SRv6 Tagging proxy

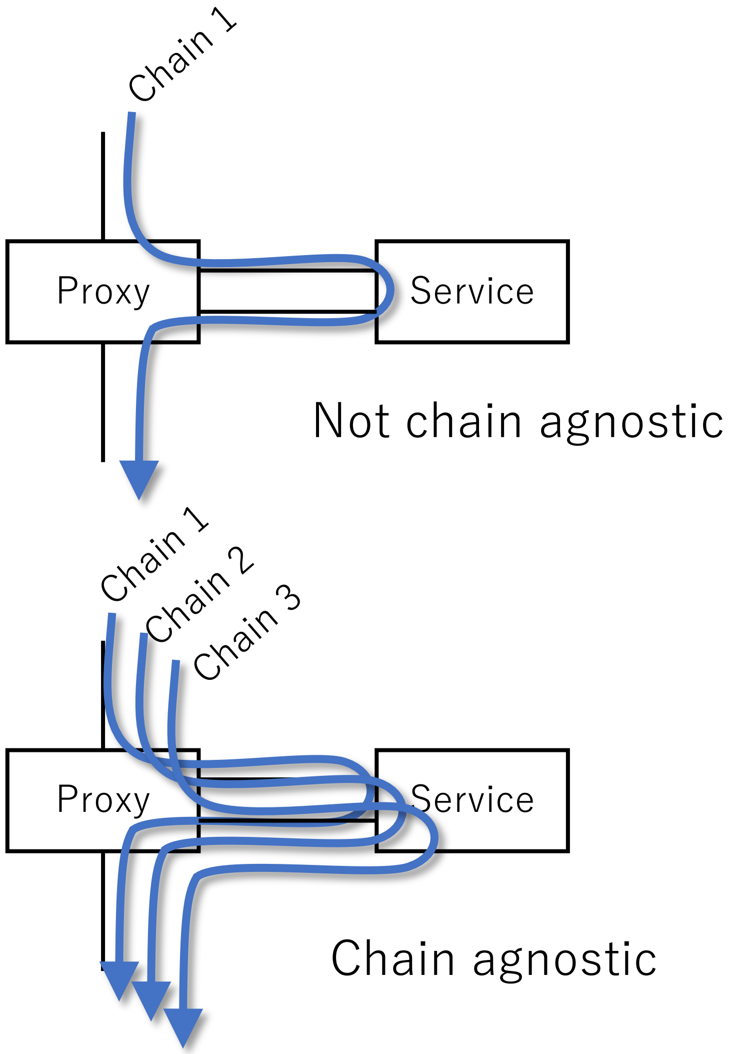
draft-eden-srv6-tagging-proxy-00

Yukito Ueno (NTT Communications),
Ryo Nakamura (The University of Tokyo),
Teppei Kamata (Cisco Systems)
Interop Tokyo 2019 ShowNet NOC team

SRv6 Proxy behaviors and multiplexing

2 behavior types of SRv6 proxy

- End.AS, End.AD
  - Not chain agnostic
  - Proxy:Chain = 1:1
- End.AM
  - Chain agnostic
  - Proxy:Chain = 1:N
  - SRv6 insertion only



To deploy practical service chain

From experience of ShowNet in Interop Tokyo, we needed:

- SRv6 proxies for most middle boxes
- Dedicated chains for each user resulting in 200-300 chains
- Both IPv4 and IPv6 connectivity for each user

SRv6 proxy which can achieve the above requirements is needed but:

- End.AS: Not chain agnostic
- End.AD: Not chain agnostic
- End.AM: SRv6 insertion only

Interop Tokyo 6/12(Wed),13(Thu),14(Fri) MAKUHARI MESSE [CHIBA JAPAN]

SHOWNET EVOLVE INTO THE NEXT GENERATION

End.AT (SRv6 Tagging*1 proxy)

