is either fixed or based on the packet rate and a fixed timer. For delay measurement, the timestamp of the color change event is exported to the control plane. An example using AMM for packet loss measurement is given in Figure 10.

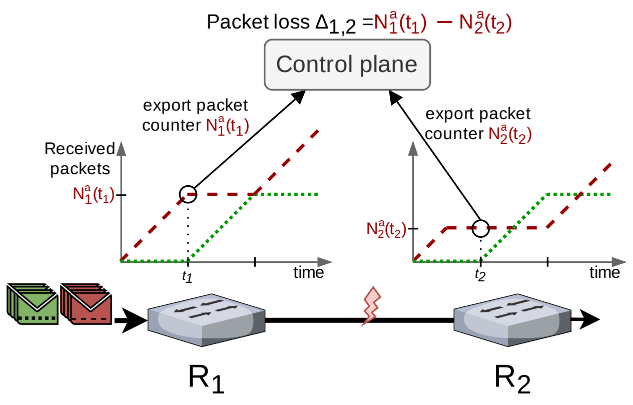

Packet loss $\Delta_{1,2} = N_1^a(t_1) - N_2^a(t_2)$
Control plane
export packet counter $N_1^a(t_1)$
export packet counter $N_2^a(t_2)$
Received packets
$N_1^a(t_1)$
$N_2^a(t_2)$
t_1
time
t_2
time
R1
R2

FIGURE 10. Packets are sent in colored batches. Each LSR counts the packets per color. If the color alternates, the packet counter is exported to the control plane. The control plane calculates packet loss based on the received counters.

In Figure 10, colored packet batches are sent over an LSP of two LSRs R_1 and R_2. The color-specific packet counters per LSR over time are shown in the plots. At t_1 the color alternates in LSR R_1, and at t_2 in LSR R_2. Therefore, the LSRs R_1 and R_2 export their counter values of color a, $N_1^a(t_1)$ and $N_2^a(t_2)$, to the control plane. The control plane then calculates the packet loss on the link from node R_1 to R_2 of this batch $\Delta_{1,2} = N_1^a(t_1) - N_2^a(t_2)$.

Packets are colored by setting a bit in the header, such as the drop eligible indicator bit for Ethernet frames. Other header fields may be used depending on the application. Cheng *et al.* [32] describe an approach to leverage eSPL entries to implement AMM in MPLS networks. In [33], Cheng *et al.* describe another approach to leverage the MNA framework to implement AMM in MPLS networks. Here, a NAS with the AMM network action is described. The encoding proposed by [33] for an AMM network action in MNA is shown in Figure 11.

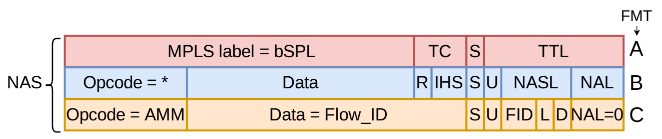


FIGURE 11. The AMM network action is indicated by an opcode in the Format C LSE. The data field of the network action contains the flow ID and the packet color [33].

AMM operates on a per-flow basis. The 18-bit wide flow ID is embedded into the data field of the network action shown in Figure 11. The network action including its AD does not fit into a Format B LSE and requires a Format C LSE. The flow ID is distributed across the 16 bits of the first data field and the two most significant bits of the second data field. The two least significant bits of the data field indicate the color for packet loss measurement (L) and delay measurement (D). This network action is implemented in P4-MNA for packet loss measurement and evaluated in Section VII-C2.

E. Network Slicing

An IETF network slice, defined in RFC 9543 [34], provides connectivity coupled with a set of specific commitments of network resources, such as reserved bandwidth or latency. Network slices provide end-to-end logical networks over a shared physical infrastructure for various applications, such as 5G networks and VPNs. They act as an overlay network, creating multiple isolated virtual networks. A network slice may stretch across multiple domains of a provider. The configuration of a network slice, such as nodes, links, buffers, queuing resources, and scheduling resources, is described in a network resource partition (NRP). An NRP must be configured prior to operation, e.g., with netconf, or via IGP signaling [35]. Each NRP is identified by an NRP selector which is added to each packet. The NRP selector indicates which logical network, i.e., network slice, a packet belongs to. An NRP selector consists of one or more fields in the packet, e.g., an IPv6 address, or an MPLS label [35]. A switch must perform operations on a packet based on the NRP selector, e.g., traffic shaping to enforce bandwidth reservations. An example network topology with two NRPs sharing a physical infrastructure is shown in Figure 12.

Two NRPs are configured in the network in Figure 12. NRP X (dotted) has low latency and high-reliability requirements, e.g., for 5G networking. NRP Y (dashed) has high bandwidth and low latency requirements, e.g., for video streaming. The requirements, e.g., bandwidth reservations,

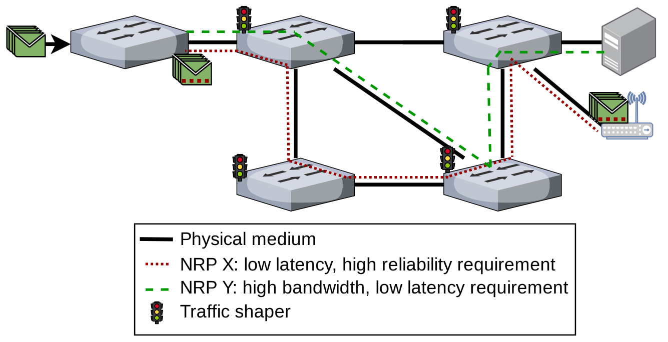

Physical medium
NRP X: low latency, high reliability requirement
NRP Y: high bandwidth, low latency requirement
Traffic shaper

FIGURE 12. Two NRPs sharing the same physical infrastructure. They have different traffic requirements that are enforced by the LSRs.

are configured in the LSRs. Both NRPs share the same physical medium but their traffic is isolated in two logical networks. At the ingress node, traffic is classified to identify to which NRP a packet belongs and a NRP selector is pushed to the packet. Each transit node in the network enforces the NRP by traffic shaping and scheduling.

In [35], Saad *et al.* describe an approach for implementing network slices in MPLS networks. Here, LER classify incoming traffic and push an NRP selector label onto the MPLS label stack. With the MNA framework, the NRP selector can be carried as a network action in the MPLS stack. Li *et al.* propose an encoding for the NRP selector in a network action which is shown in Figure 13 [36].

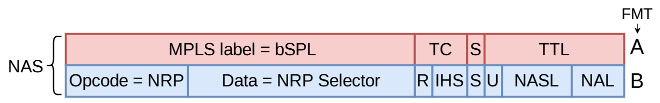


FIGURE 13. The NRP network action is indicated by an opcode in the Format B LSE. The data field of the network action contains the NRP selector [36].

A network action for bandwidth reservations with network slicing is implemented in P4-MNA and evaluated in Section VII-D2.

IV. Related Work

First, we compare the concept of IPv6 EHs with the MNA framework. Then, we review related work that identifies shortcomings in IPv6 EHs. Finally, we review implementations of IPv6 EHs.

A. Comparison of IPv6 Extension Headers with MNA

RFC 2460 [38] defines IPv6 EHs to expand the functionality of IPv6. IPv6 EHs contain options that indicate special operations such as fragmentation or allow for segment routing (SR) over IPv6 (SRv6). Table 2 compares the concepts from IPv6 EH with the MNA framework.

IPv6 EHs are added after the IPv6 base header with their presence and purpose indicated by the next header field in the base header. Similarly, in MNA, network actions are placed after a forwarding label. However, unlike IPv6, MPLS

TABLE 2. Comparison of IPv6 EH and MNA.

	IPv6 EH	**MNA**
Indication	Next header field	Indicator LSE (bSPL) followed by opcodes
Scopes	hop-by-hop (HBH), destination (I2E)	Select, HBH, I2E
Upper bound on # extensions	Unbounded, chaining of extensions via next header field	16 network actions per NAS, 1 NAS per scope
Upper bound on extension size	$(2^8-1)\ B = 255\ B$ per option from 8-bit length field in TLV	$4\ B + (2^4-1) \cdot 4\ B = 64\ B$ in a NAS from 4-bit NASL length field
Deployment	Global	Limited to a domain

does not use next header fields. Instead, network actions are indicated by a label containing a bSPL value followed by LSEs containing opcodes.

Both mechanisms allow to specify the scope of actions, i.e., on which node an action is executed. IPv6 EHs can either be processed only by the destination node (using the destination option) or by every node on the path (using the hop-by-hop option). MNA further enhances this by allowing network actions to be processed on selected nodes in the path, offering more granular control.

In terms of structure and size, IPv6 EHs provide high flexibility. They can be concatenated indefinitely by updating the next header field allowing an unbounded number of extensions per packet. However, MNA imposes stricter limits. The MNA framework supports up to 16 network actions per NAS with one NAS allowed per scope. This bounded approach simplifies implementation but reduces flexibility compared to IPv6 EHs.

IPv6 EHs support options encoded in a type-length-value (TLV) format with a theoretical maximum of 255 B per option due to the 8-bit integer width of the length field. The length field indicates the number of bytes. In contrast, the MNA framework limits the size to 64 B per NAS by the 4-bit wide NASL field offering a smaller size bound. The NASL field indicates the number of LSEs, i.e., 4 B, excluding the Format B LSE.

IPv6 allows for global, inter-domain communication between devices. This makes it difficult to deploy extensions, such as new options in a EH, because devices around the world must implement new extensions to take advantage of them. In contrast, MPLS is typically deployed by providers as a Layer 2 and Layer 3 technique within their own domain. This facilitates the deployment of extensions, such as the MNA framework or new network actions in the MNA framework. The usefulness of MNA in a provider's domain does not depend on the deployment of MNA in other domains as providers can choose which network actions to support in their own domain.

B. Analysis of IPv6 Extension Headers

While IPv6 EHs appear more flexible due to their unbounded concatenation and larger extension sizes, this flexibility introduces challenges for hardware implementations. In the following, we discuss related work on those hardware implications for IPv6 EHs.

Custura *et al.* [10] analyze the transit of packets containing IPv6 EHs on the Internet. They perform several experiments in which traffic containing different IPv6 EH types and sizes is sent to destinations around the world using RIPE Atlas probes. In their results, the measured traversal rate decreases with an increasing EH size. The authors of [39] identified that the traversal rate is halved with an EH size of $64\,\mathrm{B}$ and significantly lower with an even larger size. They conclude that the successful reception of such a packet depends on the type of EH it contains, its size, and the transport protocol used.

Gont *et al.* [12] describe a reason for the limited forwarding of packets containing IPv6 EHs. To efficiently process packets at high data rates, the packets must be processed in the fast path of packet forwarding engines, such as in the data plane of hardware implementations. The large and dynamic header size of IPv6 EHs requires the lookup engine to inspect deeply into the packet. Engines that cannot inspect deep enough into a packet to extract all relevant information typically drop the packet. Another approach used by some packet forwarding engines with limited lookup capabilities, as described in [12], is recirculation. In this approach, one IPv6 EH is processed at a time. The packet is then looped back to the ingress and processed again until all EHs have been processed. By using recirculation, each packet traverses the processing pipeline multiple times. If not enough resources for recirculation are available, the performance degrades and packets are dropped.

