

IPv6 Support for Segment Routing: SRv6+

draft-bonica-spring-srv6-plus-04

R. Bonica, S. Hegde, Y. Kamite, A. Alston, D. Henriques, J. Halpern, J.
Linkova, G. Chen

What Problem Are We Solving?

- Implement the Segment Routing architecture
 - Encode path state in each packet
 - So that transit routers do not need to maintain per path state
- Implement programmable SR paths
- Rely exclusively upon IPv6 data plane
 - No MPLS
 - *Leverage existing IPv6 features*
- Minimize SR overhead
 - Bandwidth resources
 - ASIC resources

Terminology

Paths

- Provide unidirectional connectivity from ingress node to egress node
- Can follow any route through the network
 - Least cost
 - Traffic engineered
- Contain one or more segments
- *Programmable*
- Defined by the segments that they contain

Segments

- Provide unidirectional connectivity from ingress node to egress node
- *Programmable*
- Behavior is controlled by a topological instruction
 - Executed on segment ingress node
 - Defines egress node
 - Defines method by which ingress node forwards packets to egress node
- Defined by ingress node and topological instruction
- Can be contained by multiple paths

Exactly Two Segment Types

- Strictly-Routed
 - Similar to adjacency segment
 - Topological instruction causes ingress node to forward packets through a specified interface to the egress node
- Loosely-Routed
 - Similar to node segment
 - Topological instruction causes ingress node to forward packets through the least cost path to the egress node

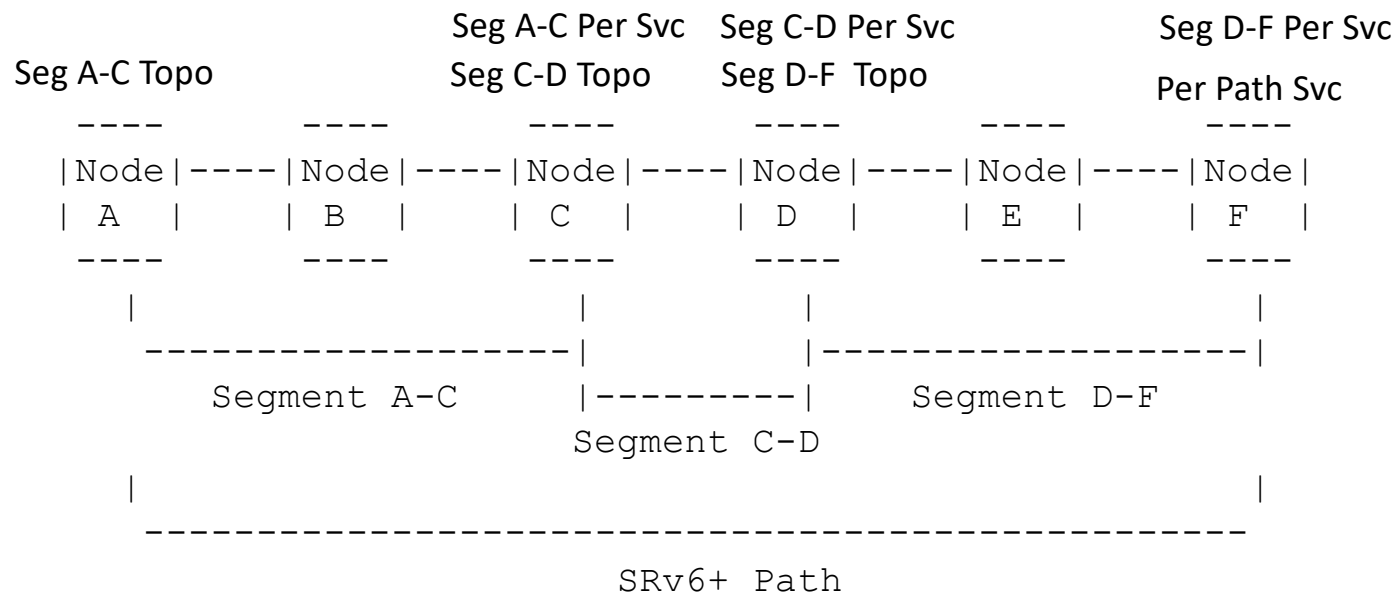
Segment Identifiers (SID)

- Identify a segment
 - Because there is a one-to-one relationship between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it
- Identifies, but does not contain a topological instruction
 - Therefore, can be encoded in relatively few bits
 - 16 and 32 bit options
- Node-local significance
 - Only processed by one node
 - To facilitate debugging, SIDs can be assigned in a manner that gives them domain-wide significance

