Implementation Experience of MNA Using P4 on Intel Tofino Hardware

by Fabian Ihle <fabian.ihle@uni-tuebingen.de>

http://kn.inf.uni-tuebingen.de
Agenda

► Background & Current state of the MNA framework

► Mutable data in MNA
  ▪ HBH NAS Replication and Preservation
  ▪ To PSD or not to PSD

► Operation modes

► Details about the implementation

► P4-MNA example network action

► Discussion
Legend

- Each slide has a colored bookmark to indicate if its content is from IETF drafts or proposed by us
  - Specific parts in the slides are marked with colored circles

- From IETF drafts
- New proposal
- Topic for discussion
For MNA to be successful, it must be efficiently implementable in hardware
  ▪ Similar case: IPv6 EH

Intel Tofino™ as reference architecture for a hardware switching ASIC
  ▪ Limited resources and operations
  ▪ Pipelined architecture

If encoding works on the Intel Tofino™, it most likely also works on vendor ASICs

Questions
  1. Can the IETF MNA encoding be implemented efficiently on hardware?
  2. What are the challenges in an implementation?
  3. To PSD or not to PSD?
MNA

BACKGROUND
The MNA Framework

- General mechanism for transmitting, processing and executing predefined network actions

- New generalized header encoding introduced for specifying network actions and ancillary data

- The Network Action Sub-stack (NAS)
  - A stack of related LSEs containing
    - Network actions
    - Ancillary data
  - Inserted into the MPLS stack
    - Below forwarding labels

What does a NAS look like?
- bSPL followed by network actions and ancillary data
  - = NAS indicator

LSE encoding redefined for network actions and ancillary data in a NAS
- Initial opcode (mandatory, Format B)
- Subsequent opcode (optional, Format C)
- Ancillary data (optional, Format D)
Some valid NAS

<table>
<thead>
<tr>
<th>MNA-Label = bSPL</th>
<th>TC</th>
<th>S</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Opcode</td>
<td>Data</td>
<td>R</td>
<td>IHS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MNA-Label = bSPL</th>
<th>TC</th>
<th>S</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Opcode</td>
<td>Data</td>
<td>R</td>
<td>IHS</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MNA-Label = bSPL</th>
<th>TC</th>
<th>S</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Opcode</td>
<td>Data</td>
<td>R</td>
<td>IHS</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MNA-Label = bSPL</th>
<th>TC</th>
<th>S</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Opcode</td>
<td>Data</td>
<td>R</td>
<td>IHS</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>Sub. Opcode</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Data</td>
<td>S</td>
<td>Data</td>
</tr>
</tbody>
</table>
**Discussion**: Can we move NASL + U + IHS + R into the TTL field of the bSPL and eliminate the initial opcode encoding?

- Simplifies the NAS encoding and saves 8 bits
- Right now, the initial opcode cannot carry ancillary data LSEs
- Definition of network actions: Having 2 encodings for opcodes with different amounts of data bits available requires to
  a) Constrain specific opcodes to initial / subsequent encoding only, or
  b) Have multiple definitions for network actions

**Are there implications with redefining values in a bSPL?**

---


---

Implementation Experience of MNA Using P4 on Intel Tofino Hardware | MPLS Interim Meeting 2024-04-24

Fabian Ihle
How to PSD in MNA

- Post-stack data (PSD) inserted below bottom of MPLS stack

- Dedicate a bit to indicate presence of PSD ("P bit")
  - Current proposal: Reserved Bit in initial opcode
  - Ongoing discussion: Use a different bit or an opcode, or don’t use PSD at all

- Post-stack encoding
  - Same principle as a NAS
  - Post-Stack Network Action Top Header ≡ NAS indicator
  - Post-Stack Network Action Header
    - encodes network action

Scopes and Placement in the MPLS Stack

► Scope defines on which nodes a network action is to be processed
  ▪ Set in IHS field in initial opcode LSE
  ▪ *HBH*: On every node along the path
  ▪ *Select*: Only on specific nodes on the path
  ▪ *I2E*: Only on the egress node

► At most one NAS per scope per hop should be allowed
  ▪ A NAS must not appear on top of an MPLS stack
  ▪ A node that exposes a NAS to the top must pop it