Routers that are unable to inspect deep enough into a packet, or that do not use recirculation, can process packets in the slow path, i.e., in the control plane software [10], [12]. Processing large packet headers in the control plane drastically reduces the performance. In addition, control plane processing consumes the resources needed to manage the router. This facilitates denial-of-service attacks such as described in [9]. Here, a $285\,\mathrm{B}$ arbitrary payload is added with multiple IPv6 options that forces the control plane to validate this payload wasting CPU resources and resulting in a denial of service. Furthermore, the large size of EHs prevents firewalls from inspecting transport layer information, making stateful filtering intractable [11]. Since there is no easy solution yet, they suggest dropping packets containing certain IPv6 EHs as a temporary solution. They further emphasize that this problem does not arise from a vendor or manufacturer issue but rather from a flaw in the protocol design that allows such large headers to be created.

With IPv6 EHs being standardized and deployed for many years, it has been proven [9]–[13] that implementations cannot efficiently support EHs to their full extent. The protocol design problem of IPv6 EH lies in the large header stacks resulting from chained EHs and bloated options [13]. They cannot be processed efficiently on hardware. IPv6 EHs can be chained arbitrarily with large payloads. In the MNA framework, network actions are stacked. However, with MNA, the maximum size of network actions and their data is limited by the protocol design. While the MNA framework is more constrained regarding the structure, number, and size of network actions, it is also more feasible for hardware implementations. This facilitates the implementation of MPLS extensions in the future. However, processing many network actions in the fast path remains a challenge due to limited hardware resources. In this work, we therefore investigate how many network actions can be processed in the fast path on hardware, i.e., in the data plane.

C. Implementations of IPv6 Extension Headers

In this section, we provide an overview of existing implementations of IPv6 EHs in both hardware and software. To the best of our knowledge, no other implementation of the MNA framework than P4-MNA currently exists.

The authors of [40] provide a programmable segment routing over IPv6 (SRv6) processor for SFC which is based on a FPGA development board. The implementation includes SRv6 processing and SFC encapsulation. Their evaluation shows that their implementation achieves an SRv6 throughput of $100\,\mathrm{Gb/s}$ for $1500\,\mathrm{B}$ packets. However, the implementation achieved approximately $70\,\mathrm{Gb/s}$ for $128\,\mathrm{B}$ packets. In [41], the authors provide an implementation of SRv6 in the Linux kernel, i.e., a software-based implementation. Their implementation supports SRv6 and an HMAC TLV option. They tested their implementation in a $10\,\mathrm{Gb/s}$ testbed. The authors of [42] provide an IOAM implementation encapsulated in IPv6 for the Linux kernel. Their software-based approach achieved approximately $4\,\mathrm{Gb/s}$ for $78\,\mathrm{B}$ packets and $41\,\mathrm{Gb/s}$ for $1236\,\mathrm{B}$ packets. A whitepaper in [43] evaluates a software-based implementation of SRv6 using the VPP framework. They achieve a forwarding rate of $48.43\,\mathrm{Gb/s}$ using $192\,\mathrm{B}$ packets. Additionally, they present an FPGA-based implementation that achieves up to $96.84\,\mathrm{Gb/s}$. Software-based implementations, while flexible, are typically limited by general-purpose CPUs and cannot achieve the throughput required for high-speed transit networks operating at hundreds of gigabits per second.

While existing implementations of IPv6 EHs achieve up to $100\,\mathrm{Gb/s}$ under certain conditions, the P4-MNA implementation presented in this paper achieves line rate processing of $400\,\mathrm{Gb/s}$ per port for both small ($128\,\mathrm{B}$) and large ($1500\,\mathrm{B}$) packet sizes. The P4-MNA implementation is capable of processing 32 network actions which are closely related to options in IPv6 EHs such as IOAM and SFC.

V. Introduction to the P4 Programming Language

In this section, we give an introduction to the programming language P4.

Programming Protocol-independent Packet Processors (P4) is a domain-specific high-level programming language for describing the data plane of programmable switches. P4 can be used to implement user-defined algorithms for packet manipulation and forwarding decisions. A P4 program can be compiled for different targets that implement a specific architecture. Such architectures can be either software-based, like the simple_switch in the BMv2 [44], or hardware-based, like the Intel Tofino™ 2 switching ASIC.

A P4 program consists of a programmable packet parser, several programmable control blocks, and a programmable packet deparser that are arranged sequentially in a pipeline. The pipeline of the Intel Tofino™ 2 switching ASIC is illustrated in Figure 14 and the components are further described in the following.

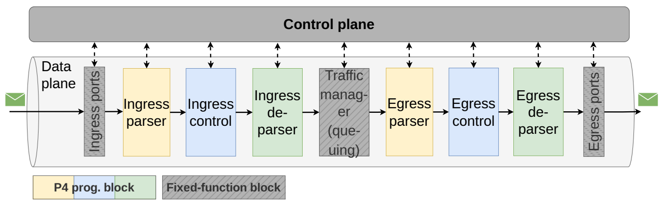


FIGURE 14. The pipeline of the Intel Tofino™ 2 switching ASIC consists of a programmable packet parser, control blocks, and a programmable packet deparser for ingress and egress control [45].

A P4 program specifies user-defined metadata and packet headers. User-defined metadata store values during packet processing and do not exit the switch. It is comparable to variables in other languages. User-defined packet headers describe the packet headers that are accessible in the P4 program. They are parsed by the P4 parser and can be manipulated during pipeline processing.

The P4 parser is modeled as a finite state machine and extracts header information from the packet according to the user-defined parser states. To that end, user-defined packet headers are defined in the P4 program reflecting the header fields, e.g., an MPLS LSE. Multiple of these headers can be aggregated to form a header stack of that specific header type, e.g., an MPLS label stack, or a NAS. Entries in a header stack must all share the same header field structure, i.e., different encodings are not possible in a single header stack. A header stack in P4 is comparable to an array in other languages. As is usual with arrays, the maximum memory required for an array must be allocated before runtime and cannot be changed dynamically. Several header stacks and headers can be concatenated to form the complete packet header. The remaining payload in the packet is ignored and passed on by the P4 program [45], [46].

During parsing, bytes are parsed into different user-defined headers, e.g., LSE encodings. An LSE parsed into the Format B encoding contains more information about the semantics of the LSE than an LSE parsed as raw bytes. The knowledge about the semantics gained during parsing facilitates later processing in the control block.

Control blocks in a P4 program contain the logic of the algorithm. They consist of match+action tables (MATs) and can make use of simple arithmetic and logical expressions as well as branching constructs to define the packet processing operations. The principle of MATs is illustrated in Figure 15.

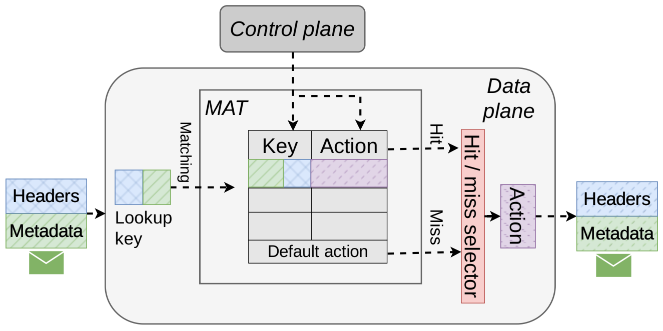


FIGURE 15. Selected header fields of a packet form a composite key and are matched in a MAT. An associated action is executed. The content of the MATs is filled by the control plane [47].

A MAT consists of a key definition and an action list. The key definition comprises selected header fields that are matched in the table. Upon matching a packet to an entry in the MAT, the associated action from the action list is executed. An action can manipulate packet fields or make a forwarding decision. While the structure of a MAT and the actions are defined in the data plane, the content of the MATs is populated by the control plane.

To facilitate line rate processing on hardware, the operations that can be applied during packet processing are limited. Resubmit is a mechanism to reuse the limited resources of a pipeline. A resubmit can be triggered in the ingress control block during pipeline processing. When a resubmit is triggered, a copy of the original packet is sent through ingress processing again. A resubmit requires no additional bandwidth and adds no processing delay as the packet is not enqueued twice. A disadvantage of resubmitting is that the packet copy does not contain any changes to the packet headers that were applied during the first pipeline processing. On the Intel Tofino™, up to eight bytes with metadata can be prepended to a resubmitted packet. A packet can only be resubmitted once [45]. The recirculation mechanism is another mechanism that allows to reuse pipeline resources. In a recirculation, packets are looped back to the ingress port after processing. However, recirculation comes at the cost of bandwidth and processing delay which is why we do not consider it in this work.

Externs extend the functionality of P4 with target-specific functions. Examples of externs are counters, registers, meters, and digest messages. Registers enable stateful processing of packets. Values stored in a register are persistent, i.e., they are kept after the packet exits the pipeline. Meters enable traffic shaping based on a configured rate. Digest messages are Intel Tofino™-specific externs that allow small user-defined packets to be sent to the control plane.

More information on P4 can be found in a survey by Hauser *et al.* [47].

VI. P4 Implementation of the MNA Framework

This section describes the P4-MNA implementation of the MNA framework on the hardware-based Intel Tofino™ 2 switching ASIC using in-stack data (ISD). First, the general architecture of the P4-MNA implementation is described. Then, we describe how the complex header encoding of the MNA framework is parsed in P4. Next, we explain the processing of network actions in the P4-MNA pipeline. Finally, we describe the implementation of example network actions for performance measurement using AMM and for network slicing.

A. General Architecture of the P4-MNA Implementation

A node running the P4-MNA implementation supports basic MPLS functionality, i.e., MPLS label pushing, popping, and swapping. In addition, P4-MNA nodes parse and process network actions with the MNA encoding according to the MPLS stack structure described in Section II-B3. The implementation uses placeholders for the bSPL value of the NAS indicator and opcodes as they are yet to be assigned by IANA. For simplicity, the implementation comprises HBH-scoped NAS and select-scoped NAS, but not the I2E-scoped NAS. However, the I2E-scoped NAS can be added using the same mechanisms described in the following.

The source code, including the data plane program in P4, and the control plane program in Rust leveraging the rbfrt library [48] is publicly available on GitHub [49]. Further, we provide a Wireshark dissector for visualizing MNA traffic, and a Python library to build MPLS stacks with MNA LSEs in the GitHub repository.