Service Instructions

- Augment, but do not define a path or segment
- Per-Segment Service Instructions
 - Executed on segment egress node
 - Examples
 - Expose a packet to a firewall policy
 - Expose a packet to a sampling policy
- Per-Path Service Instructions
 - Executed on the path egress node
 - Examples
 - De-encapsulate a packet and forward the payload over a specified VPN link
 - De-encapsulate a packet and forward the payload using a specified routing table

Paths, Segments and Instructions



Encoding SRv6+ Paths as IPv6 Header Chains

The IPv6 Extension Header Chain

- IPv6 source nodes encode additional internet-layer information in extension headers
- RFC 8200 defines a small number of extension headers
- The IPv6 header and each extension header contain a Next Header field
 - So, extension headers can be chained together
- Extension headers are processed in the order that they appear in a packet
- RFC 8200 specifies an order in which extension headers should appear

Extension Header Ordering

- Processed by every hop along the path from source to destination
 - Hop-by-hop
- Processed by segment endpoints only
 - Destination Options (preceding Routing header)
 - Routing header
- Processed by ultimate destination only
 - Fragment
 - Authentication
 - Encapsulated Security Payload (ESP)
 - Destination Option (preceding upper-layer header)

Routing Header Defines Segmented Path

- Routing header contains (among other things)
 - Segment List – List of segment endpoints to be traversed on route to destination
 - Segments Left – Number of segments still to be traversed
- Ignore if Segments Left equals zero
- Process if Segments Left is greater than zero
 - Decrement Segments Left
 - Overwrite IPv6 Destination address with address derived from Segment List member referenced by Segments Left
 - Forward

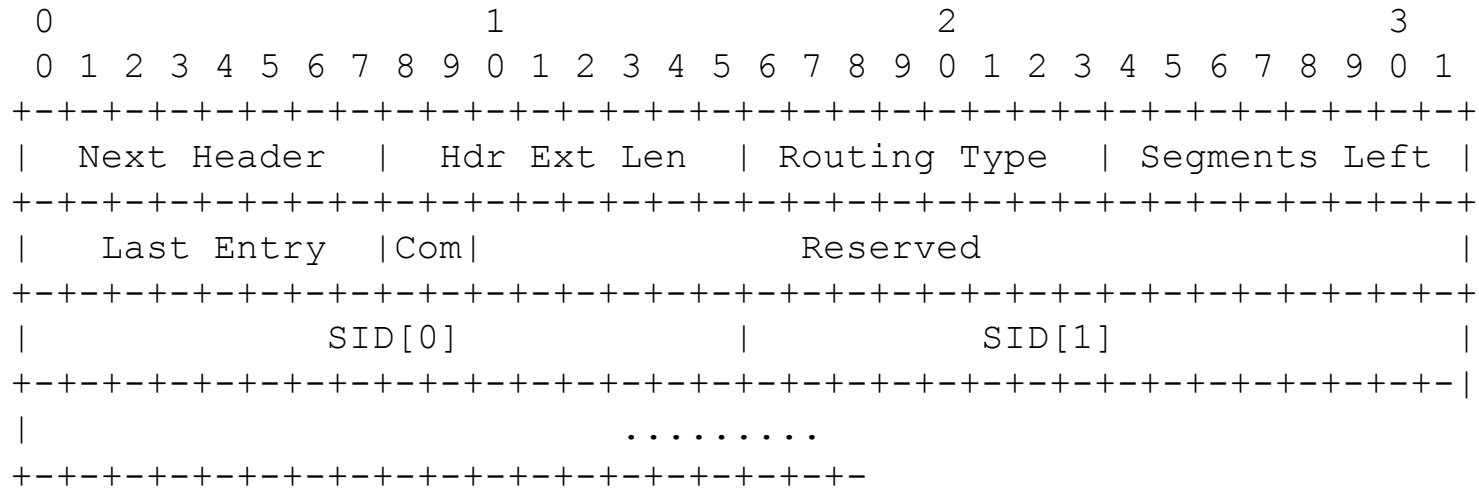
Routing Header Barrier To Deployment

- All Routing headers contain 8 bytes of overhead
- Most Routing headers represent segments as 16 bytes
- So, a Routing header that represents a 5 segment path contains 72 byte
- Large Routing headers
 - Consume bandwidth
 - Are ASIC unfriendly

