Securing IOAM in the Linux Kernel Toward Trustworthy In Situ Network Telemetry

Maxime Goffart, Emilien Wansart, Benoit Donnet

Table of Contents

Table of Contents

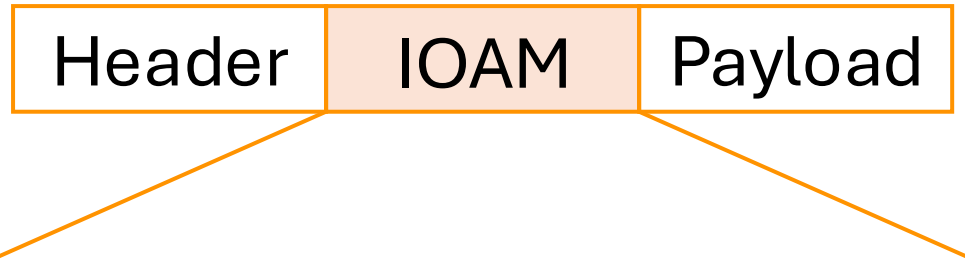
What is IOAM ?

In Situ Operations, Administration, and Maintenance

Standardized with 9 RFCs

Hybrid Telemetry Method

- Active
- Passive



- Instructions on operations to perform
- Optional telemetry data

IOAM – Encapsulation

Encapsulation in Network Protocols

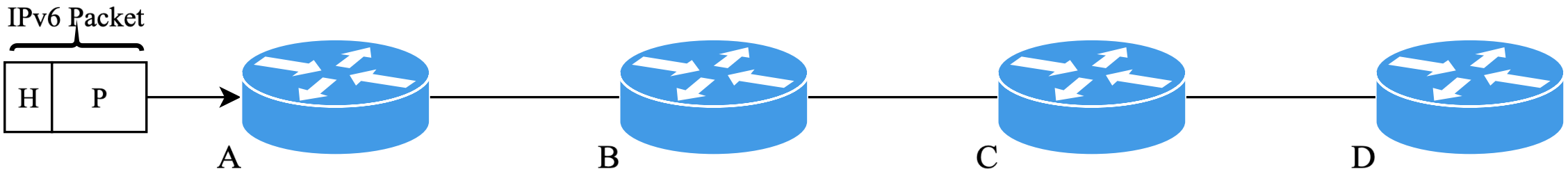
- NSH (RFC 9452)
- IPv6 (RFC 9486)

IOAM – Data fields (RFC 9197)