B. Parsing of the MPLS Stack

The header encoding introduced with the MNA framework in Section II-B brings a lot of complexity to parsing because of different encodings and a dynamic structure. In P4, the parser is implemented as a simple finite-state machine. On hardware targets such as the Intel Tofino™ 2, the parser is limited in its number of states and transitions due to hardware constraints. A dynamic structure of different encodings in a header stack increases complexity by the number of states and transitions required for parsing. The implementation of P4-MNA therefore facilitates the parsing of network actions by parsing all Format D LSEs, i.e., AD LSEs, into the encoding of a Format C LSE based on the length from the NASL field. This reduces the complexity of parsing the MNA stack while preserving most of the semantics for different encodings. The semantics between Format C and Format D LSEs are restored later in the control block using MATs. This is further explained in Section VI-C.

An example NAS containing a Format B and multiple Format C and D LSEs with its parsed internal representation is shown in Figure 16.

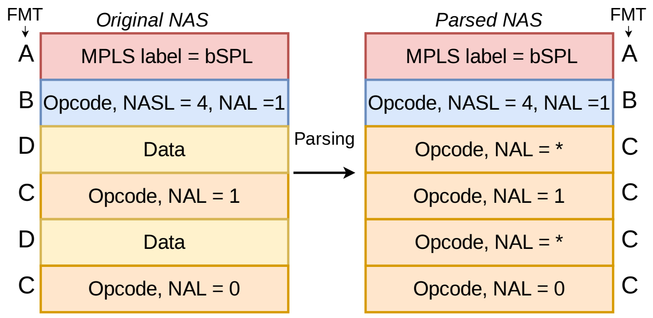
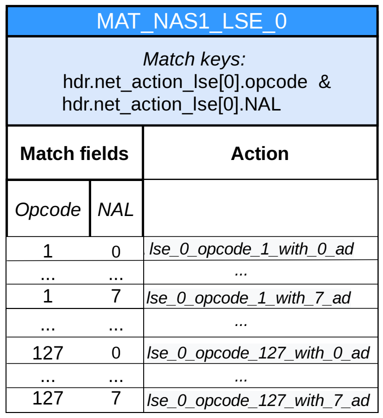


FIGURE 16. All Format D LSEs are parsed in the encoding of a Format C LSE to simplify the parser. The different encodings are later distinguished in the control blocks to restore the semantics of the Format D LSEs.

In the example, the length of the NAS is indicated by the value 4 in the NASL field in the Format B LSE. In P4-MNA, the example NAS is parsed into one Format A and B LSE, and four Format C LSEs according to the NASL field. The NAL fields, i.e., the number of AD LSEs per network action, are not relevant during parsing in P4-MNA.

C. The P4-MNA Processing Pipeline

The processing pipeline of P4-MNA executes network actions through the application of MATs. For each of the 16 LSEs in a NAS, one MAT exists in the pipeline. The MATs share the same structure illustrated in Figure 17 but match on different indices in the array of parsed LSEs.



MAT_NAS1_LSE_0

Match keys:
hdr.net_action_lse[0].opcode &
hdr.net_action_lse[0].NAL

Match fields		**Action**
Opcode	*NAL*	
1	0	*lse_0_opcode_1_with_0_ad*
...	...	...
1	7	*lse_0_opcode_1_with_7_ad*
...	...	...
127	0	*lse_0_opcode_127_with_0_ad*
...	...	...
127	7	*lse_0_opcode_127_with_7_ad*

FIGURE 17. A MAT for a network action matches on the opcode and the NAL field.

In Figure 17, a MAT that executes network actions located in the first LSE, i.e., at index 0, of a NAS is shown. Such a MAT matches on the opcode and the NAL field of the Format B or Format C LSE. The MAT has an action list with up to eight actions per opcode. Only one of those actions is matched and applied per packet. Each action accesses a different number of LSEs that follow the network action, i.e., AD LSEs. The number of accessed LSEs depends on the NAL field in the matched network action.

The ingress control block of the P4-MNA pipeline applies the MATs to a NAS. The processing pipeline of P4-MNA is shown in Figure 18.

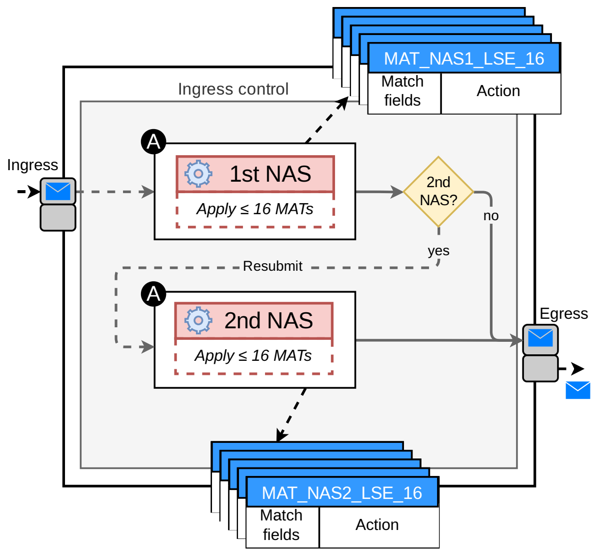


FIGURE 18. The pipeline of P4-MNA can process up to 16 network actions in a single pipeline iteration. If more network actions are present, the packet is resubmitted.

The P4-MNA pipeline can hold 16 MATs per NAS to match network actions. Therefore, if the packet contains multiple NAS, it is resubmitted as only 16 of those MATs fit into one pipeline iteration. The processing of a NAS is referred to as Ⓐ in Figure 18 and its operating principle is expanded in Figure 19.

During parsing, all Format D LSEs were parsed as Format C LSEs to facilitate the parsing of the complex MNA header structure as described in Section VI-B. However, by doing this, the knowledge about the semantics of an LSE is lost, i.e., whether the parsed LSE is an AD LSE, or a network action LSE. To restore this information, a matched action accesses a number of LSEs that follow the network action according to the NAL field and interprets those as AD LSEs. This mechanism is described in Figure 19 and expands block Ⓐ from Figure 18.

In Figure 19, the control flow for the application of network action MATs is shown with an exemplary NAS. The example NAS corresponds to the parsed example from Figure 16, i.e., all Format D LSEs are parsed as Format C LSEs. In step ❶, the first MAT is applied and matches on the first network action in the NAS. This network action has the opcode X and one AD LSE, i.e., the NAL field has the value 1. Therefore, the corresponding action with opcode X and one AD LSE is executed in step ❷. The actual logic of the network action is applied in this step. The action accesses the LSE indices 0 and 1, i.e., the network action LSE itself, and the succeeding AD LSE. In the executed action, the first

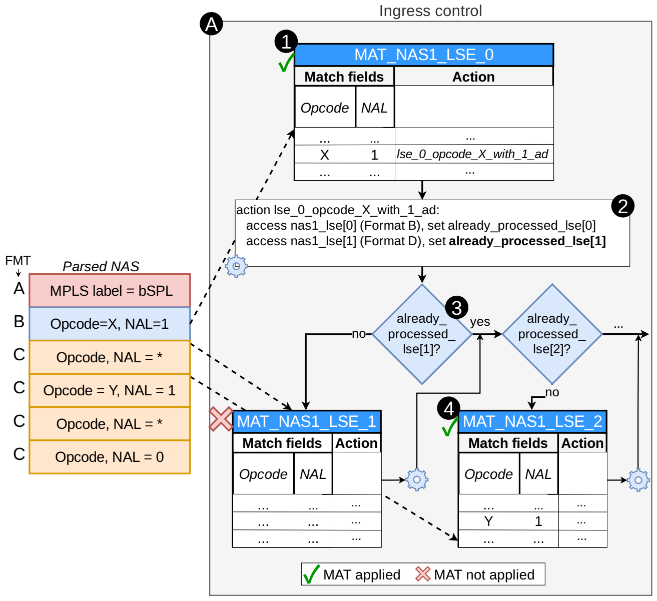


FIGURE 19. A network action MAT is applied to the first LSE of the parsed NAS. An entry is matched according to the opcode, and the NAL field. All accessed LSEs are marked as `already_processed` and are not considered further in processing.

LSE index is treated as Format B LSE[1]. All other accessed LSEs are treated as Format D LSEs. Therefore, the semantics between the Format B/C and Format D encoding that were lost during parsing are restored in this step. Further, in step ❷, the action marks both LSEs as `already_processed` in the metadata. Next, in step ❸, the control block checks if the succeeding LSE was marked as `already_processed` by a previous network action. As the LSE at index 1 in the example stack in Figure 19 was already accessed by the previous network action, the MAT for this LSE is not applied. Consequently, the next LSE index, i.e., index 2, is checked whether it is marked as `already_processed`. As this is not the case, the MAT for LSE index 2 is applied in step ❹ and the action with opcode Y is executed. This process repeats for all 16 MATs in the pipeline.

D. Implemented Network Actions

In this section we describe two implemented network actions, namely the performance measurement using the alternate-marking method (AMM) and bandwidth reservation with network slicing.

1) Network Action for Performance Measurement Using AMM

As a first example network action, we implement link-specific packet loss measurement using the alternate-marking method (AMM) as outlined in Section III-D. This implementation allows LSRs to track and report packet counters per flow enabling link-specific performance measurement.

[1]The first MAT matches on the Format B LSE, all other MATs match on Format C LSEs.

To support AMM, a P4-MNA LSR maintains two registers for each flow which count packets for the two colors used in AMM. When an MPLS packet carrying a NAS with the AMM network action is received, the LSR extracts the flow ID from the packet header and increments the corresponding color counter in the register. If the color of two consecutive packets of the same flow differs, indicating a color change event, the LSR generates a digest message containing the packet counter value, the flow id, and a timestamp. This digest message is sent to the control plane which correlates the data received from all LSRs in the network. By collating the timestamps included in the digest messages, the control plane computes the packet loss for each flow per hop. This network action is evaluated in Section VII-C2.

Time synchronization between LSRs is critical to ensure accurate correlation of the counters. For practical deployments, external synchronization protocols such as the Network Time Protocol (NTP) or the Precision Time Protocol (PTP) are necessary to maintain consistent timing across distributed nodes.

2) Network Action for Network Slicing

As a second example network action, we implement bandwidth reservation for network slices as outlined in Section III-E. The purpose of the MNA framework in network slicing is to carry the network resource partition (NRP) selector in each packet. The NRP selector identifies the network slice a packet belongs to. The LSR performs traffic shaping, e.g., bandwidth reservation, based on orchestrated NRPs. The orchestration of NRPs is handled by a management and orchestration (MANO) framework and is considered out of scope for this work.

On processing the network slicing action in a P4-MNA LSR, the NRP selector is extracted from the packet. Next, the packet is matched in an additional MAT that performs bandwidth metering using the meter extern of the Intel Tofino™ based on the extracted NRP selector. The reserved bandwidth per NRP is configured by the control plane. The bandwidth is metered according to the configured bandwidth reservation per network slice, i.e., traffic that exceeds the configured rate is dropped. Other traffic, i.e., traffic that is not part of an NRP is metered with the remaining available bandwidth. This way, the bandwidth reservation for every NRP is enforced through the NRP selector indication in the MNA network action. This network action is evaluated in Section VII-D2.