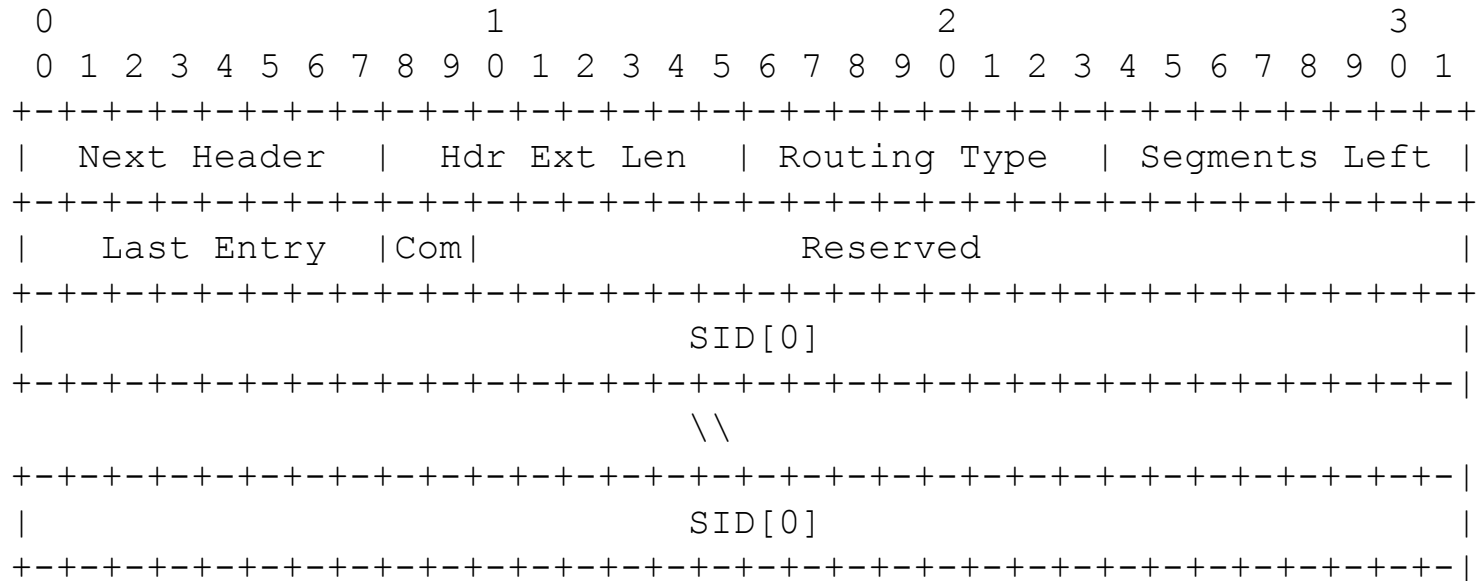
Compressed Routing Header

- SRv6+ defines a new Routing header type, called the Compressed Routing Header (CRH)
- Encodes Segment Identifiers (SID) in 16 or 32 bits
- SID Forwarding Information Base (SFIB) translates SID into
 - An IPv6 address to be copied into the IPv6 Destination Address field
 - An instruction that determines how the packet will be forwarded to the destination (strict or loose forwarding)
- Yes, computer science problems can be solved with one more layer of indirection

CRH With 16-Bit Encoding



CRH With 32-Bit Encoding



Encoding Service Instructions in Destination Options

- Per Segment Service Instructions
 - SRv6+ defines a new option, call the Per Segment Service Instruction Option
 - Encoded in Destination Option header that precedes the Routing header
 - Carries a 32-bit instruction identifier
 - Skip when unrecognized – first two bits of option identifier are 00
- Per Path Service Instructions
 - SRv6+ defines a new option, call the Per Path Service Instruction Option
 - Encoded in Destination Option header that precedes the upper-layer header
 - Carries a 32-bit instruction identifier
 - Discard packet when unrecognized – first to bits of option identifier are 10

Related Work

- Draft-bonica-6man-comp-rtg-hdr
- Draft-bonica-6man-vpn-dest-opt
- Draft-bonica-6man-seg-end-opt
- Draft-bonica-lsr-crh-isis-extensions
- Draft-sangli-idr-vpn-service-srv6-plus
- Draft-alson-spring-crh-bgp-signalling

Implementation

- JUNOS PoC
- LINUX Demo

Next Steps

- SPRING WG to adopt draft-bonica-spring-srv6-plus
- 6man WG to adopt
 - Draft-bonica-6man-com-rtg-hdr
 - Draft-bonica-6man-vpn-dest-opt
 - Draft-bonica-6man-seg-end-opt