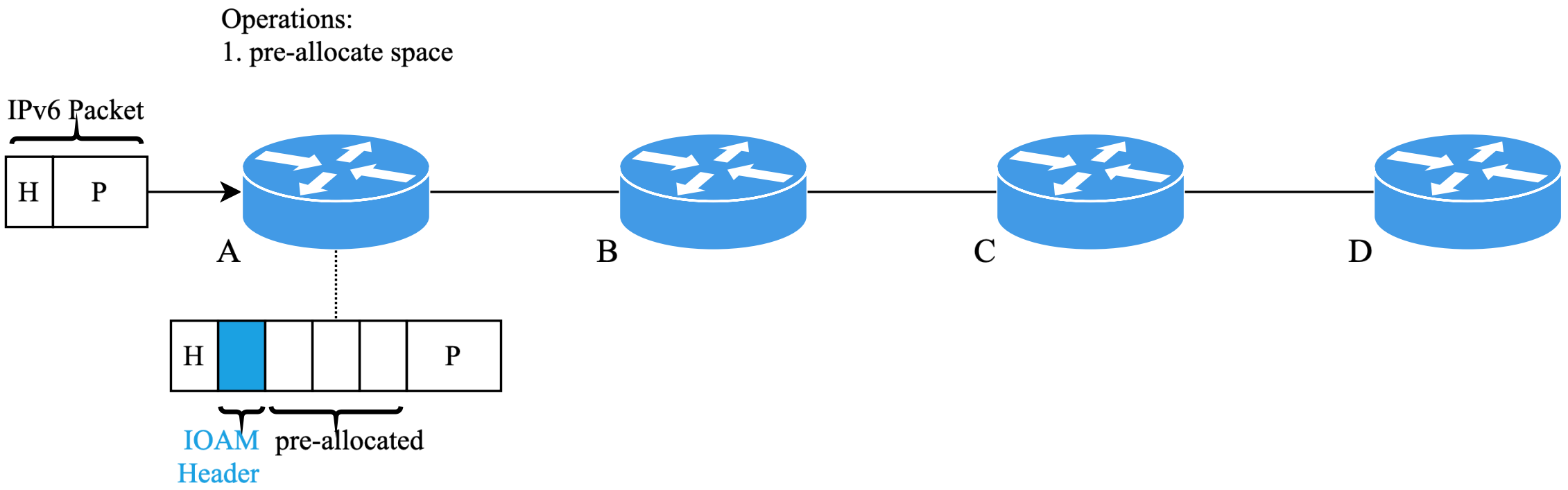
Gathered Telemetry Information

- Hop limit
- Node ID
- Interfaces IDs
- Timestamp
- Queue depth
- …

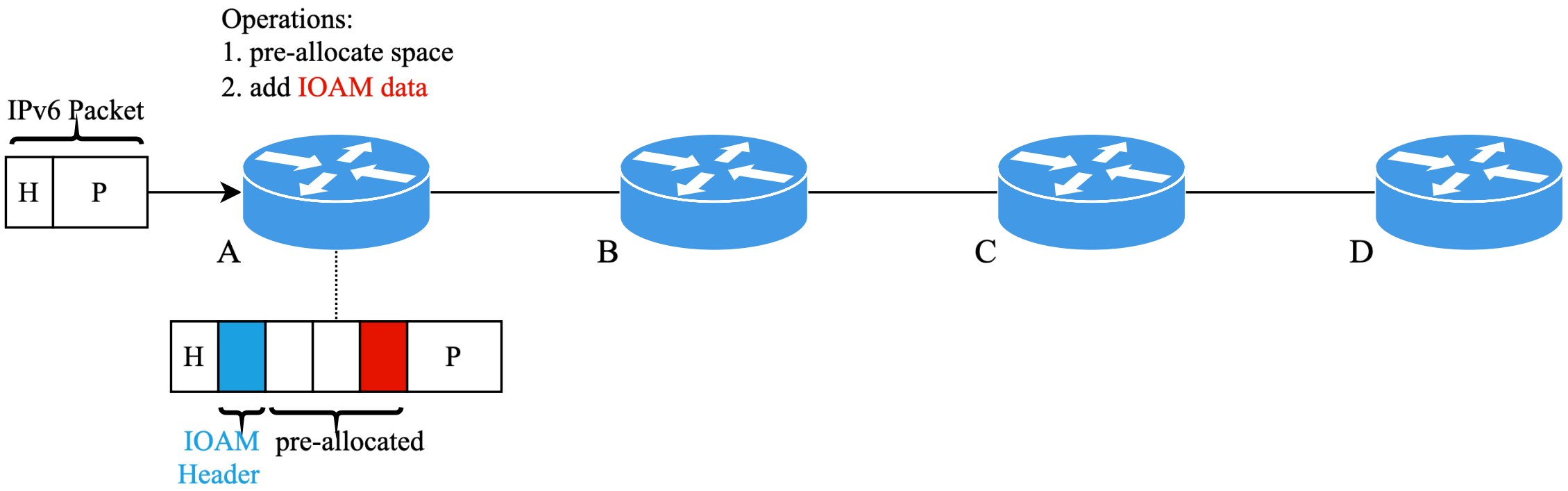
IOAM – PTO Option-Type



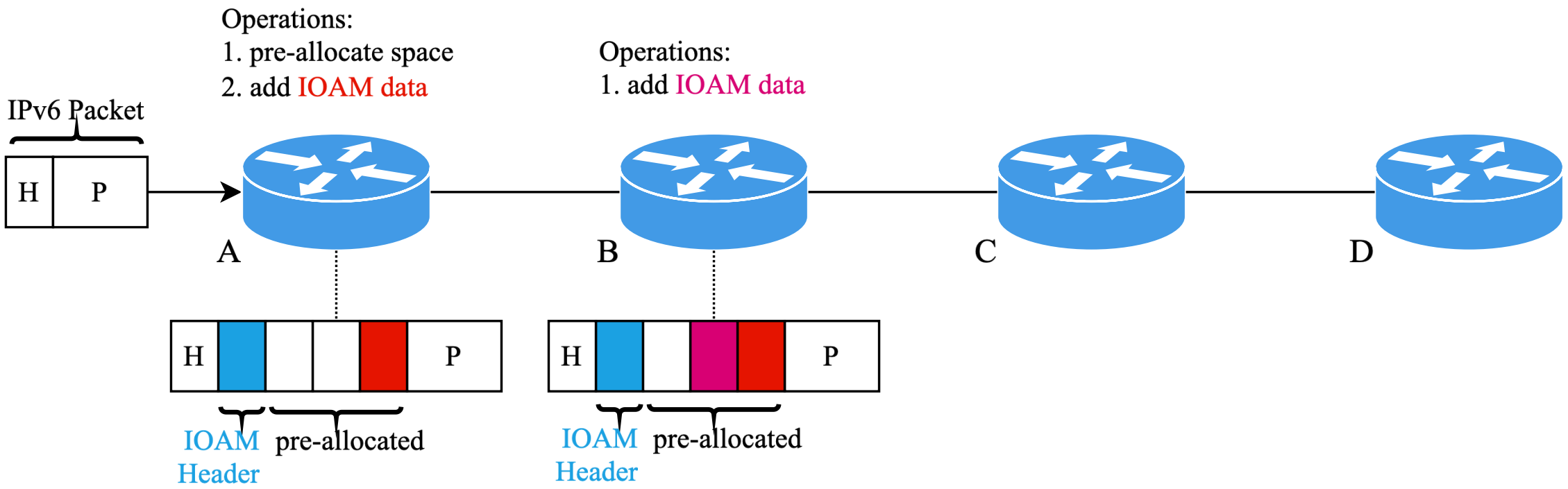
IOAM – PTO Option-Type



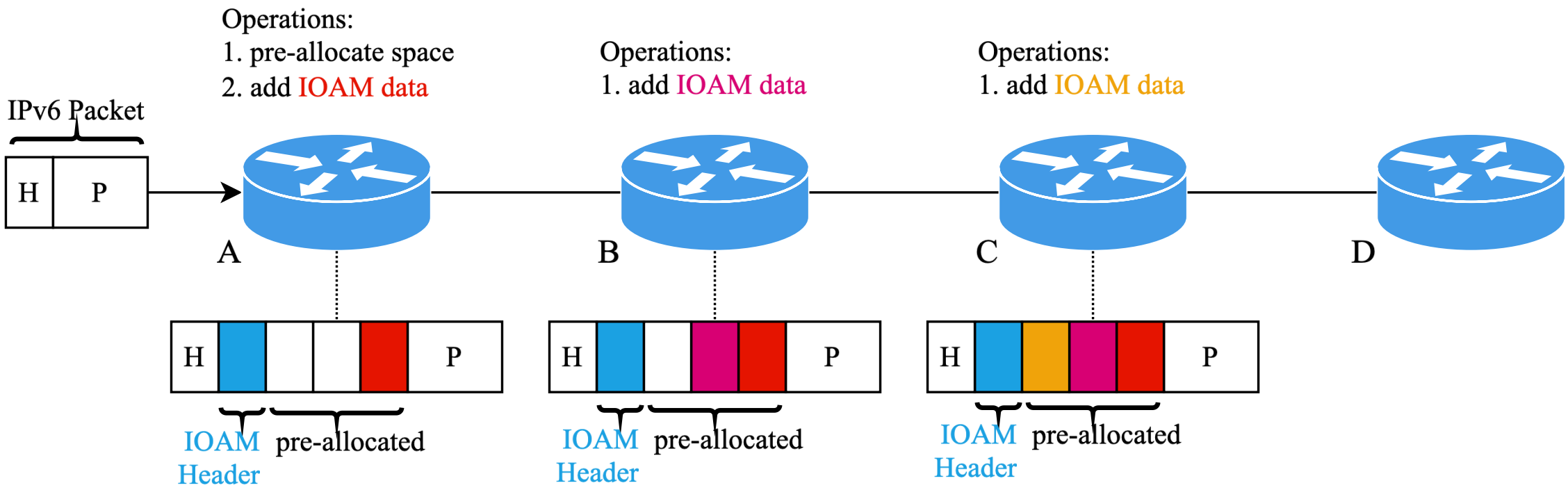
IOAM – PTO Option-Type



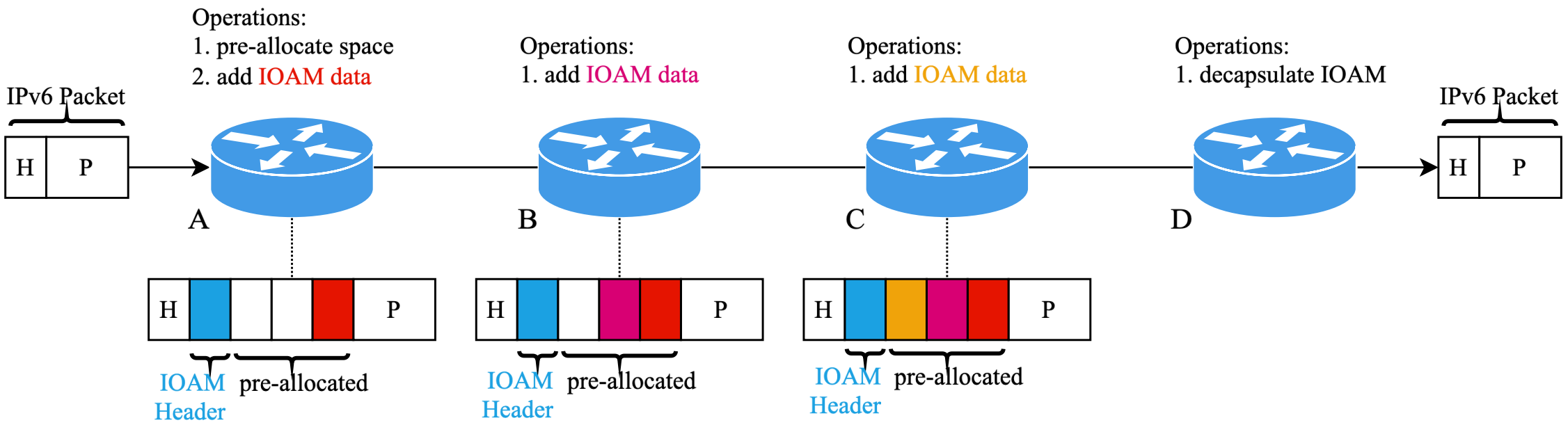
IOAM – PTO Option-Type



IOAM – PTO Option-Type



IOAM – PTO Option-Type



AEAD