► We look at SR-MPLS!
  ▪ MNA with label switching is easier

► Ingress LER must ensure that the NAS can be read by the relevant nodes
  ▪ LSR signal their *RLD* (readable label depth)
    ▪ Number of LSE a node can read without performance impact
    ▪ → Multiple copies of HBH NAS may need to be placed in the stack
      ▪ = HBH NAS Replication

Mutable data = data where the value changes during packet forwarding

Example: Collecting telemetry data along a path
- Each node on the path adds information to the NAS, e.g., the node ID
- That information must not be discarded on forwarding

For backward compatibility, only last 8 bits are mutable
- MPLS label and TC may be hashed for ECMP

Challenges
- How to carry more mutable data?
- How to carry mutable data with HBH NAS replication?

Problem: HBH NAS Replication with Mutable Data

Note: NAS consist of more than one LSE. For simplicity they are counted as one here.

Implementation Experience of MNA Using P4 on Intel Tofino Hardware | MPLS Interim Meeting 2024-04-24

Fabian Ihle
With HBH Label Replication …

- Multiple HBH copies may have to be present in the stack
  - → The stack grows larger 😊
  - Creates problems with mutable data

Our proposal: HBH NAS Preservation

- Place a single HBH NAS below the top-of-stack MPLS segment label
- A node…
  1. Pops the topmost MPLS segment label
  2. Moves the HBH scope below the next MPLS label
     - Actually: Moves the next MPLS label to the top

Advantages

- The stack only holds one HBH NAS
- Mutable data is up to date inherently
- RLD of nodes is not that important → We do not have to search for a buried HBH NAS

Disadvantages

- Every node on the path MUST be MNA-capable
- Does not follow the traditional MPLS paradigm "push, pop, swap"
Note: NAS consist of more than one LSE. For simplicity they are counted as one here.
Mutable data in MNA
- Only the last 8 bits in an LSE are mutable due to ECMP hashing compatibility
- Modifications to mutable data in HBH scopes must be reflected in all HBH copies in the stack
  - HBH Label Preservation fixes this, but only works if all nodes support MNA 😊

PSD solves the issue with mutable data in the HBH NAS replication example
- PSD not hashed for ECMP: All bits are mutable
- Mutable data kept in one place: post-stack → no copying across different NAS
- But can it be implemented efficiently?
  - Challenge: How to get to PSD?
    a) Jump over the MPLS stack to the end using a pointer or lookahead
    b) Parse the full MPLS stack
  - In P4 we have to parse the full stack!
1. Compatibility Mode
   - Implementation with HBH NAS Replication
   - Brownfield deployment: Mixture of MNA-incapable and MNA-capable nodes
     a) ISD only
        - Less restrictive wrt. RLD, but only few mutable data bits available
        - Use case: Network actions that do not require mutable data
          • E.g., draft-ietf-mpls-inband-pm-encapsulation-11 (RFC9341), IOAM DEX
     b) With ISD and PSD
        - More mutable data available, but we have to get to PSD
        - Use case: Network actions that require more mutable data
          • E.g., INT / IOAM without DEX

2. Greenfield Mode
   - Implementation with HBH NAS Preservation
   - Greenfield deployment: Only MNA-capable nodes
   - Require Entropy Labels
     - Stack must not need to be hashed
     - All data bits can be mutable
     - ISD only, no need for PSD
   - Use case: all of the above
Compatibility Mode ISD only, Compatibility Mode with ISD and PSD, Greenfield Mode

P4-MNA IMPLEMENTATION
The P4 Programming Language

- High-level programming language to describe data plane
  - Compiler maps P4 program onto programmable pipeline of target
  - Pipeline: Parser → Control block → Deparser

- P4 control blocks define packet processing operations
  - Match+Action Units
  - Branching constructs, logical and simple arithmetic expressions
  - Recirculation models loop behaviour
The dynamic combination of AD LSE and subsequent opcodes requires a lot of state in the parser.

