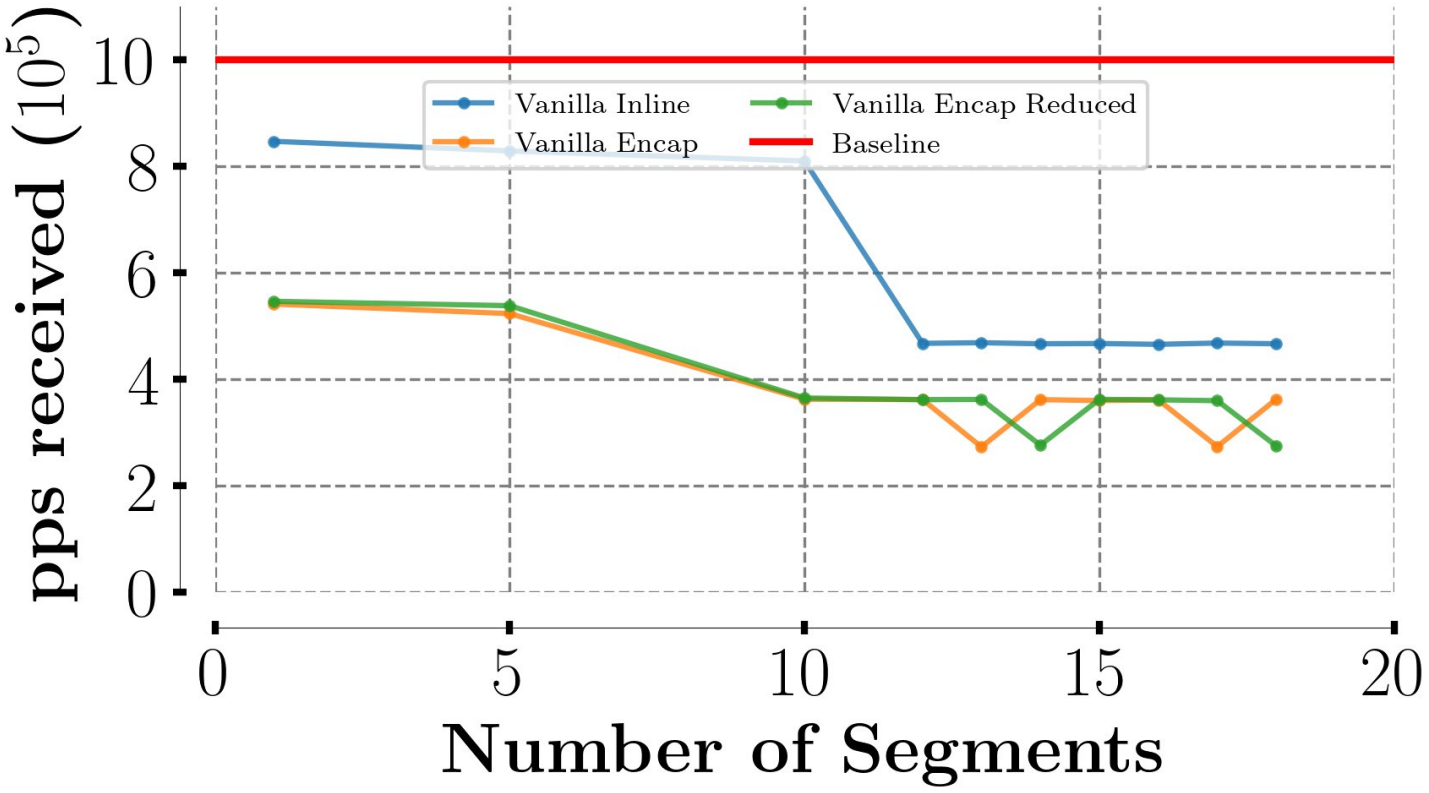
Mitigating the Double-Reallocation Issue for IPv6 Lightweight Tunnel Encapsulations

Emilien Wansart, Justin Iurman, Maxime Goffart, Benoit Donnet

NetDev 0x19 – Zagreb, Croatia – March 2025

Introduction

Affected protocols: SRv6, IOAM, and RPL



SRv6

- ± **23%** performance loss
- For segment counts
  - 13, 17, 21, (+4), ...
  - 14, 18, 22, (+4), ...

skb_cow_head function (1/2)

skb_cow_head makes sure skb has at least the specified headroom

```
static inline int skb_cow_head(
    struct sk_buff *skb,
    unsigned int headroom
);
```