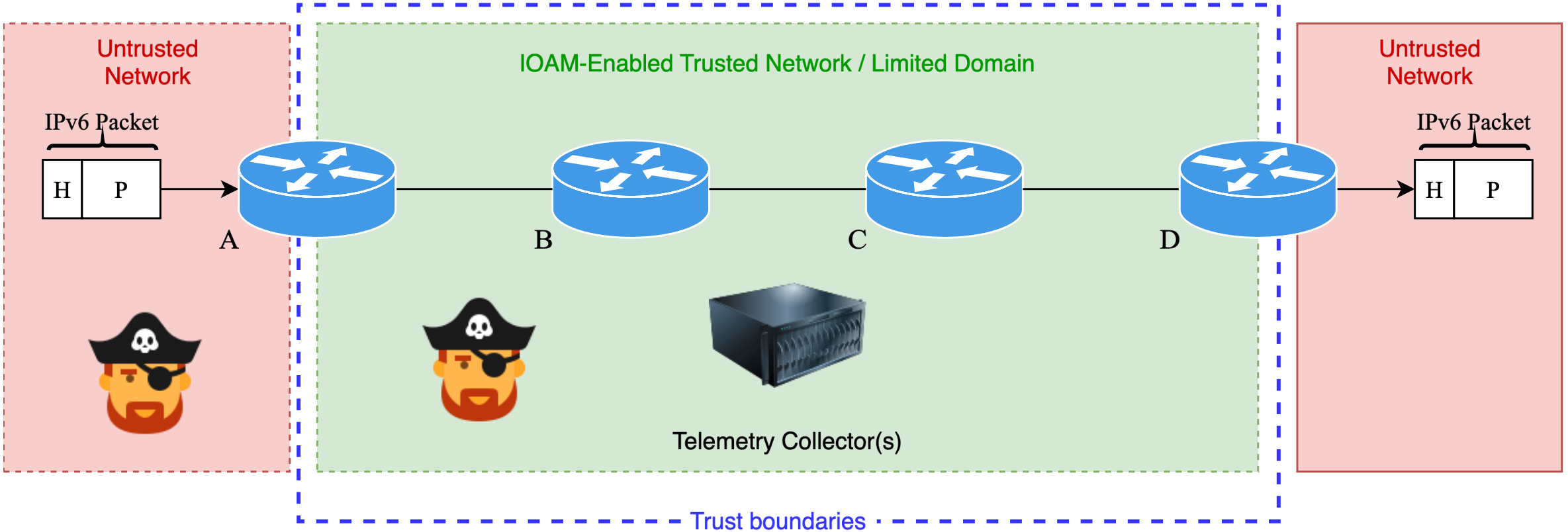
Authenticated Encryption with Associated Data

```
(Cipher, Tag) = AEAD(Key, Nonce, Associated, Plaintext)
```

Key	Key for both encryption **and** authentication
Nonce	**Must** be unique
Associated Data	Authenticated but **not** encrypted
Plaintext	Authenticated **and** encrypted

Table of Contents

IOAM – Threat Model



IOAM – Encryption and Authentication

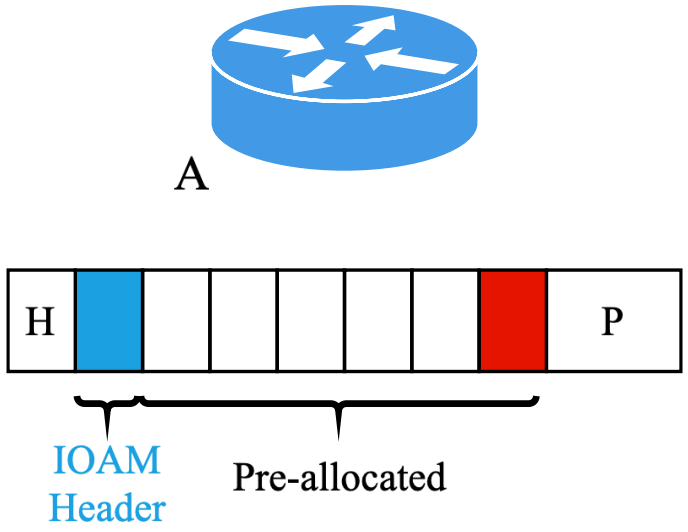
```
(Cipher, Tag) = AEAD(Key, Nonce, Associated, Plaintext)
```