VII. Evaluation

In this section, we first analyze the scalability of P4-MNA in terms of the number of implementable network actions and the complexity of the network actions. We then evaluate the impact of the number of network actions on the processing delay and forwarding rate, showing the ability of P4-MNA to process network actions with the MNA encoding. Finally, we evaluate the two implemented use cases: link-specific packet loss measurement using AMM and bandwidth reservation using network slicing. The use case evaluations demonstrate the ability of P4-MNA to implement custom network actions using the available resources on the Intel Tofino™ 2 ASIC.

A. Scalability Analysis of P4-MNA

The IETF MNA encoding allows 128 different opcodes. The P4-MNA pipeline described in Section VI-C requires up to eight entries in a MAT and up to eight actions for one opcode, i.e., opcode A with 0 AD LSE, opcode A with 1 AD LSEs and so on. In total, $128 \cdot 8 = 1024$ entries are needed to cover all combinations of opcodes with AD LSEs in one MAT. A network action MAT in P4-MNA has a size of 4096 table entries and can therefore hold all possible combinations. However, when implementing a network action, additional resources such as registers for the AMM network action, and meters for the network slicing action must be considered.

The complexity of network actions that can be implemented is limited to the functionality of the Intel Tofino™ 2 ASIC and the P4 language. Simple network actions such as the network slicing and the AMM network action are implementable. However, more sophisticated features, such as cryptographic functions, are not supported on the Intel Tofino™ 2. More hardware, e.g., a cryptographic extern offloaded to an external server or a smartNIC, is required to implement such functionality.

More complex network actions may require more instructions that may not fit in the pipeline. In this case, recirculation is required. In a recirculation, a packet is looped back to an ingress port where it is processed again, i.e., the required instructions are applied. An advantage of this is the implementation of more complex functions such as shown by [50]. However, recirculation comes at the cost of bandwidth. To avoid this, simple network actions should be preferred. The network actions currently proposed by the WG are simple enough to not require recirculation. However, a combination of many simple network actions may also exceed the packet processing resources within a single packet processing cycle and require recirculation.

B. Impact of Number of Network Actions

In this section, we evaluate the impact of the number of network actions on the processing delay and forwarding rate. First, we describe the testbed and methodology. Then, we present our results.

1) Testbed and Methodology

The P4-MNA implementation operates at line rate of $400\,\mathrm{Gb/s}$ per port. We evaluate network action processing on a path of three LSRs emulated by ingress ports on an Intel Tofino™ 2 switch. The ingress and egress LERs are emulated by a second Intel Tofino™ 2 switch running the

P4TG traffic generator [51]. We extended P4TG to enable 400 Gb/s MNA traffic generation [52]. The testbed setup for this evaluation is shown in Figure 20.

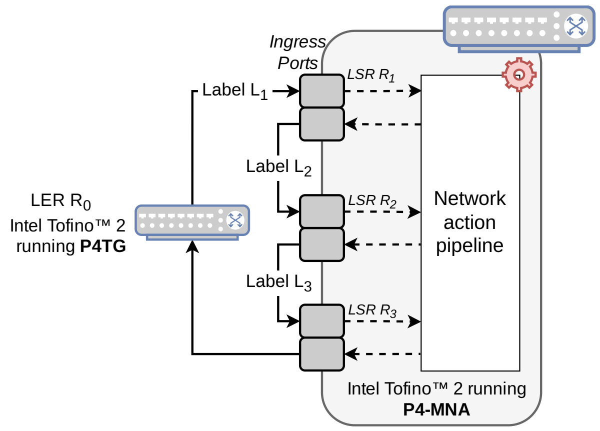


FIGURE 20. The testbed consists of three LSRs represented as ingress ports on the P4-MNA switch. Packets containing the network actions are sent from P4TG to an ingress port of the switch. Here, network actions are processed and the packets are sent to another ingress port.

In Figure 20, packets are sent to an ingress port of the Intel Tofino™ 2 switch running the P4-MNA implementation. Each ingress port of the P4-MNA switch represents an LSR processing network actions. After processing the network actions in the stack, LSRs pop the top-of-stack label and forward packets accordingly. For label L_1, L_2, and L_3 packets are sent to a different ingress port on the same Intel Tofino™ 2 switch. LSR R_3 forwards packets back to the traffic generator P4TG for traffic analysis and measurement. In the testbed setup, all LSRs operate on the same physical device and share a common time reference.

For all experiments, constant bit rate (CBR) traffic is generated for approximately 60 seconds by P4TG. The smallest frame size the traffic generator P4TG can generate to achieve 400 Gb/s is 128 B. Wide-area networks are dominated by Ethernet traffic which typically uses a default MTU size of 1500 B [53]. Therefore, we tested two packet sizes: 128 B and 1500 B. For both tested packet sizes, the results were nearly identical with only a marginal reduction in latency for 128 B packets, i.e., in the order of nanoseconds, resulting from the smaller packet size. For clarity, we focus on the 1500 B results as larger packet sizes are more representative of typical high-throughput network traffic. Each experiment in the evaluation is repeated ten times and confidence intervals with a confidence level of 99 % are calculated.

2) Performance Results

In this evaluation, we test four different MPLS stacks and measure the RTT and packet loss to measure their impact on the processing delay and forwarding rate. Figure 21 shows the MPLS stacks used in the experiments containing network actions and forwarding labels to reach the nodes R_1, R_2, and R_3 in a SR-MPLS fashion.

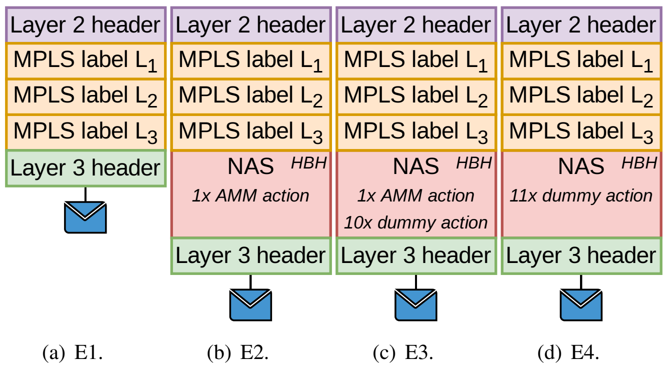


FIGURE 21. The MPLS stacks used in the evaluation for experiments E1 – E4.

In the first experiment *E1*, the MPLS stack contains only forwarding labels, i.e., labels L_1, L_2, and L_3 to validate the MPLS forwarding capability. In the second experiment *E2*, we add the AMM network action from Section III-D in an HBH-scoped NAS to the MPLS stack. In the third experiment *E3*, we add ten additional network actions to the HBH-scoped NAS to see if multiple network actions have an impact on the processing delay. Ten additional network actions is the maximum the traffic generator P4TG supports. These are dummy network actions that write arbitrary data into their AD fields upon processing. The dummy actions constitute a stress test to evaluate the system's ability to process multiple network actions. In the fourth experiment *E4*, the MPLS stack contains an HBH-scoped NAS with eleven dummy network actions. In this experiment, the AMM network action is not contained to see if the processing of the more sophisticated AMM network action impacts processing delay.

This evaluation has two goals. First, we want to verify if P4-MNA is capable of achieving line rate processing of network actions. Second, we analyze the impact on the processing delay resulting from network actions in the MPLS stack. We measure the packet loss PL and the RTT of the returned traffic using the measurement capabilities of the traffic generator P4TG. The mean results of the RTT and packet loss measurement are shown in Table 3. For clarity, we omit confidence intervals as their width was less than 0.004 % of the measured average for a confidence level of 99 %.

TABLE 3. Measured packet loss $\overline{PL}$ and $\overline{RTT}$ for the experiments E1 – E4.

	E1	**E2**	**E3**	**E4**
$\overline{PL}$	0	0	0	0
$\overline{RTT}$	4.75 µs	4.79 µs	4.79 µs	4.79 µs

Table 3 shows that no packet loss was recorded, i.e., P4-MNA achieves line rate forwarding of 400 Gb/s with MPLS traffic in the experiments E1 – E4. Adding network actions increases the RTT marginally by ≈ 40 ns from 4.75 µs in E1 to $4.79\,\mu s$ in E2 – E4. A packet is processed once by each LSR, i.e., three times in total[2]. The latency added by one LSR in E2 – E4 is therefore $\frac{40\ ns}{3} = 13.3\ ns$.

The increased transmission delay resulting from the larger frame size due to additional network actions is negligible, i.e. less than $1\,\text{ns}$. According to a latency profiling study of the Tofino™ P4 programmable ASIC-based hardware by Franco *et al.*, the number of applied tables and the number of parser states increase the latency in the order of nanoseconds [54]. The number of applied tables differs in E2 and E3 because more network actions exist in the MPLS stack, but both experiments result in the same latency. Therefore, the number of applied tables has no measurable impact on the processing delay in P4-MNA. In E2 and E3, the AMM network action is applied, which performs more sophisticated operations compared to the dummy actions such as accessing registers. However, there is no increase in RTT compared to E4 where only dummy network actions are applied. Therefore, the complexity of network actions does not affect the processing delay in P4-MNA. The difference between E1 and E2 – E4 is the additional parsing state that extracts network actions. We therefore attribute the increase in RTT to the additional parsing state. However, the observed RTT increase of $13.3\,\text{ns}$ per hop is negligible.

Although the current evaluation is limited to eleven network actions due to traffic generation constraints, the P4-MNA implementation supports up to 32 network actions. In future work, we plan to extend the P4TG traffic generator to generate larger MPLS stacks to further evaluate the P4-MNA implementation.

C. Use Case 1: Alternate Marking Method (AMM)

In this section we describe the evaluation of the first implemented use case, i.e., link-specific packet loss measurement using the alternate marking method (AMM). For this evaluation, the testbed and methodology described in Section VII-B1 are used. First, we describe the configuration and validation of end-to-end packet loss in the testbed. Then, we use the AMM network action to calculate link-specific packet loss.

1) Validation of End-to-End Packet Loss

In experiment *E5*, the LSRs R_1, R_2, and R_3 are configured to drop packets probabilistically to emulate packet loss on a link. In this evaluation, we verify the configured end-to-end packet loss. This experiment serves as a baseline for evaluating the link-specific packet loss measurement using the implemented AMM network action in experiment E6. The probability $p^{drop}_{i,i+1}$ denotes the packet drop probability on the link from node R_i to R_{i+1}. We configure the probabilities $p^{drop}_{0,1} = 0.1$, $p^{drop}_{1,2} = 0.2$, and $p^{drop}_{2,3} = 0.3$. The packets are dropped according to the configured probability in the ingress of node R_{i+1} before any packet processing is applied. The end-to-end packet loss is measured at the traffic generator P4TG.