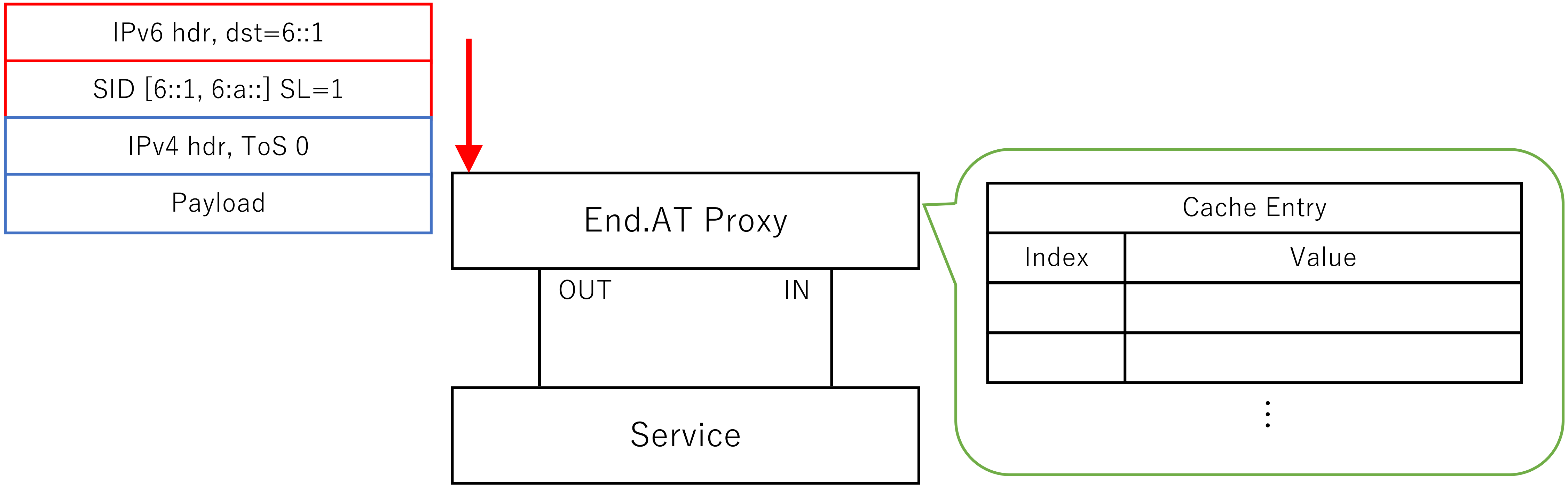
We proposed a new type of SRv6 proxy which supports:

- Chain agnostic configuration
- Transparency to chain changes
- Both IPv4 and IPv6 in SRv6 encapsulation mode

*1 "Tagging" means multiplexing chains by embedding a cache index into the inner packet header as a tag.