AEAD	AES-GCM or ChaCha20-Poly1305
Key	Unique per <node, namespace> pair
Nonce	Monotonically increasing per-namespace
Associated Data	None
Plaintext	IOAM node data

IOAM – Insertion

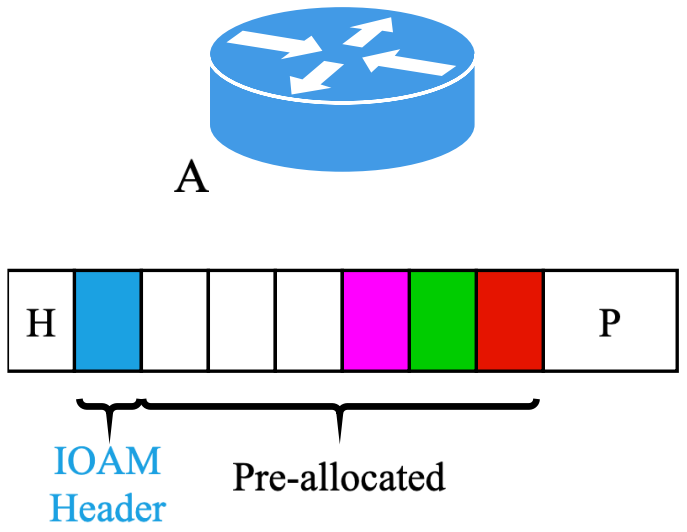
Before

Operations:
1. pre-allocate space
2. add IOAM plaintext data

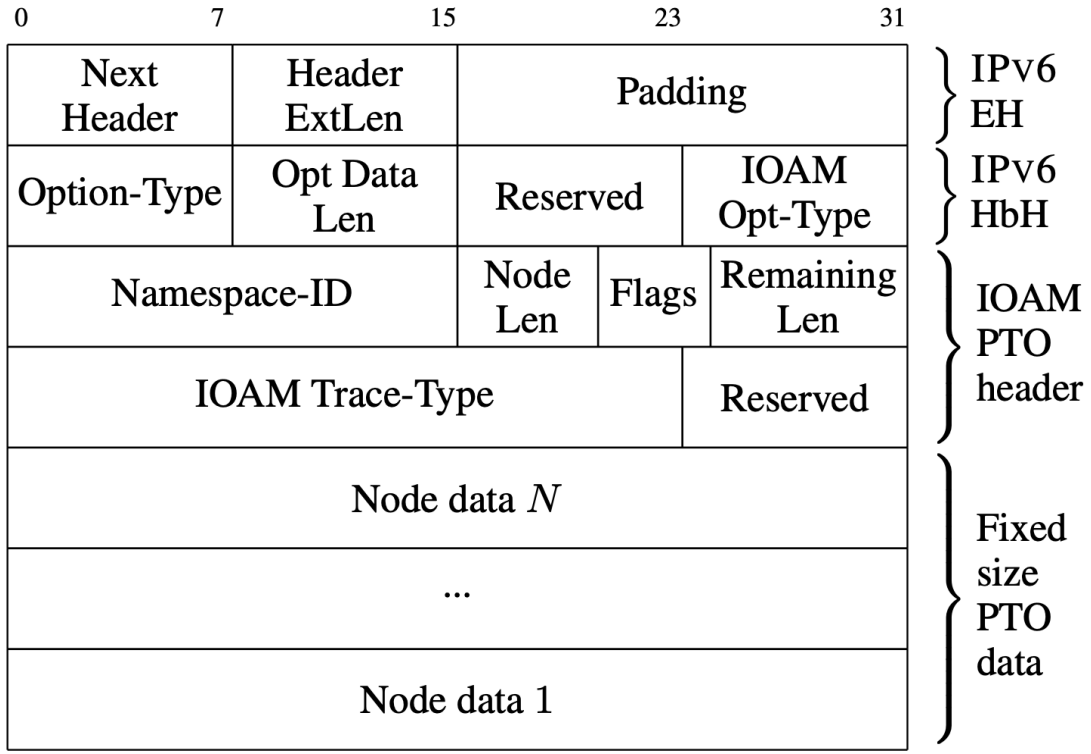


After

Operations:
1. pre-allocate space
2. add IOAM encrypted data
3. add nonce
4. add tag



IOAM – Hop-By-Hop Option Size



244 bytes for IOAM data:

- Variable payload + 12B nonce + 16B tag
- Maximum : 7 nodes with minimal payload

Multiplied by 4 :

- RemainingLen is multiple of 16B
- **Not** RFC compliant
- Variable payload + 12B nonce + 16B tag
- Minimal payload (4B) : 30 nodes
- Maximal payload (48B) : 12 nodes

Table of Contents

IOAM in the Linux Kernel

IOAM Pre-allocated Trace Option-type (PTO)

- Initial version at NetDev 0x14
- Mainline since v5.15
- Updates in subsequent kernel releases

Implementation

include/net/ioam6.h

```
struct ioam6_namespace {
        // existing fields

        atomic_t pkt_cnt;
        struct crypto_aead *aes;
        struct crypto_aead *chacha;
}

enum ioam6_sec_algo { AES_GCM, CHACHA };

struct crypto_aead *ioam6_aead(ns, algo);

int ioam6_sec_aead(...);

void ioam6_gather_data(...);
```

Implementation – Namespace Creation

```
static int ioam6_genl_addns(...) {
        // ...

        // both for AES-GCM and ChaCha20-Poly1305
        ns->aes = crypto_alloc_aead("gcm(aes)", 0, 0);
        err = crypto_aead_setkey(ns->aes, key, 32);
        err = crypto_aead_setauthsize(ns->aes, 16);

        // ...

        return 0;
}
```

Implementation – Change Key

```
$> ip ioam namespace set <NS_ID> key <KEY>
```

```
static int ioam6_genl_ns_set_key(..., struct genl_info *info) {
        u8 key[32] = {0};
        // ...

        // update existing transforms
        nla_memcpy(key, info->attrs[IOAM6_ATTR_KEY], 32);
        err = crypto_aead_setkey(ns->aes, key, 32);
        err = crypto_aead_setkey(ns->chacha, key, 32);

        // ...

        return 0;
}
```