- If enough headroom in skb → does nothing
- If not enough headroom → allocates new memory, value aligned to CPU cache line size
  - Typically 32B, 64B, or 128B depending on architecture

skb_cow_head function (2/2)

skb_cow_head makes sure skb has at least the specified headroom

```
static inline int skb_cow_head(
    struct sk_buff *skb,
    unsigned int headroom
);
```

Example

- skb has 64 bytes of headroom
- CPU with 32-byte cache line

After skb_cow_head(skb, 112), headroom is 128

- 64 + 2*32 = 128

Double reallocation issue (1/3)

- Issue found in LWT implementation of SRv6, IOAM, and RPL
- When adding new header to packet with insufficient headroom:
  - Normal case: 1 reallocation of the `skb`
  - 2-realloc issue: 2 consecutive reallocations of the `skb`
- Packet forwarding and locally generated packets affected
  - `seg6_input`, `seg6_output` functions
- Happens under specific conditions
  - Mainly depends on CPU architecture and NIC
- Results in performance degradation

Double reallocation issue (2/3)

The problematic code pattern:

```
err = skb_cow_head(skb, hdrlen + skb->mac_len);  (1)
// add the new header ("hdrlen" bytes)           (2)
// determine the output device "dev"             (3)
err = skb_cow_head(skb, LL_RESERVED_SPACE(dev)); (4)
```

- (1) ensures sufficient headroom for header insertion and anticipates Layer-2 header (`mac_len` is typically 14)
- (2) inserts new header in socket buffer
- (3) determines output device (depends on packet header)
- (4) ensures hardware header length alignment for output device, including any extra headroom for NIC

Double reallocation issue (3/3)

The problematic code pattern:

```
err = skb_cow_head(skb, hdrlen + skb->mac_len);   (1)
// add the new header ("hdrlen" bytes)            (2)
// determine the output device "dev"              (3)
err = skb_cow_head(skb, LL_RESERVED_SPACE(dev)); (4)
```

Example

- skb has 22 bytes of headroom
- A header of 40 bytes is added
- CPU with 32-byte cache line

- (1) headroom = 54 (22 + 32)
  - First reallocation occurs
- (2) headroom = 14 (54 - 40)
- (4) `LL_RESERVED_SPACE(dev)` = 16
  i40e driver needs 14 bytes, aligned to 16
  - Double reallocation occurs (14 < 16)

Double reallocation triggers

IOAM	
Inline mode	PTO of 236 or 240 bytes
Encap. mode	PTO of 196 or 200 bytes
SRv6	
Inline mode	None
Encap. mode	For 13, 17, 21, 25, 29, 33, .. segments
Encap. L2 mode	For 13, 17, 21, 25, 29, 33, .. segments
Encap. Red mode	For 14, 18, 22, 26, 30, 34, .. segments
Encap. L2 Red mode	For 14, 18, 22, 26, 30, 34, .. segments
RPL	
Inline mode	None