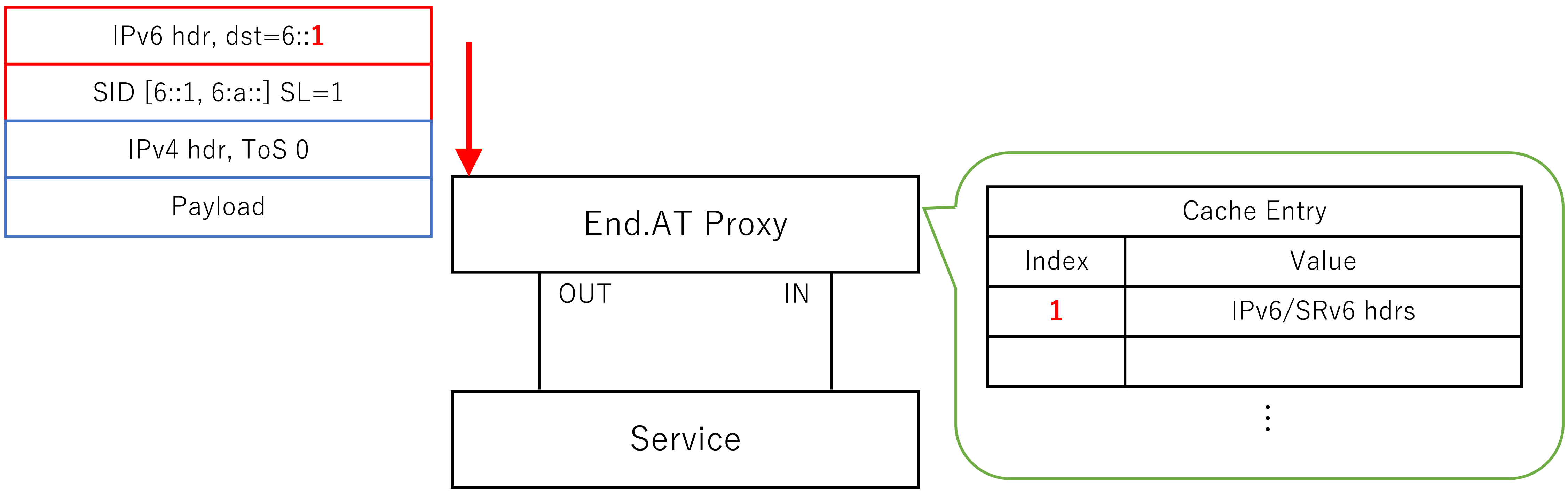
End.AT in detail

An IPv4 in SRv6 packet comes



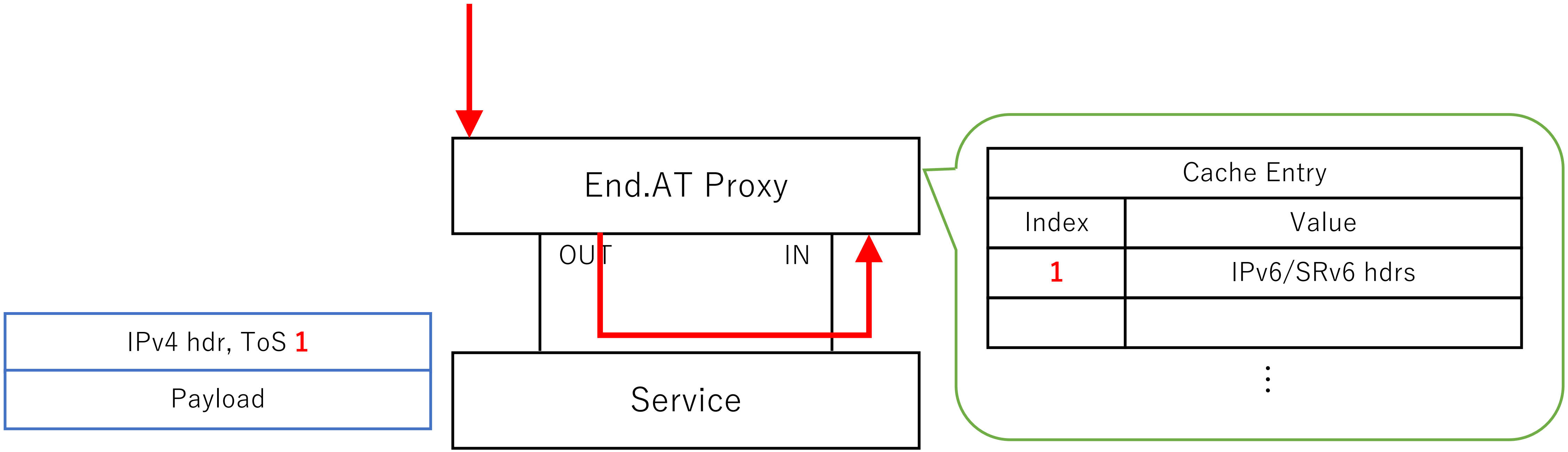
End.AT in detail

Store the outer IPv6 and SR header into a cache entry indexed by the argument field of SRv6 SID