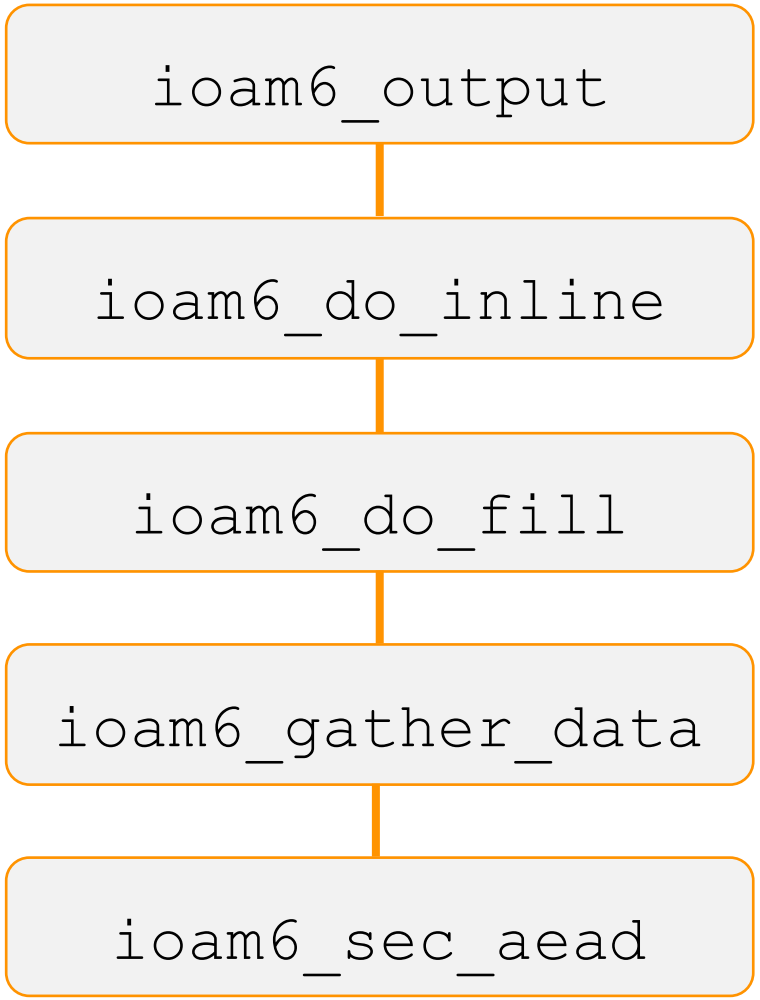
Implementation – Encapsulation

Tunnel Creation

```
static int ioam6_build_state(...) {
        // ...

        // len_aligned == size of IOAM data space
        if (! net->ipv6.sysctl.ioam6_encap_security)
                len_aligned = ALIGN(trace->remlen * 4, 8);
        else
                len_aligned = ALIGN(trace->remlen * 4 * 4, 8);

        // ...

        return 0;
}
```

Implementation – Encapsulation

Kernel Flow



Every packet leaving the machine

Insert IPv6 EH with HbH option

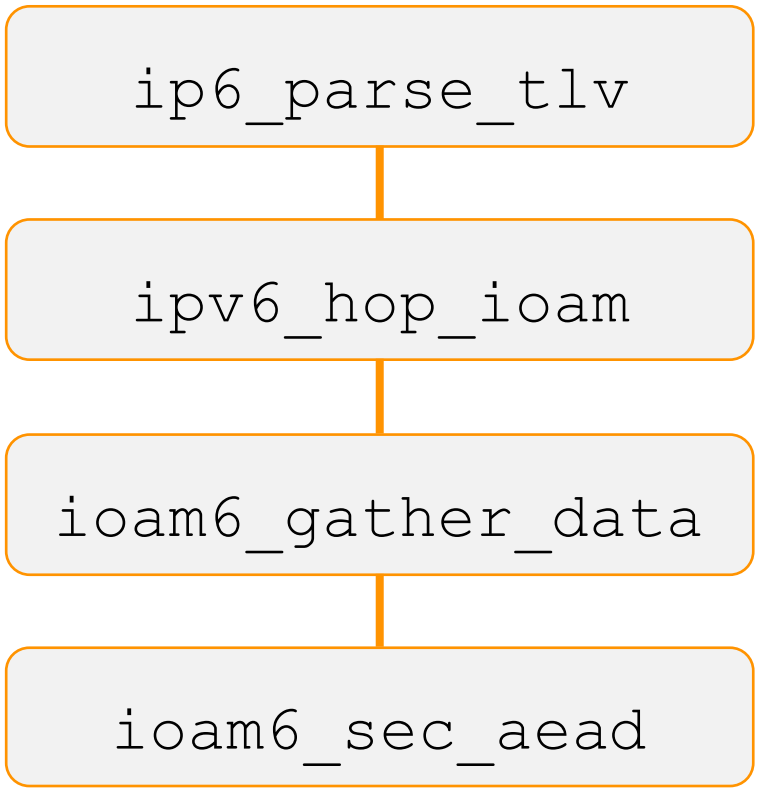
Fill HbH option with node data

Gather plaintext IOAM data

AEAD computation

Implementation – Transit

Kernel Flow



Parse TLV-encoded IPv6 HbH option

Process IOAM HbH option

Gather plaintext IOAM data

AEAD computation

Implementation – TLV Parsing

ip6_parse_tlv | ipv6_hop_ioam | ioam6_gather_data | ioam6_sec_aead

```
static bool ip6_parse_tlv(void) {
        // ...

        if (!ipv6_hop_ioam(skb, off))
                return false;

        // Size of data is * 4
        if(READ_ONCE(__in6_dev_get(skb->dev)->cnf.ioam6_sec_enabled)) {
                off+=3*(optlen-12);
                len-=3*(optlen-12);
        }

        // ...

}
```

Implementation – HbH Processing

ip6_parse_tlv | **ipv6_hop_ioam** | ioam6_gather_data | ioam6_sec_aead

```
static bool ipv6_hop_ioam(...) {
        // ...

        ioam6_gather_data(skb, ns, trace, true, payload);

        ioam6_sec_aead(CHACHA|AES_GCM, payload, nonce, tag, ...);

        // ...

        addr = memcpy(addr, nonce, 12) + 12;
        addr = memcpy(addr, tag, 16) + 16;
        memcpy(addr, payload, payload_len);

        // ...
}
```