Therefore, we ignore the different encodings of subsequent opcodes and AD entries:
- They are all parsed as a single subsequent opcode stack using the NASL length field.
- The semantics of AD and opcodes are later derived in the control block using the NAL length field.
The internal representation of the MPLS stack in the P4 program is divided into 3 substacks:

1. Active Segment Stack
   - Forwarding label of this segment
   - NAS relevant for the node of this segment

2. Intermediate Segments Stack
   - LSE that are not relevant for this node, but must be parsed, if
     - A HBH NAS buried in the stack has to be searched
     - We have to reach the bottom of stack, e.g., for PSD

3. Persistent Stack
   - HBH NAS buried in the stack
   - PSD

This is only specific to our P4 implementation!

Each substack has a fixed (maximum) allocated size:
- The space is limited
- Substack sizes can be optimized depending on requirements
  - Long paths in the network → increase intermediate segments stack size
  - Many / large HBH actions → increase persistent stack size
**Compatiability Mode ISD only**
- Multiple HBH replications

**Compatiability Mode with ISD and PSD**
- Single HBH NAS at bottom: In RLD anyway

**Greenfield Mode**
- Single HBH NAS at top
One match+action table (MAT) for each LSE in a NAS required
- Each MAT holds network actions for an LSE in the NAS
- 16 MATs for HBH scope, 16 MATs for Select Scope
- We ignore the I2E scope for now
- MATs are applied sequentially

We ignored the different LSE encodings during parsing 😐
- How to find out if an LSE is an opcode or contains AD?
- We match on the NAL field of a subsequent opcode LSE
- We execute an action that treats the next <NAL> LSE as AD entries

Pipeline space is limited
- We can fit \( n \) MATs with network actions into one pipeline iteration
  - Then we have to recirculate
► In the first iteration a maximum NAS with 16 network actions can be processed!
► Recirculations are only required in the worst case, i.e., if we have multiple NAS
Requirements to the P4-MNA PSD Implementation

1. Everything must be within RLD!
   - Full MPLS stack and PSD
     - We can do this on the Intel Tofino™ at line-rate 😊

2. Optimize the distribution of the substacks to the network requirements

- PSD processing may increase the number of recirculations
  - Parsing and processing is more difficult with PSD than with ISD only
  - Future work: Optimize the number of recirculations!

Fig.: Example of how the stack sizes could be distributed.
If we are doing the **Compatibility Mode with PSD** in P4…

- We have the whole MPLS stack within RLD
- → We do not need copies of HBH NAS in the stack
- → We place a single HBH NAS at the bottom of stack to indicate the PSD
- → We can place all HBH network actions and their data in PSD
- → We can shrink the size of the HBH NAS to
  - 2 LSE (bSPL + initial opcode), or
  - 1 LSE if we move the flags of the initial opcode into the bSPL
- → This reduces the number of recirculations in our implementation
- This does not require changes in the encoding / specification!

**Discussion:** Should we use the HBH NAS only to indicate PSD?

- Do there have to be “ISD-only” HBH actions in parallel with PSD or can we move everything to PSD?
- Is this an implementation-specific constraint or should this be considered in a draft?

---

Fig.: Example of how the stack sizes could be distributed.
Comp. Mode ISD Only vs. Greenfield Mode

P4-MNA EXAMPLE NETWORK ACTION
Simplified version of a “Path Tracing” mechanism in the context of IOAM

- **Goal:** Collect the Node ID of each traversed node and evaluate the path at the tail end node
- **Method:**
  1. Head end node pushes NAS with opcode to indicate that INT is desired
  2. Treat data field of subsequent opcodes as flags: Indicate which data to collect
  3. Reserve AD LSE to fill in the data → transit nodes do not have to add LSE
  4. Mutable data field of sub. Opcode holds index of position to be filled with data
  5. Node ID (e.g., 4 bit) is filled into data field of AD LSE

---

**Implementation Experience of MNA Using P4 on Intel Tofino Hardware**

1. Opcode 42 to indicate “IOAM”
2. Flag set to indicate “Node ID collection”
3. 5 AD entries reserved in this case
4. Index where to write data in AD
5. Node ID will be filled into the data field of AD LSE
P4-MNA Example Network Action: Comp. Mode ISD Only

Layer 2 header
- MPLS label A
- MPLS label B
- MPLS label C
- MPLS label D
- MPLS label E
- NAS indicator
- Initial opcode
  - op=42, flag=16, idx=0
  - AD, data=0
  - AD, data=A
  - AD, data=0
  - ∆D, data=0
  - AD, data=0
  - Layer 3 header