For experiment E5, i.e., with probabilistic packet drop, the measured packet loss at P4TG is at $49.6\,\%$. The expected end-to-end drop probability $p^{drop}_{0,3}$ of a frame being dropped by one of the LSRs is given in Equation 1.

$$p^{drop}_{0,3} = 1 - \prod_{i\in\{0,1,2\}} (1 - p^{drop}_{i,i+1}) \tag{1}$$

For the configured drop probabilities the expected end-to-end drop probability of a frame is $p^{drop}_{0,3} = 0.496$. The measured packet loss of $49.6\,\%$ matches exactly with the expected probability, validating the configuration.

2) Validation of Link-Specific Packet Loss

In experiment *E6*, we evaluate the link-specific packet loss measurement of the implemented AMM network action. The LSRs are configured to drop packets on ingress according to experiment E5 in Section VII-C1. We calculate the link-specific packet loss resulting from the reported packet counters of the AMM network action and compare them to the configured packet loss. In this experiment, the color in the generated AMM network action is alternated ten times approximately every five seconds at a rate of $400\,\text{Gb/s}$. Every LSR exports its packet counter value to the control plane when the color changes. For evaluation, the control plane uses the last counter value received to calculate the packet loss of each link.

For a link from an LSR R_i to its successor LSR R_{i+1}, the link-specific packet loss $\Delta_{i,i+1}$ is calculated in the control plane based on the reported counter values triggered by the AMM network action. The value N^a_i denotes the counter value of LSR R_i and color a, i.e., the number of received packets of color a at LSR R_i. Equation 2 calculates the link-specific packet loss between LSR R_i and R_{i+1} based on both color counter values. This equation can be further simplified to Equation 4.

$$\begin{aligned}\Delta_{i,i+1} &= (N^a_i - N^a_{i+1}) + (N^b_i - N^b_{i+1}) &(2)\\ &= (N^a_i + N^b_i) - (N^a_{i+1} + N^b_{i+1}) &(3)\\ &= N^{total}_i - N^{total}_{i+1} &(4)\end{aligned}$$

Equation 4 measures the link-specific packet loss $\Delta_{i,i+1}$ by correlating the counter values of two consecutive LSRs R_i and R_{i+1} regardless of packet color. In the evaluation, we can use Equation 4 to calculate the link-specific packet loss. In general, however, both colors are required to have a synchronized export trigger as described in Section III-D.

In the evaluation testbed, LSR R_1 does not have a preceding LSR. To calculate the packet loss $\Delta_{0,1}$ on the link from the traffic generator to LSR R_1, the total number of sent packets is known to the control plane.

The formula for the calculated link-specific packet loss rate $\hat{p}^{drop}_{i,i+1}$ on a link from LSR R_i to R_{i+1} is shown in Equation 5.

[2]The per-packet processing delay is constant on the Intel Tofino™.

$$\hat{p}^{drop}_{i,i+1} = \frac{\Delta_{i,i+1}}{N^{total}_{i}} \quad (5)$$

The mean results from experiment E6 are shown in Table 4. The width of the confidence intervals was less than $0.005\,\%$ of the measured average and therefore, they are not shown.

TABLE 4. Mean counter values and calculated packet loss per link with the AMM network action.

	Generated packets	**LSR R_1**	**LSR R_2**	**LSR R_3**
#Received packets of color *a* N^a_{i+1}		860879414	688688859	482063758
#Received packets of color *b* N^b_{i+1}		860953198	688751879	482136175
N^{total}_{i+1}	1913168832	1721832612	1377440738	964199933
Calc. loss with AMM $\Delta_{i,i+1}$		191336220	344391874	413240805
Calc. loss rate with AMM $\hat{p}^{drop}_{i,i+1}$		0.1000	0.2000	0.3000
Conf. loss $p^{drop}_{i,i+1}$		0.1	0.2	0.3

Table 4 shows that the calculated link-specific packet loss rate $\hat{p}^{drop}_{i,i+1}$ matches the configured drop probabilities $p^{drop}_{i,i+1}$ for each LSR. For LSR R_1, $10\,\%$, for LSR R_2 $20\,\%$, and for LSR R_3 $30\,\%$ packet loss was measured. This matches the configured drop probability at each LSR exactly. Thus, the implemented AMM network action is capable of precisely measuring link-specific packet loss.

D. Use Case 2: Network Slicing

In this section we describe the evaluation of the second implemented use case, i.e., bandwidth reservation with network slicing. We apply the same methodology as explained in Section VII-B1 but use a different testbed. First, we describe the testbed, and then, present our results.

1) Testbed

In experiment $E7$, we evaluate the implemented network slicing action for bandwidth reservation described in Section VI-D2. We use the testbed as shown in Figure 22.

The testbed in Figure 22 contains two Intel Tofino™ 2 switches that are connected with a physical $400\,\mathrm{Gb/s}$ and a $100\,\mathrm{Gb/s}$ link. The first Tofino™ runs the P4TG traffic

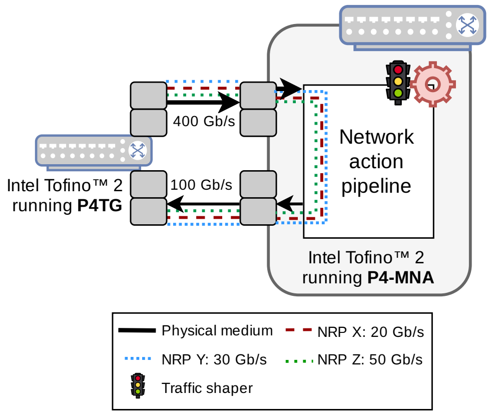


FIGURE 22. Three NRPs are configured with bandwidth reservations. They are generated by P4TG and are sent via a 400 Gb/s link to the P4-MNA switch. The traffic is returned via a 100 Gb/s bottleneck link to P4TG for measurement.

generator and the second runs the P4-MNA implementation. In the testbed, three NRPs are configured. NRP X has a bandwidth reservation of $20\,\mathrm{Gb/s}$, NRP Y of $30\,\mathrm{Gb/s}$, and NRP Z of $50\,\mathrm{Gb/s}$. P4TG generates three streams containing a network action with the NRP selectors X, Y, and Z. They are generated with the respective reserved bandwidth and are sent via the $400\,\mathrm{Gb/s}$ link to the P4-MNA switch. The P4-MNA switch performs traffic shaping according to the configured bandwidth reservations of the NRPs and the extracted NRP selector from the network action. For that purpose, the control plane configures the meter extern in the data plane with the reserved bandwidth for each NRP. Then, the switch returns the traffic via the $100\,\mathrm{Gb/s}$ link to P4TG for measurement. The $100\,\mathrm{Gb/s}$ link creates a bottleneck in the network that causes congestion in the switch if the capacity of the link is exceeded leading to packet loss. We add a fourth stream generated on the $400\,\mathrm{Gb/s}$ link that exceeds the bottleneck link's capacity to force congestion on the egress link. This stream is interference traffic and does not contain an NRP selector. Therefore, no bandwidth is reserved for the interference traffic stream. In total, $200\,\mathrm{Gb/s}$ of traffic is generated.

2) Validation of Bandwidth Reservations

First, we measure the packet loss per stream and the RTT without generating interference traffic, i.e., no congestion is caused on the link. Next, we add interference traffic but do not apply the network slicing action, leading to congestion. Finally, we apply the network slicing action. The mean measured packet loss per stream for $E7$ is shown in Figure 23. The width of the confidence intervals was less than $0.05\,\%$ of the measured average and therefore invisible.

In Figure 23, no packet loss is visible if the link is not overloaded. This is the baseline. If the interference traffic is added and the NRPs are not enforced by the switch, all

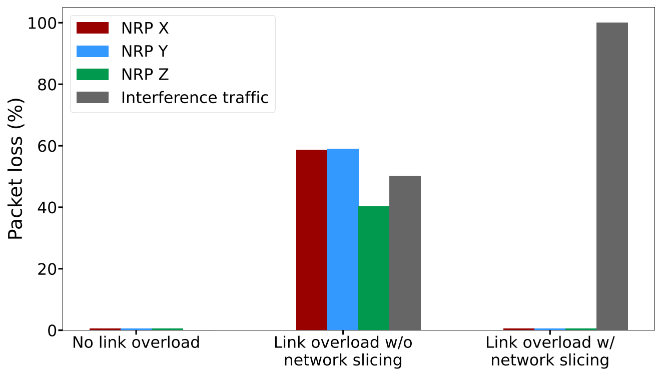


FIGURE 23. **Mean packet loss measured by P4TG with and without traffic shaping according to NRPs.**

streams see a high packet loss between 40 % and 60 %[3]. Therefore, the reserved bandwidth for the NRPs is not guaranteed. If the NRPs are enforced by the switch, the interference traffic is dropped. As a result, the egress link is not overloaded and the NRP streams see no packet loss. Their reserved bandwidth is guaranteed.

This evaluation has shown that the MNA network slicing action can indicate NRPs in the MPLS stack to enforce network resource reservations. The focus in this work lies on the indication of the NRP selector in a network action. For practical use, a management and orchestrator (MANO) framework is required. More information about network slicing with bandwidth guarantees in P4 switches can be found in [55].

VIII. Constraints for Header Stacking in MNA

In this section, we explain two constraints that apply to header stacking in MNA. First, we describe a constraint resulting from the available resources (RLD) in hardware. Then, based on our implementation, we describe a constraint arising from the maximum NAS sizes that need to be supported in the proposed MNA framework. As the latter constraint makes LSRs with small RLD not usable for MNA, we suggest that LSRs may support lower NAS sizes and to signal these values in addition to the RLD so that hardware with lower processing capabilities can be integrated into MNA domains.

A. Constraints for Header Stacking due to Readable Label Depth (RLD)

In general, hardware can only parse limited bytes of a packet header. This limitation results from the hardware resources available on the device such as memory. Additionally, the complexity of a packet parser is limited to facilitate packet processing within a single packet cycle. Header sizes that exceed the available resources for parsing cannot be processed in a single packet cycle and, therefore, are an obstacle for packet forwarding at line rate. Thus, for practical deployment, the relevant information in the header must fit within the readable bytes. In MNA, this translates to the so-called readable label depth (RLD) [8] which indicates the number of LSEs a node can parse.

The RLD is a critical parameter of LSRs when deploying MNA in MPLS networks. In the current proposal, LSRs signal their RLD to the ingress LERs using a routing protocol such as IS-IS [56] or OSPF [57]. The ingress LER must ensure that a NAS determined for a node is within the RLD when the packets reach that node. This is done based on the signaled RLD parameter. An HBH-scoped NAS can be located at the bottom of stack. However, if this HBH-scoped NAS is not within the RLD of an intermediate node, the ingress LER places copies of that NAS within the RLD to ensure that the NAS can be found by that node. Only the topmost HBH NAS copy is processed by a node. An example is given in Figure 24 and further described below.