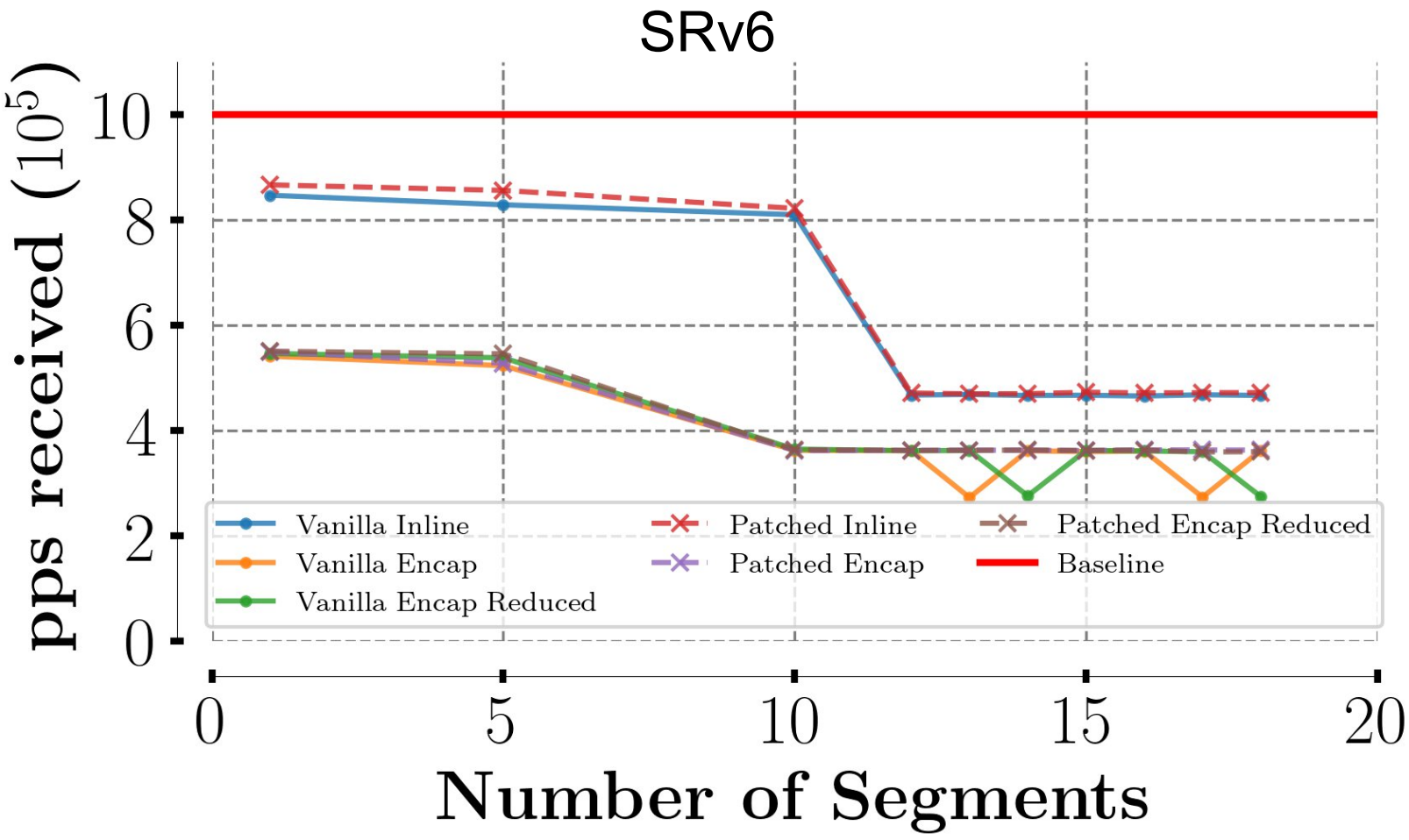
Mitigation

New inline function:

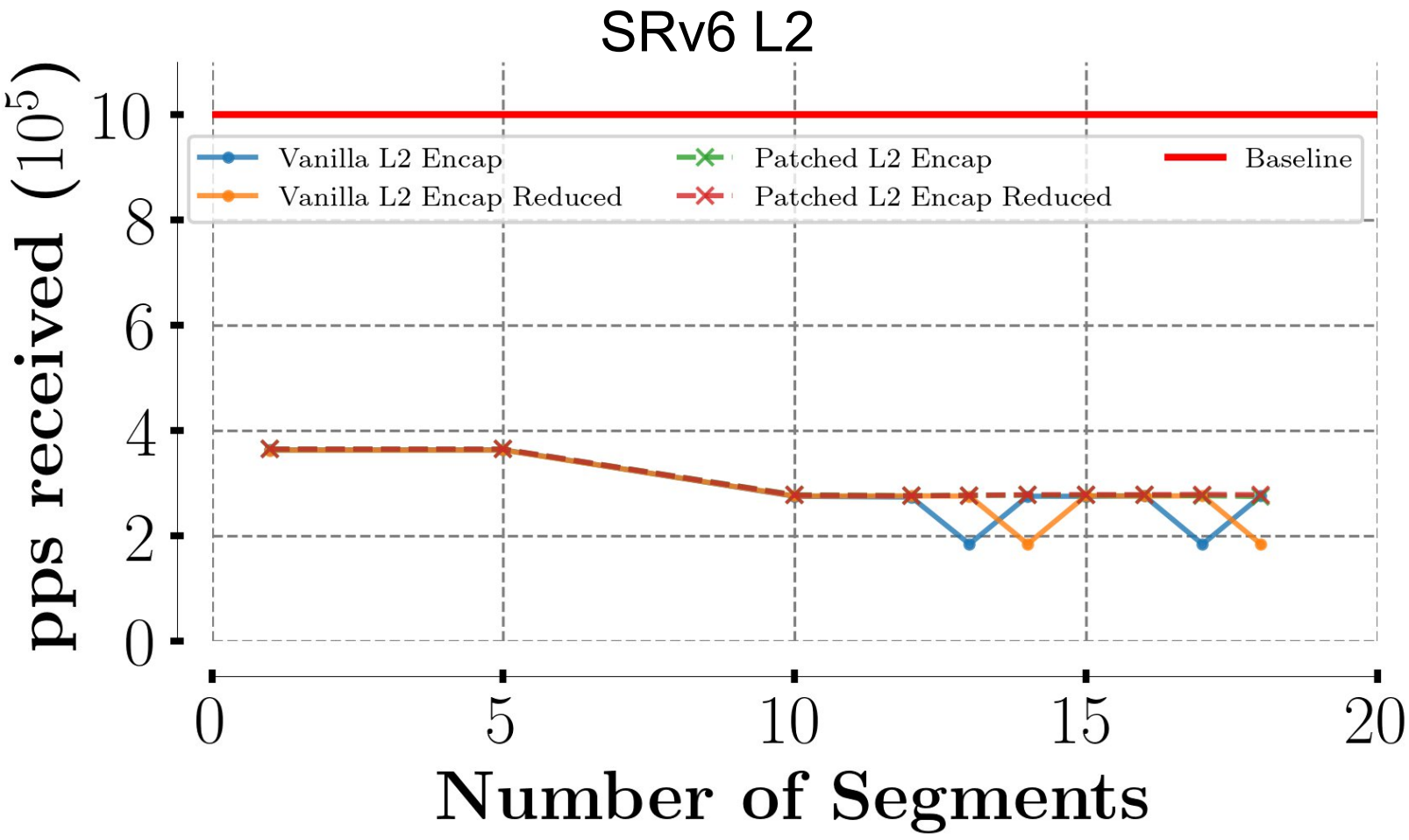
```
static inline unsigned int dst_dev_overhead(
    struct dst_entry *dst, struct sk_buff *skb)
{
    if (likely(dst))
        return LL_RESERVED_SPACE(dst->dev);
    return skb->mac_len;
}
```