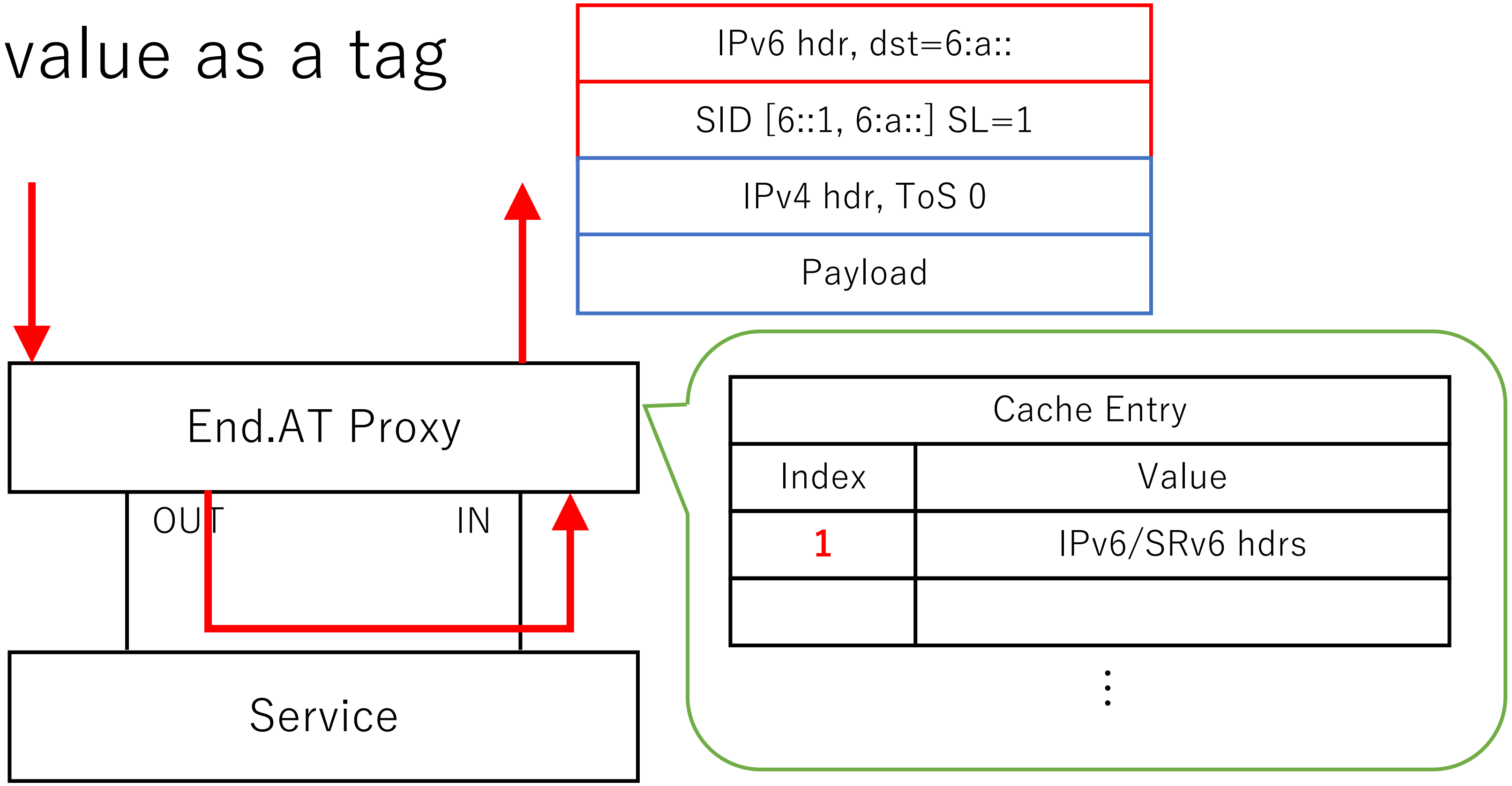
End.AT in detail

Embed argument value into the ToS or TC field of inner IP header and transmit the decapsulated inner packet



End.AT in detail

Restore the outer IPv6 and SR header from the cache by using ToS or TC value as a tag



Characteristics

- Chain agnostic
  - Up to 256 paths (because of 8bit ToS field)
- Transparent to chain changes
- Stateless
  - No cache expiration mechanism needed (overwritten every time packet comes)

} Advantage

- Incompatible with QoS
  - Due to exploiting ToS or TC field
  - In need of transparency of service functions

} Disadvantage

Implementation status

- Linux AF_XDP implementation
  - https://github.com/edenden/end.ac
  - Used for all IPv4 traffic in ShowNet 2019
- Linux kernel implementation (out-of-tree)
  - https://github.com/upa/linux/tree/seg6-shownet
  - Tested by IXIA and Spirent testers in ShowNet 2019

Q & A