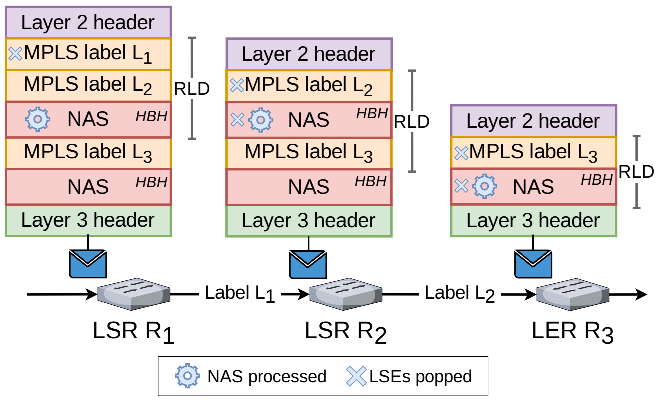


FIGURE 24. **An example MPLS label stack with one HBH-scoped NAS in the RLD for each node. As the RLD of LSR R_1 and R_2 is limited to 3 LSEs, a copy of the HBH-scoped NAS is placed within the RLD of those LSRs.**

In Figure 24, the nodes have an RLD of three LSEs[4]. Placing a single HBH-scoped NAS at the bottom-of-stack is therefore not sufficient because it is not within the RLD of LSR R_1. The ingress LER must place the HBH-scoped NAS below the forwarding label L_2 and a copy of the HBH-scoped NAS below forwarding label L_3 to ensure that all nodes can process the NAS in their RLD. R_1 parses all LSEs in its RLD and processes the HBH-scoped NAS found below the forwarding label L_2. R_2 processes the HBH-scoped NAS found below the forwarding label L_2 and pops that NAS after processing. Finally, R_3 processes the copy of the HBH-scoped NAS found below the forwarding label L_3.

[3]We observe different drop probabilities for the different flows although they travel jointly over the same link. The reason for this phenomenon is that the traffic is strictly CBR. Therefore, we see periodic arrival behavior at the bottleneck link and packets of some flows get dropped more often than packets of other flows within a period due to combinatorial effects.

[4]For simplicity in the example, we treat a NAS as a single LSE. In reality, each NAS may contain up to 17 LSEs.

B. Constraints for Header Stacking due to Maximum NAS Sizes

Both select-scoped NAS and HBH-scoped NAS can contain up to 17 LSEs. Since a node may need to process both a select-scoped NAS and an HBH-scoped NAS, it requires two arrays of maximum NAS size to parse both NAS. We denote these sizes as $maxLSEs_{NAS}^{select}$ and $maxLSEs_{NAS}^{HBH}$. Since the HBH-scoped NAS may not immediately follow the select-scoped NAS but be located deeper in the stack, the LSEs in between are parsed into a third array which we call the in-between stack and which is $maxLSEs_{stack}^{btwn}$ LSEs large. Thus, the RLD comprises one forwarding label, a maximum select-scoped NAS, the in-between stack, and a maximum HBH-scoped NAS. As $maxLSEs_{NAS}^{select}$ and $maxLSEs_{NAS}^{HBH}$ are 17 LSEs according to the proposed standard [14], and the RLD is given by the hardware, the size of the in-between stack can be computed by Equation 6.

$$\begin{aligned}\text{maxLSEs}_{stack}^{btwn} = RLD - \text{maxLSEs}_{NAS}^{select}\\ - \text{maxLSEs}_{NAS}^{HBH} - 1.\end{aligned} \tag{6}$$

This holds under the assumption that the hardware implementation cannot share the memory for parsing the select-scoped NAS, the HBH-scoped NAS, and the in-between stack, which is the case for P4-MNA. The relation between the three arrays and the RLD is depicted on the left side in Figure 25. In case of P4-MNA, $RLD = 51$, therefore, the in-between stack can contain up to 16 LSEs. They contain any LSE between the two NAS, i.e., forwarding labels, special purpose labels, and network actions. This seems a low value compared to a maximum NAS size. Moreover, we conclude from Equation 6 that MNA-capable hardware must have an RLD of at least 35 LSEs, which may be problematic for some hardware platforms.

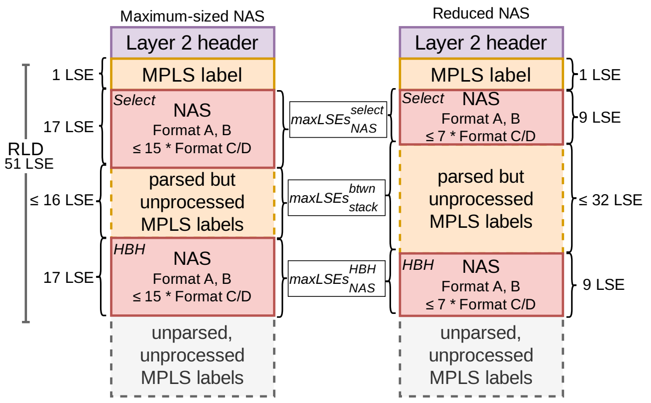

Maximum-sized NAS
Reduced NAS
Layer 2 header
MPLS label
1 LSE
17 LSE
RLD 51 LSE
≤ 16 LSE
Select NAS Format A, B ≤ 15 * Format C/D
parsed but unprocessed MPLS labels
HBH NAS Format A, B ≤ 15 * Format C/D
unparsed, unprocessed MPLS labels
maxLSEs select NAS
maxLSEs btwn stack
maxLSEs HBH NAS
Select NAS Format A, B ≤ 7 * Format C/D
HBH NAS Format A, B ≤ 7 * Format C/D
9 LSE
≤ 32 LSE

FIGURE 25. **The maximum size of a NAS in the IETF proposal is $maxLSEs_{NAS}^{select} = maxLSEs_{NAS}^{HBH} = 17$. P4-MNA has a RLD of 51 LSEs. This leaves 17 LSEs for other purposes. If the parameters $maxLSEs_{NAS}^{select}$ and $maxLSEs_{NAS}^{HBH}$ are reduced, more LSEs are available for other purposes.**

C. Signaling Maximum NAS Sizes

The problems observed with a small in-between stack or a minimum RLD of 34 LSEs for MNA-capable hardware are due to the fact that a NAS can be up to 17 LSEs in size. However, many network actions, or even a combination of many actions, do not require 17 LSEs in a NAS. We now propose that a NAS can be up to 17 LSEs in size, but nodes can also support smaller NAS. To avoid receiving packets with larger NAS, we suggest that the node-specific maximum NAS sizes $maxLSEs_{NAS}^{select}$ and $maxLSEs_{NAS}^{HBH}$ are signaled to all ingress LERs, just like the RLD. This allows hardware with a lower RLD to be used for MNA.

An example is given on the right side of Figure 25. We choose $maxLSEs_{NAS}^{select} = maxLSEs_{NAS}^{HBH} = 9$ for P4-MNA. With an RLD of 51 LSEs this leaves $maxLSEs_{stack}^{btwn} = 32$ LSEs for the in-between stack according to Equation 6. Moreover, there is no longer a minimum RLD for MNA-capable nodes as the maximum NAS sizes can be adjusted to fit within the RLD of the node. However, in turn, the supported NAS sizes may be too small for some applications.

IX. Challenges with Mutable Data in an ISD Implementation

In this section, we explain the concept of mutable data and describe a challenge when using mutable data with in-stack data (ISD).

Mutable data in MNA refers to data in which the value in one of the data fields of a NAS changes during packet forwarding. An example of mutable data in MNA is the collection of telemetry data along a path in the context of IOAM in passport mode. Here, each node on the path adds information to the NAS, such as its node ID. This information must not be discarded during packet forwarding, i.e., the NAS containing this information must not be popped.

For backward compatibility, the range of use cases that can be implemented with ISD is limited because mutable data is constrained. This constraint results from the protocol design and not from the P4-MNA implementation. The first 20 bits, i.e., the MPLS label, are often hashed for ECMP load balancing and therefore must not be altered in transit. The number of LSEs used for hashing varies in implementations from one up to 16 MPLS labels [58]–[60]. Entropy labels [2] may be introduced to account for ECMP load balancing and to avoid the need to hash the MPLS stack. However, this does not provide full backward compatibility as legacy devices may not support entropy labels.

A NAS with the maximum possible number of mutable bits according to the MNA encoding is illustrated in Figure 26. The first 20 bits of an LSE are hashed for ECMP and must not be modified. The remaining 12 bits are partially occupied by length fields and other flags, such as the bottom of stack bit. Therefore, in a Format B LSE, zero bits, in a Format C LSE seven bits, and in a Format D LSE, eleven bits are mutable.

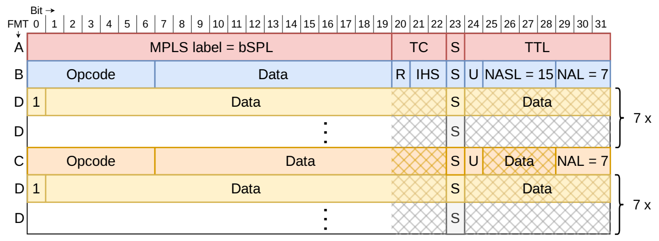

Bit
FMT
A MPLS label = bSPL TC S TTL
B Opcode Data R IHS S U NASL = 15 NAL = 7
D 1 Data S Data
D S
C Opcode Data S U Data NAL = 7
D 1 Data S Data
D S
7 x
7 x

FIGURE 26. **A NAS carrying the maximum number of mutable bits (hatched) consists of a Format B LSE, followed by seven Format D LSEs, one Format C LSE, and seven Format D LSEs.**

In Figure 26, a NAS with 424 data bits is shown. However, only 161 of those data bits are mutable. The overall size of the NAS is 544 bits (68 B). Therefore, only a fraction of the NAS, i.e., $\approx 30\%$, can be used for mutable data. Further, the number of AD LSEs for a single network action is limited to seven. Therefore, only 77 mutable bits are available to a network action in the Format B encoding, and 84 bits in the Format C encoding.

The passport method described for IOAM in Section III-B collects information from LSRs along a path, e.g., ingress timestamps. Here, each LSR adds a timestamp to the packet. Because collected timestamps are 32-bit wide [22], an in-stack network action can carry only two timestamp values as mutable ISD for an entire LSP.

If a network action requires a lot of mutable data, ISD is inefficient because it wastes bits that cannot be mutated. Alternatively, mutable data can be carried as post-stack data (PSD) where all bits are mutable [61]. With PSD, AD is placed after the bottom-of-stack. An implementation of the MNA framework leveraging PSD is out-of-scope for this work but will be explored in the future.

When using SR-MPLS, the top-of-stack forwarding label is popped per segment. Therefore, a NAS below the forwarding label is also popped. These NAS cannot contain mutable data because they do not reach the destination. When SR-MPLS is used with in-stack mutable data, the mutable data must be at the bottom of the stack to avoid them being popped. In this case, the RLD of each LSR must be large enough to parse the entire MPLS stack, i.e., the mutable data located in the NAS at the bottom of the stack.