- LWT caches the corresponding new `dst_entry`
- If cache is empty, use the generic `mac_len` value
  - 2-realloc issue persists for the first packet (negligible)

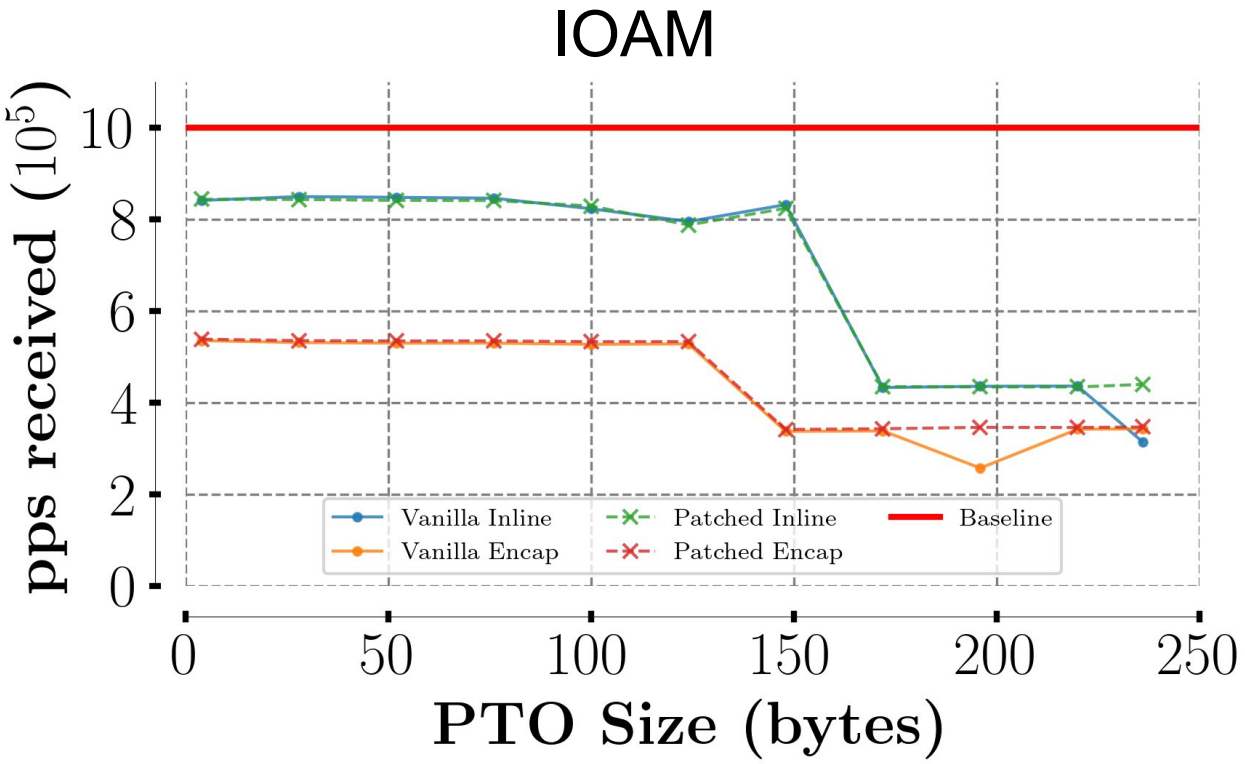
Performance results (1/3)

