Advertising Per-Topology and Per-Algorithm Label Blocks

draft-bowers-spring-adv-per-algorithm-label-blocks-02

Chris Bowers cbowers@juniper.net
Pushpasis Sarkar psarkar@juniper.net
Hannes Gredler hannes@juniper.net
Uma Chunduri uma.chunduri@ericsson.com

SPRING Working Group
IETF94 Yokohama

Computing locally significant labels for shortest path next-hops

SPF_Label(X,D) = Label_Block(X) + Node_Index(D)

- Label_Block(X) is the label block advertised by X
- D is the destination node
- SPF_Label(X,D) is the value of the label that neighbors need to apply to a packet so that X will forward the packet along the shortest path next-hop to D.

Computing locally significant labels for next-hops corresponding to other topologies and algorithms

Option 1: per-topology / per-algorithm node index

Label(X,D,T,A) = Label_Block(X) + Node_Index(D,T,A)

Option 2: per-topology / per-algorithm label block

Label(X,D,T,A) = Label_Block(X,T,A) + Node_Index(D)

- T is the topology
- A is the algorithm for computing destination-based forwarding next-hops
- D is the destination node
- X is the next hop along the path to D that is determined by algorithm A for topology T
- Label(X,D,T A) is the value of the label that neighbors need to apply to a packet so that X will forward the packet along the next-hop to D determined by algorithm A for topology T.

Option 2: per-topology / per-algorithm label block

label blocks

100-109	0	1	2	3	4	5	6			
110-119	0	1	2	3	4	5	6			
120-129	0	1	2	3	4	5	6			
130-139	0	1	2	3	4	5	6			
140-149										

With option 2, label values must be logically related to the topology/algorithm pair and the node SID.

Option 1: per-topology / per-algorithm node index

label block

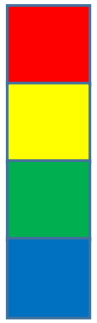
100-149	0	1	2	3	4	5	6			
	0		2	4		5			2	
	0	1	3	5	4	3				5
	1		6		4		6			
		1	2	3	6		0			

T=0,A=0

T=0,A=4

T=0,A=5

T=7,A=1



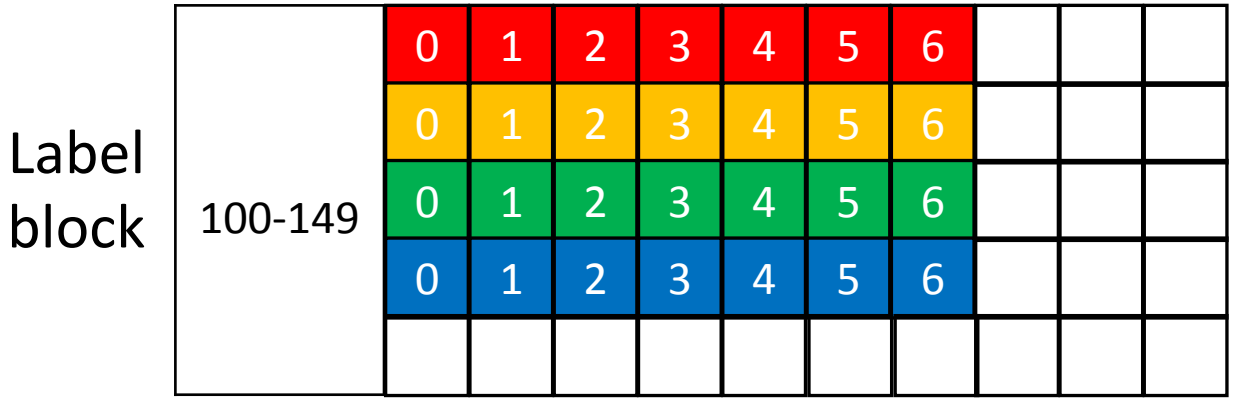
With option1, label values do not need to have any logical relationship to the topology/algorithm pair and the default topology/algorithm node SID.

node#

#

Organization of label values needs to be imposed externally.

Option 1: per-topology / per-algorithm node index plus the configured offset mapping method



Config on node#3
base_node_index=3

label_block_size=50
topology=0 algorithm=0 offset=0
topology=0 algorithm=4 offset=10
topology=0 algorithm=5 offset=20
topology=7 algorithm=1 offset=30

Config on node#4
base_node_index=4

label_block_size=50
topology=0 algorithm=0 offset=0
topology=0 algorithm=4 offset=10
topology=0 algorithm=5 offset=20
topology=7 algorithm=1 offset=30

Config on node#4
base_node_index=5

label_block_size=50
topology=0 algorithm=0 offset=0
topology=0 algorithm=4 offset=10
topology=0 algorithm=5 offset=20
topology=7 algorithm=1 offset=30

Identically configured offset mappings via CLI (or management interface) can be used to impose order on the label assignment