Implementation – AEAD computation

ip6_parse_tlv | ipv6_hop_ioam | ioam6_gather_data | ioam6_sec_aead

```
int ioam6_sec_aead(...){
        struct crypto_aead tfm = ioam6_aead(...);
        struct aead_request req = aead_request_alloc(tfm, ...);

        struct scatterlist input[1], output[2];
        // initialization of src and dest scatterlist ...

        aead_request_set_ad(req, 0);
        aead_request_set_crypt(req, input, output, ..., nonce);

        int err = crypto_aead_encrypt(req);
        // error handling ...
        aead_request_free(req);
        return 0;
}
```

Usage

1) Enable protected IOAM data

```
$> sysctl -w net.ipv6.ioam6_encap_security=1
$> sysctl -w net.ipv6.conf.<iface>.ioam6_sec_enabled=1
```

2) Choose AES-GCM or ChaCha20-Poly1305

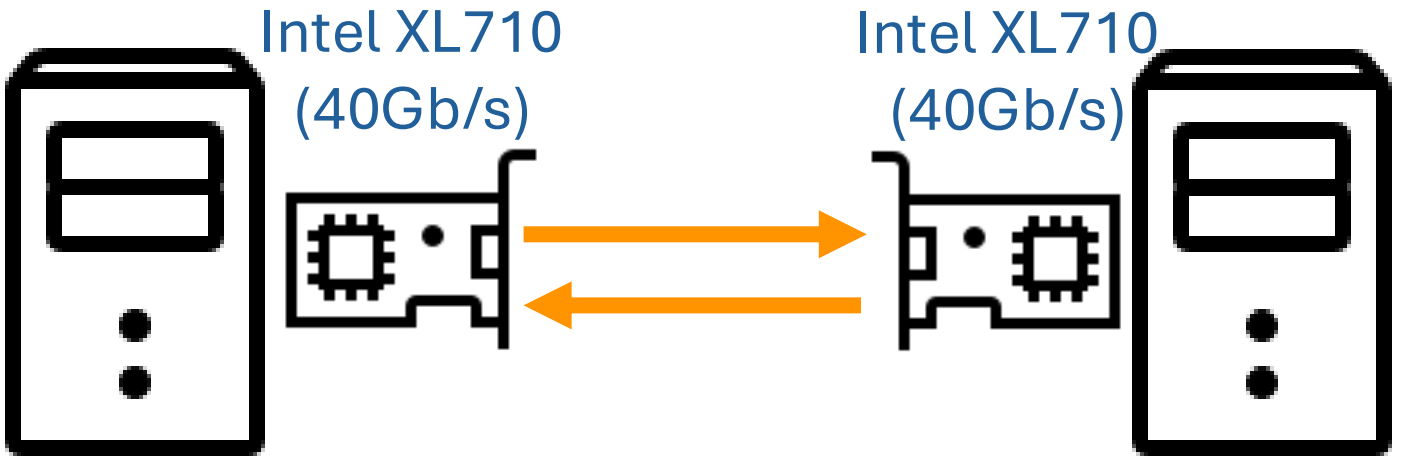
```
$> sysctl -w net.ipv6.ioam6_encap_security_chacha={0,1}
$> sysctl -w net.ipv6.conf.<iface>.ioam6_sec_chacha={0,1}
```

3) Configure secret key for given IOAM namespace

```
$> ip ioam namespace set <NS_ID> key <KEY>
```