Layer 2 header
- MPLS label B
- MPLS label C
- MPLS label D
- MPLS label E
- NAS indicator
- Initial opcode
  - op=42, flag=16, idx=1
  - AD, data=A
  - AD, data=0
  - AD, data=B
  - AD, data=0
  - ∆D, data=0
  - AD, data=0
  - Layer 3 header

Layer 2 header
- MPLS label C
- MPLS label D
- MPLS label E
- NAS indicator
- Initial opcode
  - op=42, flag=16, idx=2
  - AD, data=A
  - AD, data=B
  - AD, data=0
  - ∆D, data=0
  - AD, data=C
  - AD, data=0
  - Layer 3 header

Layer 2 header
- MPLS label D
- MPLS label E
- NAS indicator
- Initial opcode
  - op=42, flag=16, idx=3
  - AD, data=A
  - AD, data=B
  - AD, data=0
  - ∆D, data=C
  - AD, data=0
  - AD, data=0
  - Layer 3 header

Layer 2 header
- MPLS label E
- NAS indicator
- Initial opcode
  - op=42, flag=16, idx=4
  - AD, data=A
  - AD, data=B
  - AD, data=0
  - ∆D, data=C
  - AD, data=0
  - AD, data=0
  - Layer 3 header

LSR A ➔ LSR B ➔ LSR C ➔ LSR D ➔ LSR E
P4-MNA Example Network Action: Greenfield Mode

Implementation Experience of MNA Using P4 on Intel Tofino Hardware | MPLS Interim Meeting 2024-04-24 | Fabian Ihle
We proposed 3 operational modes for deployment of MNA in the network to address the identified challenges:

1. Compatibility mode with ISD only
2. Compatibility mode with ISD and PSD
3. Greenfield Mode

- We implemented all 3 modes in P4 on the Intel Tofino™ 1
  - Side products: Python library for building MNA packets, Wireshark dissector extension for visualizing MNA traffic

1. Can the IETF MNA encoding be implemented efficiently on hardware?
   - → Yes! The encoding works. Depending on the size and distribution of the substacks, we have to recirculate

2. What are the challenges in an implementation?
   - → Mutable data is difficult with ISD and HBH NAS Replication, PSD brings some harsh requirements
Future Work

► Optimize the implementation
  ▪ Reduce the number of required recirculations for large stacks
  ▪ Better performance (more LSE and less recirculations) with the Intel Tofino™ 2 expected
  ▪ Make the greenfield mode more compatible with MNA-incapable nodes

► Extend the range of implemented network actions
  ▪ E.g., draft-ietf-mpls-inband-pm-encapsulation-11 (RFC9341), IOAM DEX

► Paper about our findings is WIP
  ▪ Open for collaborations!
Thanks for your time!

DISCUSSION
<table>
<thead>
<tr>
<th>Requirements to Nodes</th>
<th>Compatibility Mode with ISD only</th>
<th>Compatibility Mode with ISD and PSD</th>
<th>Greenfield Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNA-capable + MNA-incapable</td>
<td>MNA-capable + MNA-incapable</td>
<td>MNA-capable + MNA-incapable</td>
<td>MNA-capable</td>
</tr>
<tr>
<td>Requirements to RLD</td>
<td>Active Segment Stack up until first HBH NAS is found</td>
<td>Full MPLS Stack and PSD in RLD</td>
<td>Active Segment Stack + next MPLS label</td>
</tr>
<tr>
<td>HBH NAS Handling</td>
<td>HBH NAS Replication</td>
<td>Single HBH NAS at bottom of stack</td>
<td>HBH NAS Preservation</td>
</tr>
<tr>
<td># Mutable Bits per LSE</td>
<td>• 0 if multiple HBH NAS present in stack &lt;br&gt; • 4 (sub. Opcode), 8 (AD LSE) if single HBH NAS in stack which is placed at the bottom</td>
<td>• 32 bits (Post-stack encoding)</td>
<td>• 30 bits (AD LSE) &lt;br&gt; • 16 bits (sub. Opcode)</td>
</tr>
<tr>
<td>Location of Mutable Bits</td>
<td>ISD</td>
<td>PSD</td>
<td>ISD</td>
</tr>
<tr>
<td>Other Requirements</td>
<td>-</td>
<td>-</td>
<td>Entropy Labels</td>
</tr>
<tr>
<td>Use Cases</td>
<td>Network actions that do not require mutable data, e.g., IOAM-DEX</td>
<td>Network actions that require mutable data, e.g., Per-hop tracing IOAM</td>
<td>All of the others, greenfield deployment, e.g., satellite networks</td>
</tr>
</tbody>
</table>
BACKUP SLIDES
MPLS and Special Purpose Labels (SPL)