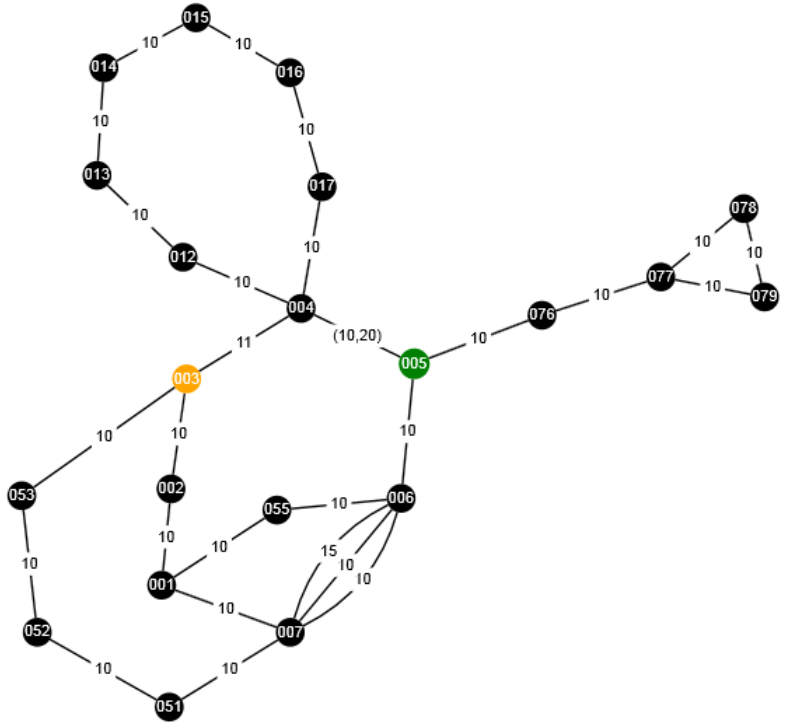
Why is option 2 (per-topo/per-algo label blocks) better than option 1 (per-topo/per-algo node indexes)?

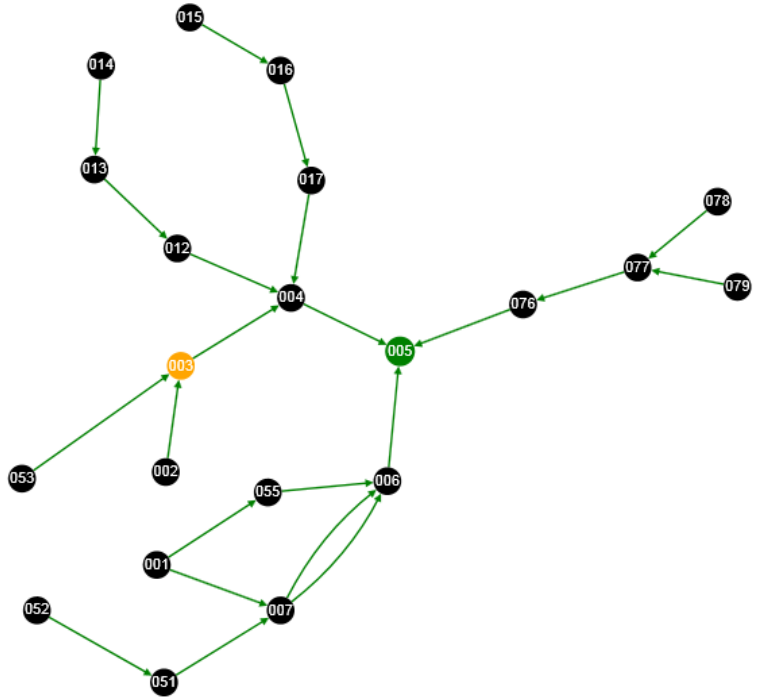
- Using SR to distribute labels for shortest path routes
- Advantage over LDP that any node (not just neighbors of X) can determine the FEC-label bindings distributed by node X.
  - **Good thing**: no need for targeted LDP sessions
  - **Bad thing** : need to assign and maintain tightly packed, domain-unique node index values