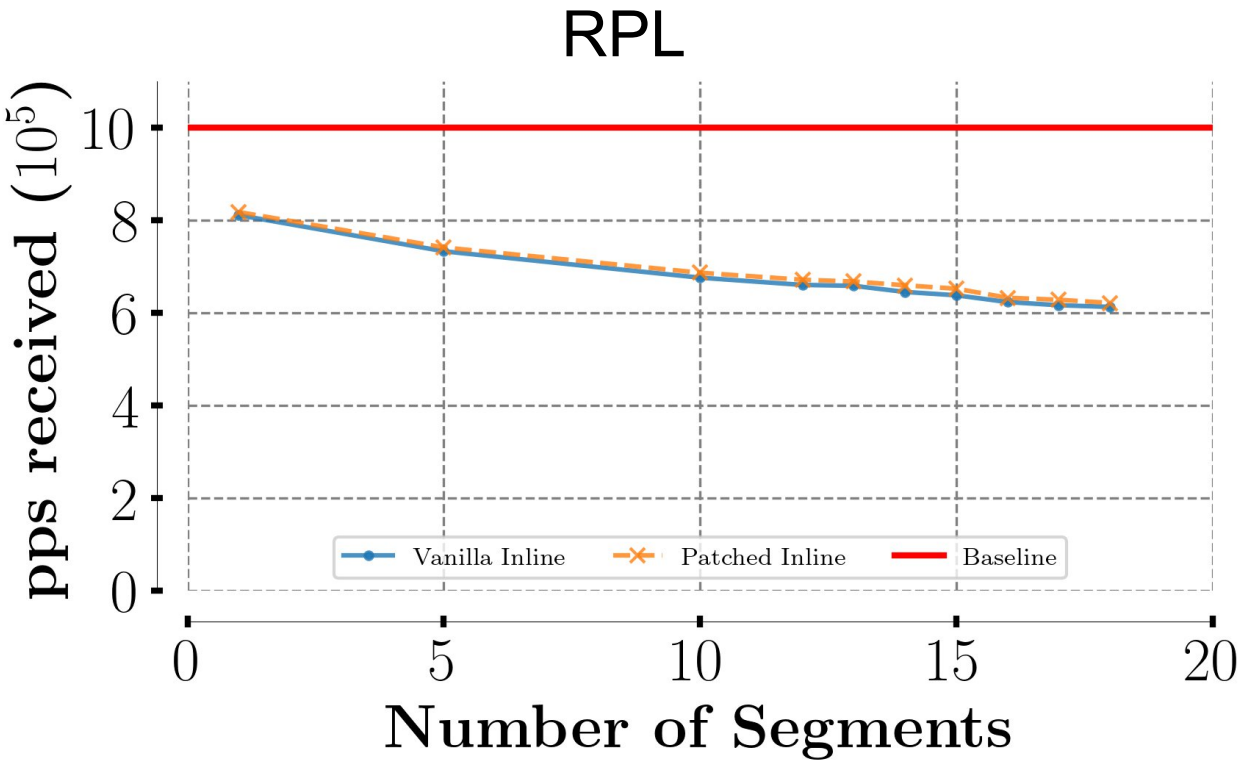


Performance results (2/3)



Performance results (3/3)





Conclusion

- Found issue in LWT implementation of SRv6, IOAM, and RPL
- Fixed 2-realloc issue using `dst_entry` of LWT for caching device
- Performance gain of 28.8% on average
- Patch merged into mainline Linux

Mailing list archive thread

Thank you for listening

Any questions?

Mailing list archive thread

emilien.wansart@uliege.be