- **MPLS label stack**
  - … consists of Label Stack Entries (LSE)

- **Forwarding labels**
  - Indicated in orange

- **Base Special Purpose Labels (bSPL)**
  - Indicated in red
  - 16 reserved label values for special purposes
  - E.g., ECMP load balancing (Entropy labels)

- **Extended Special Purpose Labels (eSPL)**
  - Increase the range of reserved label values for special purposes
  - ✳ Some extensions assume only one eSPL per packet
  - We want to be flexible with network actions!
The MNA Header Encoding

- NAS indicator
  - bSPL entry to indicate a following NAS
  - “Traditional“ LSE encoding

MPLS label = bSPL

<table>
<thead>
<tr>
<th>Bit</th>
<th>Initial Opcode</th>
<th>Sub. Opcode</th>
<th>Data</th>
<th>S</th>
<th>Data</th>
<th>NASL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R IHS S Res U</td>
<td>1</td>
<td>S</td>
<td></td>
<td></td>
<td>NAL</td>
</tr>
</tbody>
</table>

The MNA Header Encoding

- Initial Opcode LSE (mandatory)
  - Opcode to identify the first network action. Must be non-zero
  - Data, e.g., flags
  - IHS: Scope of this NAS (more later)
  - NASL: Number of following subsequent opcodes and ancillary data in this NAS
    - here: 3, allows for a maximum of 15 subsequent opcodes and ancillary data entries

![MNA Header Encoding Diagram]

_Initial Opcode LSE (mandatory)
  - Opcode to identify the first network action. Must be non-zero
  - Data, e.g., flags
  - IHS: Scope of this NAS (more later)
  - NASL: Number of following subsequent opcodes and ancillary data in this NAS
    - here: 3, allows for a maximum of 15 subsequent opcodes and ancillary data entries

![MPLS Header Diagram]

---


Implementation Experience of MNA Using P4 on Intel Tofino Hardware | MPLS Interim Meeting 2024-04-24

Fabian Ihle
The MNA Header Encoding

- **Subsequent Opcode LSE (optional)**
  - Opcode to identify the network action. Must be non-zero
  - Data, e.g., flags
  - NAL: Number of related ancillary data entries following this subsequent network action
    - here: 1, and 0 respectively

| Bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|-----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|     |   |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| MPLS label = bSPL | TC | S | TTL |
| Initial Opcode | Data | R | IHS | S | Res | U | NASL |
| Sub. Opcode | Data | S | Data | NAL |
| 1 | Data | S | Data |
| Sub. Opcode | Data | S | Data | NAL |

Ancillary data (AD) LSE (optional)
- Only follows a subsequent opcode
- Most significant bit must be set
- Immutable data
- Mutable data

User-defined packet headers are defined in the P4 program
- Struct objects: reflect header fields, e.g., an MPLS LSE
- Aggregate structs to header stacks, e.g., an MPLS stack
  - Fixed maximum length
  - All entries of this stack must share the same structure!
- Combine header structs and header stack to describe the entire packet header

Programmable P4 parser
- Modeled as finite state machine
  - States extract header information from packets

```
state parse_ethernet {
  pkt.extract(hdr.ethernet);
  transition select(hdr.ethernet.ether_type) {
    ETHERTYPE_IPV4: parse_ipv4;
    ETHERTYPE_MPLS: parse_mpls;
    default: accept;
  }
}
```
Implementation Experience of MNA Using P4 on Intel Tofino Hardware | MPLS Interim Meeting 2024-04-24

P4-MNA Pipeline – Compatibility Mode ISD only

**Ingress control**

- **Stage 0**
  - Select Scope
  - MPLS label A
  - MPLS label B
  - NAS
  - HBH Scope
  - MPLS label C

