Using Entropy Label for Network Slice Identification

draft-decraene-mpls-slid-encoded-entropy-label-id-01

Bruno Decraene (Orange)
Clarence Filsfils (Cisco)
Wim Henderickx (Nokia)
Tarek Saad (Juniper)
Vishnu Pavan Beeram (Juniper)
Luay Jalil (Verizon)
Conflicting Wishes?

- Advertisement of new fields in the MPLS stack.
  - E.g., Slice ID, flags

- No increase in the size of the stack/header
  - In particular for SR-MPLS
Entropy Label

• Defined in RFC 6790, used for load-balancing.
• Uses a stack of 2 labels:
  – Entropy Label Indicator (Special-Purpose Label 7)
  – Entropy Label (a per-flow entropy value)
Entropy Label

While keeping its usage for Entropy and backward compatibility, there is some freedom the use of some fields:

- Entropy Label: freely chosen by the ingress for a given flow.
- EL’s TTL: MBZ, not read
  - i.e. same semantic as “reserved” (for future use)
Extension #1: Entropy Label Control field

- Redefine the unused EL’s TTL as the “Entropy Label Control” field.
  - Set of 8 flags: ELC0…ELC7

- Semantic of the bits are user defined (i.e. not standardized)
  - To maximize the reusability of this scarce resource in the MPLS header
  - An application using a flag MUST have it configurable
Extension #2: SLice IDentifier (SLID)

- As a use case of extension #1
  - One of the ELCk flag is set to 1
- The slice ID is encoded in a portion of the Entropy Label
  - In the most significant bits.
  - Size is user defined. Must be consistent in the domain.
SLID: Ingress LSR

• Push the Entropy Label:
  • MSB: Slide ID
  • LSB: entropy information as defined in RFC6790 section 4.2

• Sets the SLID Presence indicator
  • One flag of the EL Control field
  • Indicates the presence of the Slide ID, hence that the packet belong to a slice.
SLID: Transit LSR

- No change for load balancing
  - EL treated as an opaque field
  - Backward compatible
- If SLID Presence indicator set
  - Read the slice ID
  - Slicing behavior is out of scope of this spec.
Benefits

• Backward compatible with EL routers
• New feature in the *existing* shim header stack
  • No increase in the stack/header size
  • Reuses EL signaling (capability, MSD)
  • Incremental deployment with incremental benefits
    as egress LSR already supports EL
Feedback welcomed