Table of Contents

Performance Measurements



Traffic Generator
Cisco TRex on Linux 5.10

Device Under Test (DUT)
Patched Linux 6.18

2 Operations

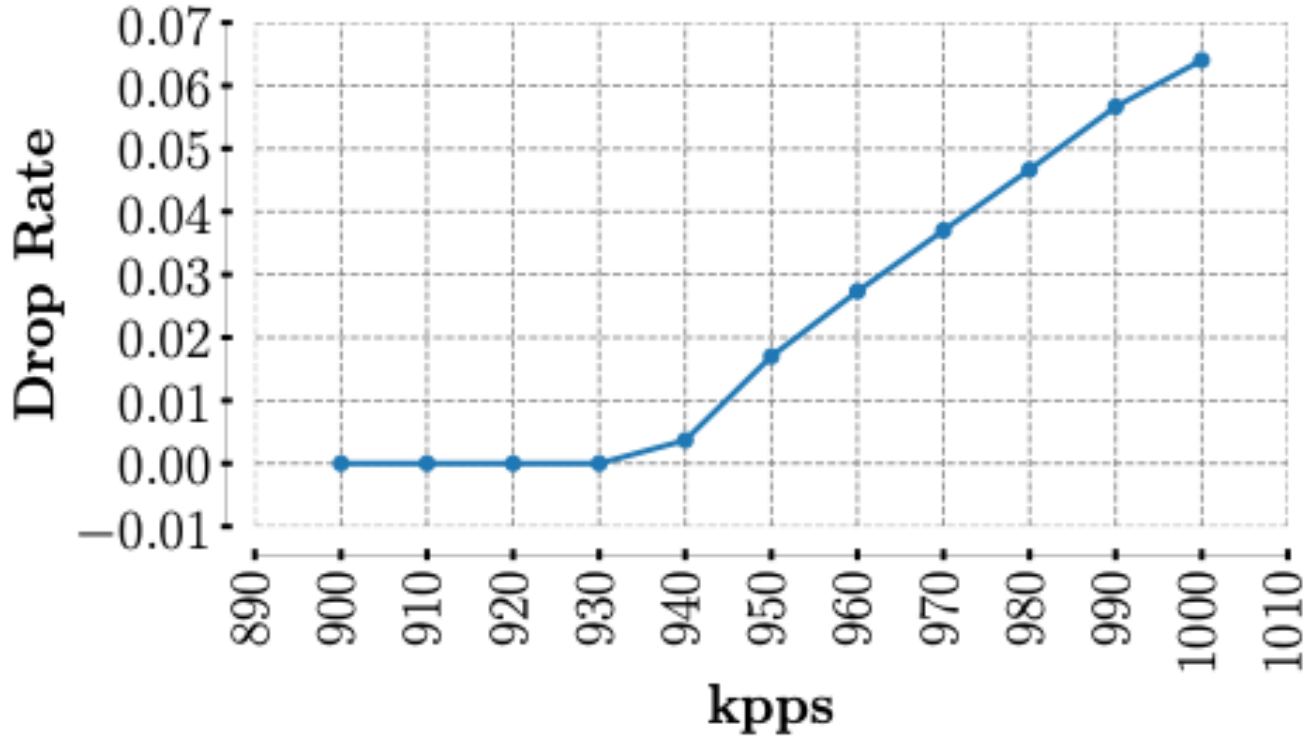
- Encapsulation
- Transit

Configuration on DUT

- Receiving on **single** queue
- Available on GitHub

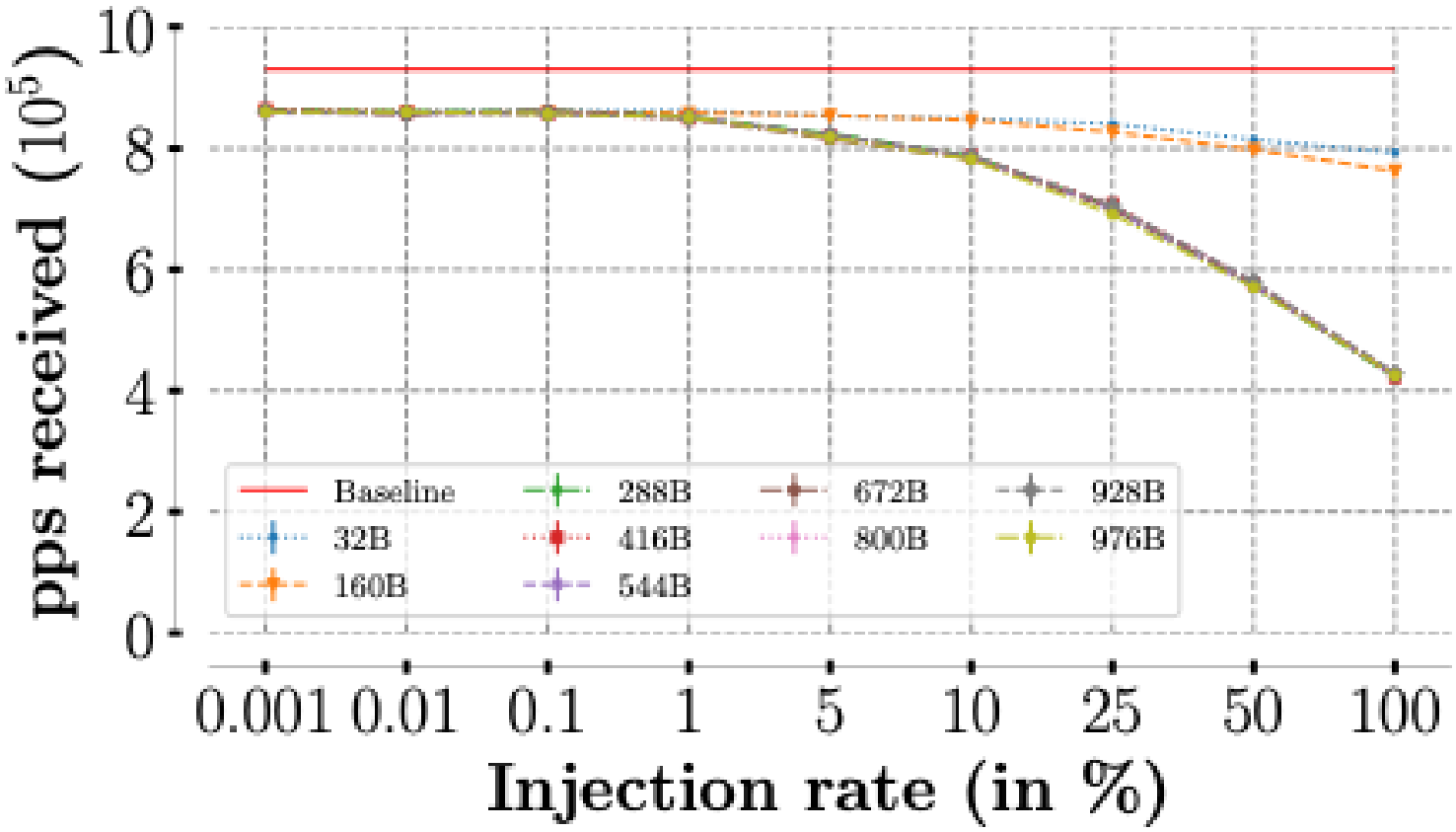