X. Security Considerations

In this section we describe security risks and considerations arising from the use of the MNA framework. Currently, there are three security risks that are considered in the MNA drafts [8], [14]. The first is link-level security, the second is MNA information originating from other domains, and the third is the syntax and semantics of network actions.

A. Link-Level Security

The information contained in network actions is sensitive and may be exploited or manipulated for attacks. Thus, MNA traffic should be protected from eavesdropping and manipulation. However, the MPLS protocol does not have a built-in security mechanism. Link-level security mechanisms such as MACSec can be employed to prevent eavesdropping on traffic using confidentiality and authentication. Further, there is a draft for opportunistic encryption for hop-by-hop and end-to-end encryption in MPLS [62]. However, end-to-end encryption is not feasible with hop-by-hop MNA processing as nodes need to inspect the contents of the MPLS stack [8].

B. MNA Information Originating from Other MPLS Domains

An MPLS network must be protected from processing MPLS labels originating outside the network, e.g., label stacks forged by an attacker [8], [14]. LSRs must process only label stacks including network actions intended for them. This is trivial within a single domain, but challenging in a multi-domain context. We consider the example in Figure 27 where traffic from customer domain D_1 is tunneled through a provider domain D_2 into customer domain D_3. D_1 and D_3 are under the same administrative control while D_2 is not. LER R_1 pushes LSEs including network actions onto packets that are intended for processing by R_1 and R_4 but not for processing by R_2 and R_3.

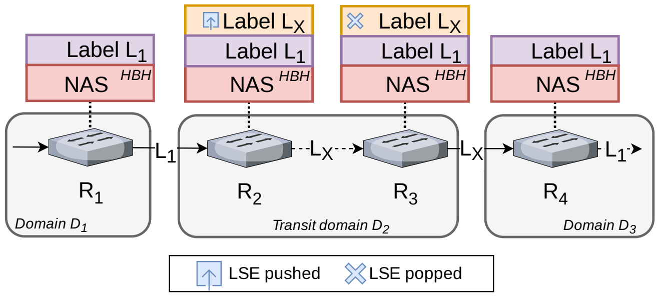


FIGURE 27. **Traffic between two domains under the same administrative control is tunneled through a provider domain under different administrative control.**

In the provider domain D_2, the LERs R_2 and R_3 must be protected from erroneously processing the HBH-scoped NAS which is intended only for LSRs R_1 and R_4. Several options are possible. First, if MNA is not supported in the provider domain, the network actions are not executed and can be forwarded through the provider domain. However, in this case, the provider domain cannot take advantage of MNA. Second, if the ingress LER has an RLD equal to or greater than all other LSRs in the provider domain, it can drop all incoming packets that contain network actions in reach for any LSR of D_2. However, this does not allow MNA traffic to pass through the provider domain. In a third solution, the ingress LER R_2 pushes an entire new MPLS stack including a bottom of stack bit so that all LSRs in the provider domain stop parsing the stack beyond that bit. This also prevents them from executing the HBH-scoped network actions because it isolates the MNA information into two separate domain-specific stacks [8].

The LSRs in the customer domain D_3 must be protected from executing network actions that may be inserted by an adversarial provider. As a simple solution, the ingress LER R_4 may drop any incoming traffic containing network actions. However, then network actions cannot be used between D_1 and D_3 which are under the same administrative control. To support MNA, the ingress LER R_4 may forward external packets from a set of non-critical network actions such as NFFRR and block critical network actions such as for network slicing. As a third solution the customer trusts its service provider and accepts all MNA traffic for execution in domain D_3.

C. Syntax and Semantics of Network Actions

Operators must ensure that ingress LERs enforce syntactically correct NAS. Otherwise, the MPLS stack will be corrupted resulting in packet loss. In addition, network actions are semantically unbounded, i.e., they are not restricted to a fixed feature set but can describe arbitrary operations. For example, a network action could be defined that unintentionally enables manipulating a node's memory to compromise forwarding or facilitate eavesdropping. The IETF must therefore ensure that only secure network actions are defined that do not compromise the security of the network. Some opcodes are reserved for locally defined network actions. In this case, implementers and network operators must ensure that these network actions are secure [14].

XI. Conclusion

In this paper, we gave an overview of the concepts and the header encoding in the MNA framework proposed by the IETF. We summarized the use cases identified by the working group and gave an outlook on how they can be implemented in the MNA framework. Network actions in the MNA framework are conceptually similar to IPv6 extension headers (EH). However, IPv6 EH are insufficiently supported on hardware platforms, mostly due to their flexible header encoding that allows for large payloads. Therefore, we verified the feasibility of the encoding introduced with MNA for hardware platforms.

This paper presents P4-MNA, the first implementation of the MNA framework. It shows that the proposed MNA framework is well implementable on programmable hardware such as the Intel Tofino™ 2 switching ASIC with a line speed of $400\,\mathrm{Gb/s}$ per port. We analyzed the scalability of P4-MNA regarding the number of processable LSEs, the number of implementable network actions, and the complexity of network actions. P4-MNA is capable of parsing and processing 51 LSEs containing up to 32 network actions at a line speed of $400\,\mathrm{Gb/s}$. The evaluation demonstrated that P4-MNA can efficiently process multiple network actions while maintaining line speed and minimal latency validating its suitability for high-speed networks. Further, we showed the capability of P4-MNA to implement custom network actions such as link-specific packet loss measurement using AMM and bandwidth reservation with network slicing. The source code is publicly available on GitHub [49].

We explained how the readable label depth (RLD) constrains header stacking in MNA and argued that header stacking is also constrained by maximum NAS sizes. Therefore, we proposed to make maximum NAS sizes a hardware parameter and signal them like the RLD. This facilitates the integration of hardware with smaller RLD into MNA-enabled domains. A detailed signaling proposal will be introduced in the future.

We explained that some bits of the MPLS header for in-stack data (ISD) must not be changed by intermediate hops for compatibility reasons so that only $30\,\%$ of the available bits can be used for mutable data. Therefore, the IETF has also proposed the post-stack data (PSD) approach where all available bits can be used for mutable data, but PSD is considered more complex than ISD. Future work will include an implementation of MNA with PSD and its analysis as well as a demonstration of use cases with on-path data collection.

Finally, we summarized the security risks currently being considered by the working group which include link-level security, MNA information originating from other MPLS domains, and the syntax and semantics of network actions.

Erratum

After publication in IEEE OJCOMS [63], it was identified that Figures 8(a), 8(b), and 27 were incorrectly rendered due to duplication and formatting issues introduced during production. An official Erratum has been published and is linked on the IEEE Xplore page [64]. The figures included in this version are correct.

List of Abbreviations

AD	ancillary data
AMM	alternate-marking method
bSPL	base Special Purpose Label
CBR	constant bit rate
ECMP	equal-cost multi-path
EH	Extension header
eSPL	extended Special Purpose Label
FRR	Fast Reroute
HBH	hop-by-hop
I2E	ingress-to-egress
ISD	in-stack data
LER	Label Edge Router
LSE	Label Stack Entry
LSP	Label Switched Path
LSR	Label Switching Router
MAT	match+action table
MNA	MPLS Network Actions
MPLS	Multiprotocol Label Switching
NAL	Network Action Length
NASL	Network Action Sub-stack Length
NAS	Network Action Sub-stack
NRP	network resource partition

NSH	network service header
OAM	operations, administration and maintenance
P4	Programming Protocol-independent Packet Processors
PSD	post-stack data
RLD	readable label depth
SFC	service function chaining
SID	segment identifier
SR	segment routing
TC	traffic class

REFERENCES

[1] E. C. Rosen, A. Viswanathan, and R. Callon, "RFC3031: Multiprotocol Label Switching Architecture," Jan. 2001.

[2] C. Filsfils, Ed., P. Francois, Ed., M. Shand, B. Decraene, J. Uttaro, N. Leyman, and M. Horneffer, "RFC6790: The Use of Entropy Labels in MPLS Forwarding," Nov. 2012.

[3] K. Kompella and W. Li, "No Further Fast Reroute," Internet-Draft draft-kompella-mpls-nffrr-04, IETF, Oct. 2023. Work in Progress.

[4] A. Farrel, S. Bryant, and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining." RFC 8595, June 2019.

[5] M. Haeberle, B. Steinert, M. Weiss, and M. Menth, "A Caching SFC Proxy Based on eBPF," in *International Conference on Network Softwarization (NetSoft)*, pp. 171–179, 2022.

[6] R. Gandhi, F. Brockners, B. Wen, B. Decraene, and H. Song, "MPLS Network Action for Transporting In Situ OAM Data Fields," Internet-Draft draft-gandhi-mpls-ioam-12, IETF, Mar. 2024. Work in Progress.

[7] P. Manzanares-Lopez, J. P. Muñoz-Gea, and J. Malgosa-Sanahuja, "Passive In-Band Network Telemetry Systems: The Potential of Programmable Data Plane on Network-Wide Telemetry," *IEEE Access*, vol. 9, pp. 20391–20409, 2021.

[8] L. Andersson, S. Bryant, M. Bocci, and T. Li, "MPLS Network Actions (MNA) Framework," Internet-Draft draft-ietf-mpls-mna-fwk-15, IETF, Dec. 2024. Work in Progress.

[9] M. Naagas and A. P. Gamilla, "Denial of Service Attack: An Analysis to IPv6 Extension Headers Security Nightmares," *International Journal of Electrical and Computer Engineering (IJECE)*, June 2022.

[10] A. Custura, R. Secchi, E. Boswell, and G. Fairhurst, "Is it possible to extend IPv6?," *Computer Communications*, vol. 214, pp. 90–99, Jan. 2024.

[11] A. Gamilla and M. Naagas, "Header of Death: Security Implications of IPv6 Extension Headers to the Open-Source Firewall," *Bulletin of Electrical Engineering and Informatics*, vol. 11, pp. 319–326, Feb. 2022.

[12] F. Gont, N. Hilliard, G. Döring, W. A. Kumari, G. Huston, and W. S. LIU, "Operational Implications of IPv6 Packets with Extension Headers." RFC 9098, Sept. 2021.

[13] L. Hendriks, P. Velan, R. d. O. Schmidt, P.-T. de Boer, and A. Pras, "Threats and Surprises behind IPv6 Extension Headers," in *Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–9, June 2017.

[14] J. Rajamanickam, R. Gandhi, R. Zigler, H. Song, and K. Kompella, "MPLS Network Action (MNA) Sub-Stack Solution," Internet-Draft draft-ietf-mpls-mna-hdr-12, IETF, Mar. 2025. Work in Progress.

[15] MPLS Working Group, "Multiprotocol Label Switching (mpls))." https://datatracker.ietf.org/wg/mpls/documents/. *Last accessed on 09.10.2024*.

[16] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture." RFC 8402, July 2018.

[17] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing with the MPLS Data Plane." RFC 8660, Dec. 2019.