- **Stage 1**
  - NAS
  - Select Scope

- **Stage 2**
  - MPLS label A
  - MPLS label B
  - NAS
  - HBH Scope
  - MPLS label C

- **Stage 3**
  - NAS
  - HBH Scope

**Egress control**

- **Stage 1**
  - NAS
  - Select Scope

- **Stage 2**
  - MPLS label A
  - MPLS label B
  - NAS
  - HBH Scope
  - MPLS label C

- **Stage 3**
  - NAS
  - HBH Scope

- **Drop unknown opcode?**

- **Recirculate**

Fabian Ihle
Implementation Experience of MNA Using P4 on Tofino Hardware | MPLS Interim Meeting 2024 04 24

Fabian Ihle
1. Match on the NAL field in each MAT along with the opcode
2. Treat \(<\text{NAL}>\) following entries as AD in the matched action
3. Set them to “already processed” in metadata
4. Only match in the next MAT if it is not set to “already processed”

→ Entry[1] is marked as processed
→ We do not treat the AD entry as an opcode!
1. Match on the NAL field in each MAT along with the opcode
2. Treat <NAL> following entries as AD in the matched action
3. Set them to “already processed” in metadata
4. Only match in the next MAT if it is not set to “already processed”
1. Execute the first subsequent opcode
2. Treat the following 2 entries as AD (NAL)
3. Set all 3 entries to „processed“

Entry[1] is marked as processed
We do not treat the AD entry as an opcode!
2. Extending in-stack actions to use PSD

- New ISD network action: Post-Stack Network Action Offset
  - Initial or subsequent opcode
  - Data: Start offset of the Post Stack Network Action Top Header after BoS
    - Value of 1: PS starts 4 bytes after BoS

- New network action: PS-IS-NA ordering
  - Define order of execution for PSD / ISD
  - Data field contains one or more 7-bit opcodes that defines the order of execution

► Signaling
- Nodes must signal their Maximum Post-Stack MNA length

Wireshark Dissector

| Frame 10906: 175 bytes on wire (1408 bits), 176 bytes captured (1408 bits) on interface ens18, id 0 |
| Ethernet II, Src: b2:48:81:3d:33:90 (b2:48:81:3d:33:90), Dst: 08:00:00:00:00:02:22 (08:00:00:00:00:02:22) |
| MPLS Stack |
| MPLS Forwarding Label 50 |
| NAS Select, Length: 7 |
| MPLS Forwarding Label 70 |
| 0000 0000 0000 0000 8106 8110 .... .... .... = Label: 70 |
| .... .... .... .... .... .... = Traffic Class: 7 |
| .... .... .... .... .... 0 .... .... = Bottom of Stack: 0 |
| .... .... .... .... .... 0011 1111 = TTL: 63 |
| NAS Select, Length: 7 |
| NAS BH, Length: 6 |
| MNA bSPL NAS Indicator |
| MNA NAS Initial Opcode LSE |
| 1000 0000 .... .... .... .... .... = Opcode: 64 |
| .... .... 0 0000 0000 0000 .... .... = Data1: 0 |
| .... .... .... .... .... .... .... = Reserved: 0 |
| .... .... .... .... .... .... 01 .... .... = Scope: 1 |
| .... .... .... .... .... .... 0 .... .... = Bottom of Stack: 0 |
| .... .... .... .... .... .... 0000 .... .... = Reserved: 0 |
| .... .... .... .... .... .... .... .... = Unknown Action Handling: 0 |
| .... .... .... .... .... .... .... .... 0110 = NASL: 6 |
| MNA NAS Subsequent Opcode LSE |
| MNA NAS Ancillary data LSE |
| MNA NAS Subsequent Opcode LSE |
| MNA NAS Subsequent Opcode LSE |
| MNA NAS Subsequent Opcode LSE |
| MNA NAS Subsequent Opcode LSE |
| MPLS Forwarding Label 60 |
| Internet Protocol Version 4, Src: 10.0.1.1, Dst: 10.0.2.2 |
| User Datagram Protocol, Src Port: 50081, Dst Port: 50083 |
| Data (18 bytes) |