Performance – Baseline

1500-byte IPv6 Packet Forwarding on Single Core

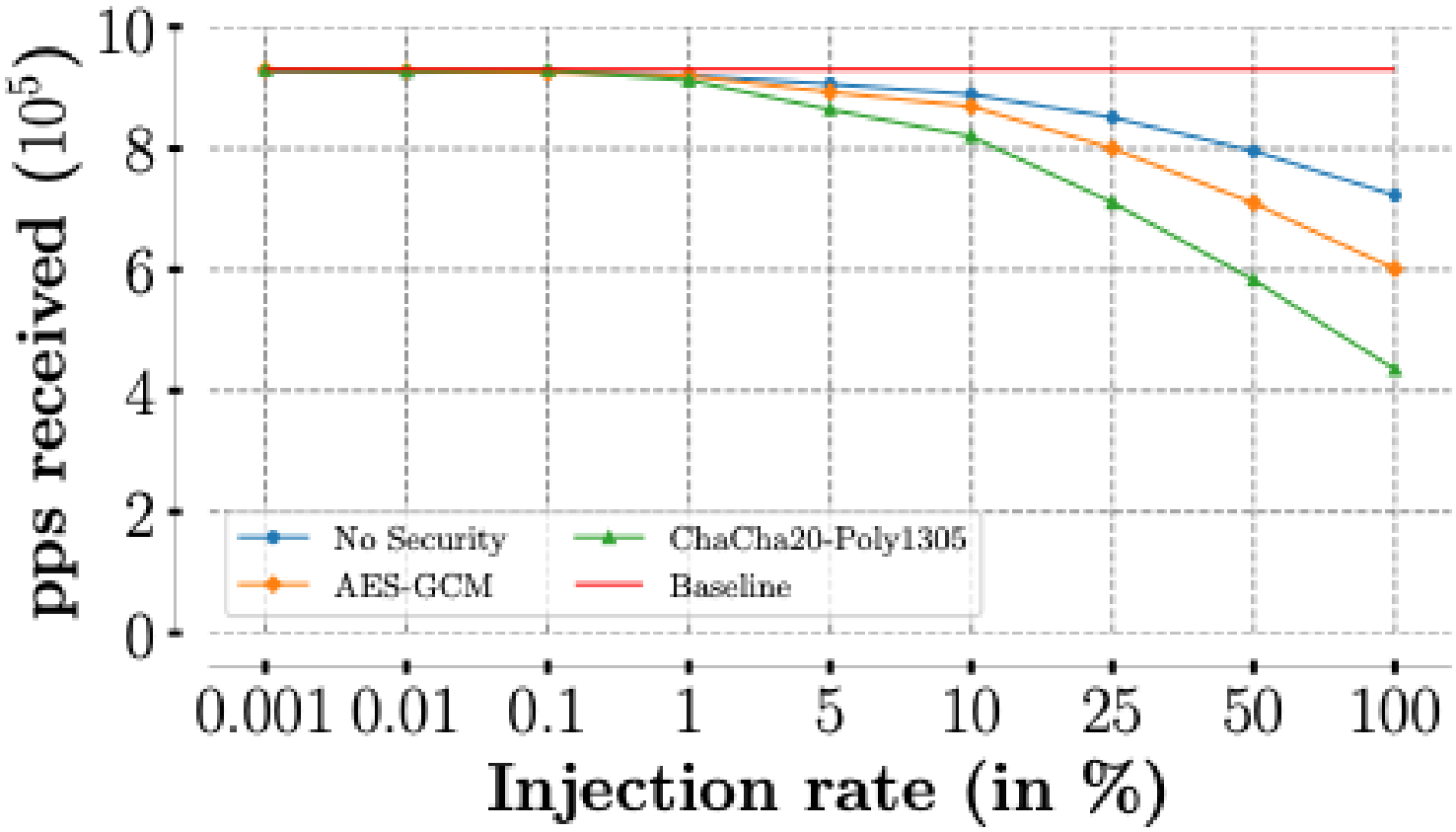


930,000 pps ≈ 11.16 Gbps

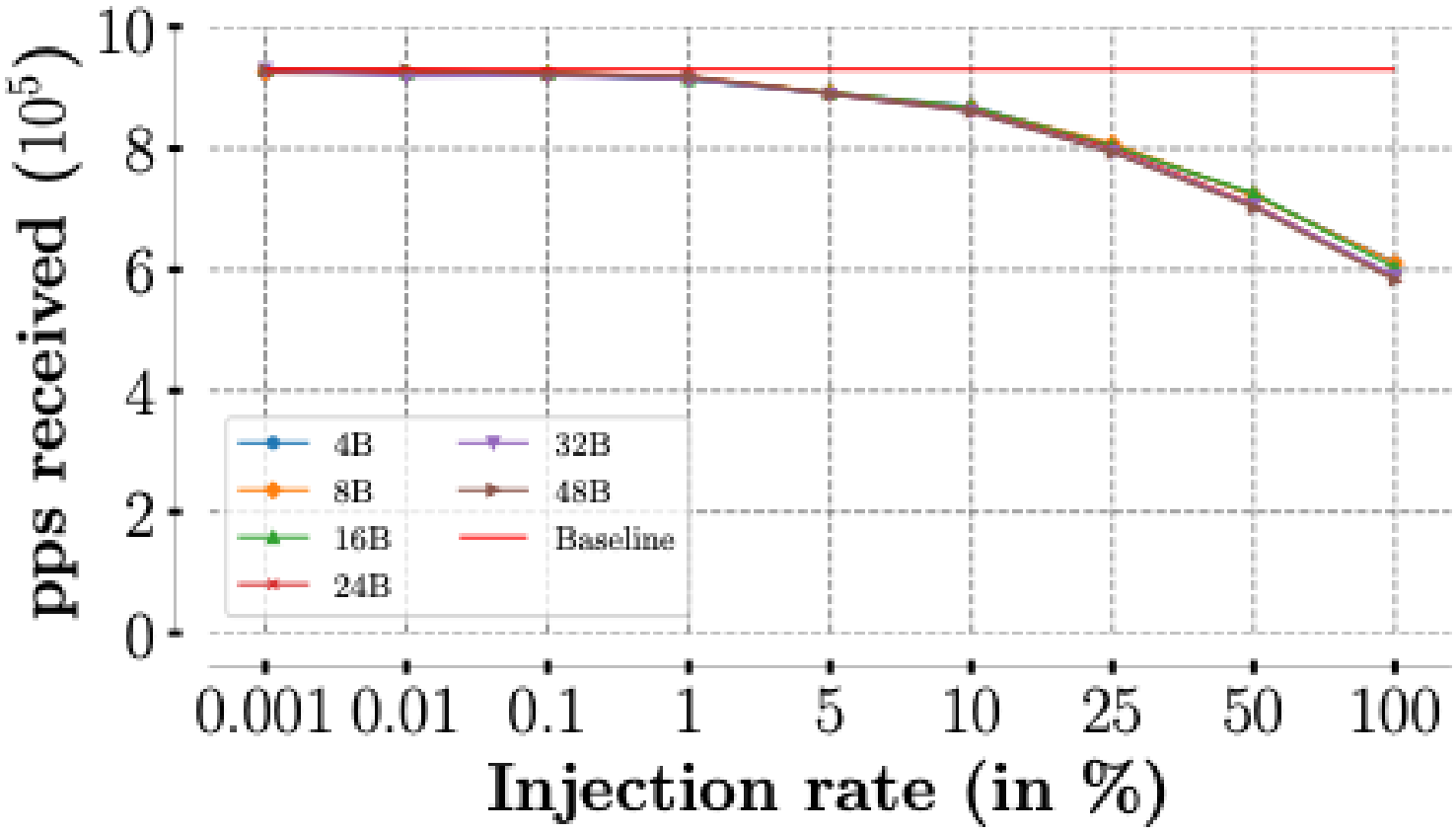
Performance – Encapsulation



Performance – Transit



Performance – Transit



Thank You For Listening

Do you have any question/feedback?

maxime.goffart@uliege.be