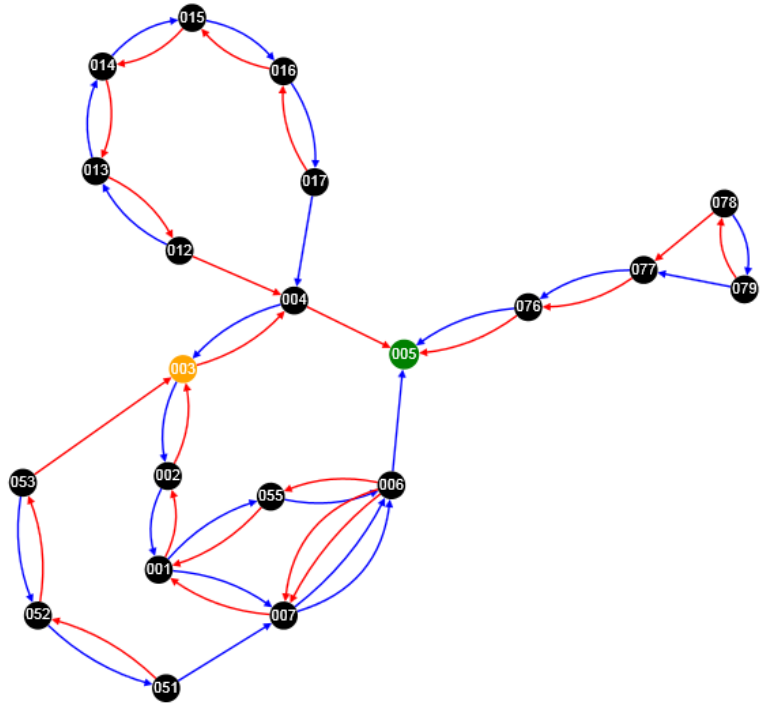
- Generalizing SR to distribute labels for other topologies and algorithms
  - Option 1 allows the **Bad Thing** to propagate.
    - Configured offset mappings required to manage the badness
    - Extra configuration adding no value and error prone
  - Option 2 puts the **Bad Thing** in a box.

Algorithms and topologies:

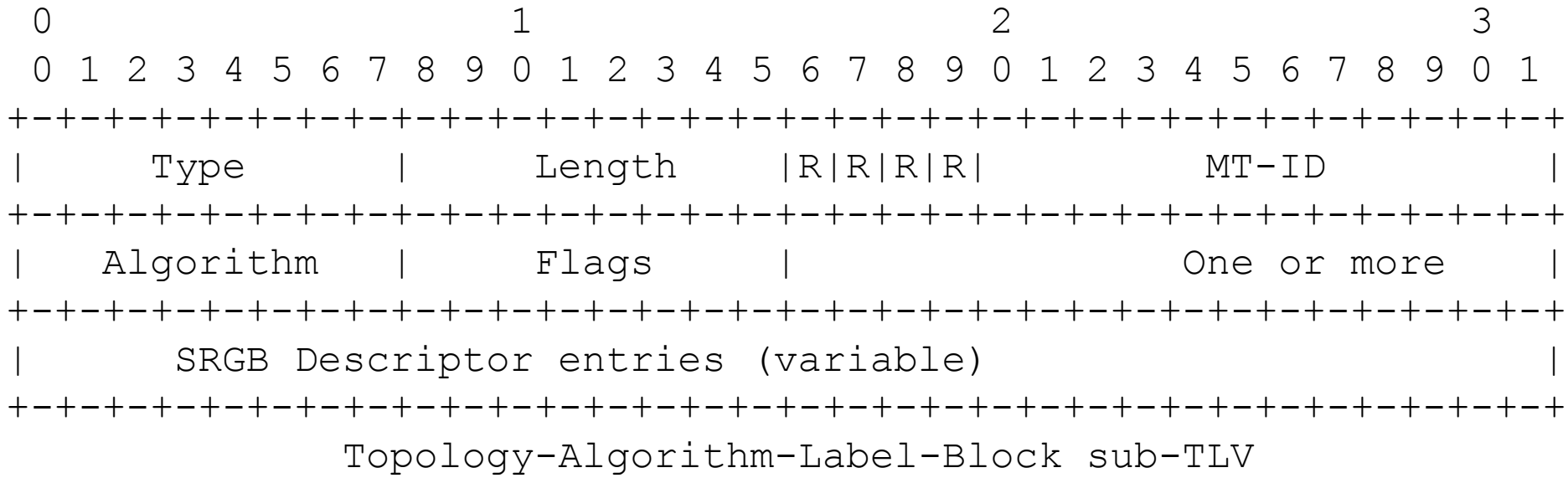
- Simple example in draft uses shortest path based on latency as alternative algorithm.
- SR algorithm mechanism seems like an elegant method for distributing labels for maximally redundant trees (MRT)
  - eg. MRT-Red next-hops = algo#4 , MRT-Blue next-hops = algo#5
  - Single label signifying destination prefix and algorithm (label stack depth = 1)
- Complexity of managing per-algorithm node-SID assignment makes it less attractive.







Proposed ISIS extension to support per-topology/per-algorithm label blocks



Topology-Algorithm-Label-Block sub-TLV

- Same format as SR-Capabilities sub-TLV for specifying label block, plus topology and algorithm field.
- Backward compatible:
  - MT-ID = 0, Algorithm=0 label block is only carried by the SR Capabilities sub-TLV.
  - Topology-Algorithm-Label-Block sub-TLV only carries MT-ID and algorithm value pairs where at least one is non-zero