[18] L. Andersson, K. Kompella, and A. Farrel, "Special-Purpose Label Terminology." RFC 9017, Apr. 2021.

[19] H. Song, T. Zhou, L. Andersson, Z. J. Zhang, and R. Gandhi, "MPLS Network Actions using Post-Stack Extension Headers," Internet-Draft draft-song-mpls-extension-header-13, IETF, Oct. 2023. Work in Progress.

[20] T. Saad, K. Makhijani, H. Song, and G. Mirsky, "Use Cases for MPLS Network Action Indicators and MPLS Ancillary Data," Internet-Draft draft-ietf-mpls-mna-usecases-15, IETF, Oct. 2024. Work in Progress.

[21] T. Saad, I. Meilik, T. Li, and J. Drake, "MPLS Network Actions for No Further Fast Reroute," Internet-Draft draft-li-mpls-mna-nffrr-01, IETF, Oct. 2022. Work in Progress.

[22] F. Brockners, S. Bhandari, and T. Mizrahi, "Data Fields for In Situ Operations, Administration, and Maintenance (IOAM)." RFC 9197, May 2022.

[23] H. Song, B. Gafni, F. Brockners, S. Bhandari, and T. Mizrahi, "In Situ Operations, Administration, and Maintenance (IOAM) Direct Exporting." RFC 9326, Nov. 2022.

[24] M. Vigoureux, S. Bryant, and M. Bocci, "MPLS Generic Associated Channel." RFC 5586, June 2009.

[25] N. Sprecher and L. Fang, "An Overview of the Operations, Administration, and Maintenance (OAM) Toolset for MPLS-Based Transport Networks." RFC 6669, July 2012.

[26] G. Mirsky, M. Boucadair, and T. Li, "Supporting In-Situ Operations, Administration, and Maintenance Direct Export Using MPLS Network Actions," Internet-Draft draft-mb-mpls-ioam-dex-08, IETF, June 2024. Work in Progress.

[27] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture." RFC 7665, Oct. 2015.

[28] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)." RFC 8300, Jan. 2018.

[29] G. Fioccola, M. Cociglio, G. Mirsky, T. Mizrahi, and T. Zhou, "Alternate-Marking Method." RFC 9341, Dec. 2022.

[30] G. Fioccola, M. Cociglio, A. Sapio, R. Sisto, and T. Zhou, "Clustered Alternate-Marking Method." RFC 9342, Dec. 2022.

[31] S. Bryant, C. Pignataro, M. Chen, Z. Li, and G. Mirsky, "MPLS Flow Identification Considerations." RFC 8372, May 2018.

[32] W. Cheng, X. Min, T. Zhou, J. Dai, and Y. Peleg, "Encapsulation For MPLS Performance Measurement with Alternate-Marking Method," Internet-Draft draft-ietf-mpls-inband-pm-encapsulation-18, IETF, Oct. 2024. Work in Progress.

[33] W. Cheng, X. Min, R. Gandhi, G. Mirsky, and G. Fioccola, "MNA for Performance Measurement with Alternate Marking Method," Internet-Draft draft-cx-mpls-mna-inband-pm-06, IETF, Mar. 2025. Work in Progress.

[34] A. Farrel, J. Drake, R. Rokui, S. Homma, K. Makhijani, L. M. Contreras, and J. Tantsura, "A Framework for Network Slices in Networks Built from IETF Technologies." RFC 9543, Mar. 2024.

[35] T. Saad, V. P. Beeram, J. Dong, J. M. Halpern, and S. Peng, "Realizing Network Slices in IP/MPLS Networks," Internet-Draft draft-ietf-teas-ns-ip-mpls-05, IETF, Mar. 2025. Work in Progress.

[36] T. Li, J. Drake, V. P. Beeram, T. Saad, and I. Meilik, "MPLS Network Actions for Network Resource Partition Selector," Internet-Draft draft-li-mpls-mna-nrp-selector-01, IETF, June 2024. Work in Progress.

[37] P. Pan, G. Swallow, and A. Atlas, "RFC4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels," May 2005.

[38] S. Deering and R. Hinden, "RFC2460: Internet Protocol Version 6 (IPv6) Specification," Dec. 1998.

[39] R. Léas, J. Iurman, E. Vyncke, and B. Donnet, "Measuring IPv6 Extension Headers Survivability with James," in *ACM Internet Measurement Conference*, p. 746–747, 2022.

[40] Z. Liu, G. Lv, J. Wang, and X. Yang, "A Programmable SRv6 Processor for SFC," *MDPI Electronics*, vol. 11, Sept. 2022.

[41] D. Lebrun and O. Bonaventure, "Implementing IPv6 Segment Routing in the Linux Kernel," in *Applied Networking Research Workshop*, p. 35–41, July 2017.

[42] J. Iurman, B. Donnet, and F. Brockners, "Implementation of IPv6 IOAM in Linux Kernel," in *Netdev*, Aug. 2020.

[43] Chuck Tato, "Segment Routing Over IPv6 Acceleration Using Intel FPGA Programmable Acceleration Card N3000." https://www.intel.com/content/dam/www/central-libraries/us/en/documents/wp-01295-hcl-segment-routing-over-ipv6-acceleration-using-intel-fpga-programmable-acceleration-card-n3000-white-paper.pdf, Oct. 2019. *Last accessed on 17.12.2024*.

[44] P4 Language Consortium, "GitHub: Behavioural Model Version 2 (BMv2)." https://github.com/p4lang/behavioral-model. *Last accessed on 09.10.2024*.

[45] Intel®, "$P4_{16}$ Intel® Tofino™ Native Architecture – Public Version." https://github.com/barefootnetworks/Open-Tofino/blob/master/

PUBLIC_Tofino-Native-Arch.pdf, Apr. 2021. *Last accessed on 09.10.2024*.

[46] The P4 Language Consortium, "$P4_{16}$ Language Specification." https://p4.org/p4-spec/docs/P4-16-v1.2.2.pdf, May 2021. *Last accessed on 09.10.2024*.

[47] F. Hauser, M. Häberle, 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," *Journal of Network and Computer Applications (JNCA)*, vol. 212, Mar. 2023.

[48] E. Zink, M. Flüchter, S. Lindner, F. Ihle, and M. Menth, "Rust Barefoot Runtime (RBFRT): Fast Runtime Control for the Intel Tofino," in *KuVS Workshop on Network Softwarization (KuVS NetSoft)*, Apr. 2025.

[49] Chair of Communication Networks, University of Tuebingen, "GitHub: P4-MNA." https://github.com/uni-tue-kn/P4-MNA.

[50] D. Merling, S. Lindner, and M. Menth, "Hardware-Based Evaluation of Scalable and Resilient Multicast With BIER in P4," *IEEE Access*, vol. 9, pp. 34500–34514, 2021.

[51] S. Lindner, M. Häberle, and M. Menth, "P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks," *IEEE Access*, vol. 11, pp. 17525–17535, Feb. 2023.

[52] F. Ihle, E. Zink, S. Lindner, and M. Menth, "Enhancements to P4TG: Protocols, Performance, and Automation," in *KuVS Workshop on Network Softwarization (KuVS NetSoft)*, Apr. 2025.

[53] Y. Choi, J.-S. Yoon, Y. Moon, and K. Park, "Is Large MTU Beneficial to Cellular Core Networks?," *Asia-Pacific Workshop on Networking*, June 2023.

[54] D. Franco, E. Ollora Zaballa, M. Zang, A. Atutxa, J. Sasiain, A. Pruski, E. Rojas, M. Higuero, and E. Jacob, "A Comprehensive Latency Profiling Study of the Tofino P4 Programmable ASIC-based Hardware," *Computer Communications*, vol. 218, pp. 14–30, 2024.

[55] Y.-W. Chen, C.-Y. Li, C.-C. Tseng, and M.-Z. Hu, "P4-TINS: P4-Driven Traffic Isolation for Network Slicing With Bandwidth Guarantee and Management," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3290–3303, 2022.

[56] X. Xu, S. Kini, P. Psenak, C. Filsfils, S. Litkowski, and M. Bocci, "Signaling Entropy Label Capability and Entropy Readable Label Depth Using IS-IS." RFC 9088, Aug. 2021.

[57] X. Xu, S. Kini, P. Psenak, C. Filsfils, S. Litkowski, and M. Bocci, "Signaling Entropy Label Capability and Entropy Readable Label Depth Using OSPF." RFC 9089, Aug. 2021.

[58] Juniper Networks, "Load Balancing MPLS Traffic." https://www.juniper.net/documentation/us/en/software/junos/mpls/topics/topic-map/load-balancing-mpls-traffic.html. *Last accessed on 01.08.2024*.

[59] Huawei, "Configuring the ECMP Load Balancing Mode." https://support.huawei.com/enterprise/en/doc/EDOC1100137942/60585466/configuring-the-ecmp-load-balancing-mode/. *Last accessed on 01.08.2024*.

[60] Cisco, "ASR9000XR: Load-balancing Architecture and Characteristics." https://community.cisco.com/t5/service-providers-knowledge-base/asr9000-xr-load-balancing-architecture-and-characteristics/ta-p/3124809#field. *Last accessed on 05.03.2025*.

[61] J. Rajamanickam, R. Gandhi, R. Zigler, T. Li, and J. Dong, "Post-Stack MPLS Network Action (MNA) Solution," Internet-Draft draft-ietf-mpls-ps-mna-hdr-00, IETF, Feb. 2025. Work in Progress.

[62] A. Farrel and S. Farrell, "Opportunistic Security in MPLS Networks," Internet-Draft draft-ietf-mpls-opportunistic-encrypt-03, IETF, Mar. 2017. Work in Progress.

[63] F. Ihle and M. Menth, "MPLS Network Actions: Technological Overview and P4-Based Implementation on a High-Speed Switching ASIC," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 3480–3501, Apr. 2025.

[64] F. Ihle and M. Menth, "Erratum to "MPLS Network Actions: Technological Overview and P4-Based Implementation on a High-Speed Switching ASIC"," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 4341–4341, 2025.

Fabian Ihle received his bachelor's (2021) and master's degrees (2023) in computer science at the University of Tuebingen. Afterwards, he joined the communication networks research group of Prof. Dr. habil. Michael Menth as a Ph.D. student. His research interests include software-defined networking, P4-based data plane programming, resilience, and Time-Sensitive Networking (TSN).

Michael Menth, (Senior Member, IEEE) is professor at the Department of Computer Science at the University of Tuebingen/Germany and chairholder of Communication Networks since 2010. He studied, worked, and obtained diploma (1998), PhD (2004), and habilitation (2010) degrees at the universities of Austin/Texas, Ulm/Germany, and Wuerzburg/Germany. His special interests are performance analysis and optimization of communication networks, resilience and routing issues, as well as resource and congestion management. His recent research focus is on network softwarization, in particular P4-based data plane programming, Time-Sensitive Networking (TSN), Internet of Things, and Internet protocols. Dr. Menth contributes to standardization bodies, notably to the IETF.