Implementing SRv6 Flexible Algorithms

Today's evolving networking environment requires advanced routing solutions that offer greater customization, efficiency, and adaptability.

The SRv6 Flexible Algorithm (Flex-Algo) meets the demands by empowering network operators to define routing behaviors tailored to specific operational requirements, extending the capabilities of Segment Routing over IPv6 (SRv6), and integrating seamlessly with Interior Gateway Protocols (IGPs), like IS-IS. Flex-Algo enables precise path computation beyond traditional shortest-path routing.

This chapter explains the key concepts of the SRv6 Flexible Algorithm. It also provides guidance on how to configure it.

SRv6 Flexible Algorithms with IS-IS

Flex-Algo is an SRv6 feature that:

- enables custom path computation within an IGP such as IS-IS by associating specific metric types and constraints, rather than relying solely on the default IGP metric
- allows logical network segmentation by distributing multiple, parallel routing topologies through a standard IGP protocol such as IS-IS, and
- supports adaptive, traffic-engineered paths by associating prefix-SIDs with each algorithm to meet specific operational requirements.

You can identify each Flex-Algo by a numeric value from 128 to 255, and configure it through a Flex-Algo Definition.

Flex-Algo definition

A Flex-Algo definition (FAD) is a mechanism that combines a calculation type, a metric type, and a set of constraints. IS-IS advertises this definition using a Flex-Algo definition sub-TLV (type, length, value).

Flex-Algo definition Sub-TLV Format

The Flex-Algo definition Sub-TLV format includes these fields.

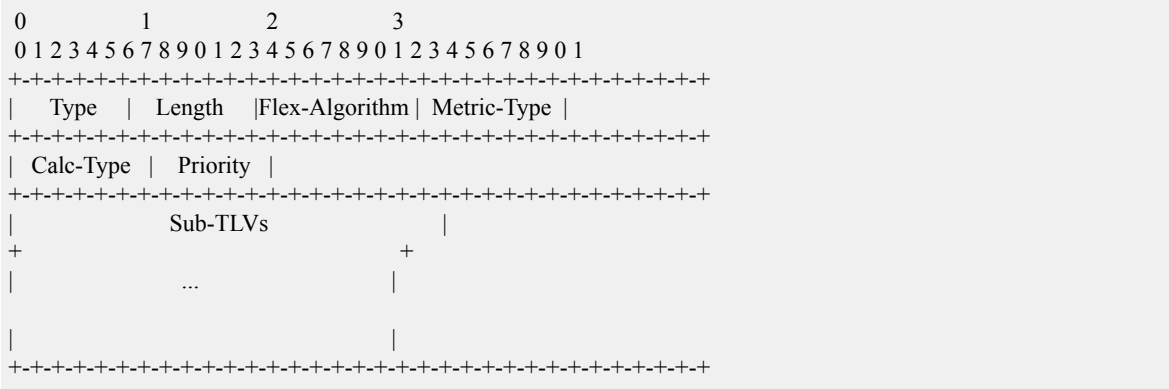

Table 1: Flex-Algo definition field description

Field	Description
Type	Specifies the sub-TLV type.
Length	Specifies the size of the data that follows the Type and Length fields, including any sub-TLVs. Varies based on the sub-TLVs included.
Flex-Algorithm	Represents the Flex-Algo number with a single-octet value ranging from 128 to 255.
Metric-type	Identifies the metric used for path calculation, such as IGP-metric or delay-metric.
Calc-type	Defines the calculation logic with a value from 0 to 127.
Priority	Indicates the advertisement priority with a value between 0 and 255.
Sub-TLVs	Optional; used to specify additional attributes and constraints.

Key concepts of SRv6 Flex-Algo

- Algorithm: An algorithm defines how IGP computes the best path. Routers advertise the support for the algorithm as a node capability. Routers also advertise prefix-SIDs with an algorithm value, so each SID is tightly coupled with its algorithm."An algorithm is a one octet value. You can reserve values from 128 to 255 for Flex-Algo representation.
- Metric types: Metric types are values used by routing protocols to evaluate and select the best network paths. They help determine the preferred route when multiple paths exist to a destination. Each metric type measures specific network characteristics, which may include IGP-metric and delay-metric.
- Constraints: Constraints are rules or conditions that affect which network paths are considered valid or preferred. If you set constraints, you can control route selection based on specific requirements or policies. Constraints can be set to include (require) or exclude (avoid) specific link-affinities in the path

computation. Alternatively, constraints can instruct the routing algorithm to avoid using multiple links from the same SRLG.

Benefits of SRv6 Flex-Algo

Modern networks require advanced capabilities to compute optimal paths, incorporating diverse constraints and metrics beyond traditional shortest-path routing. Flex-Algo addresses these demands by allowing operators to define custom routing behaviors. All routers in a domain share a common understanding of these user-defined algorithm values. This consistency ensures reliable path computation and prevents traffic loops. This approach facilitates several key advantages:

- Custom path selection: Flex-Algo enables operators to define custom algorithms that consider factors beyond the traditional IGP shortest path. This includes advanced constraints such as plane selection in multi-plane networks, extended metrics (like delay), and specific link affinities (avoiding or preferring links with certain attributes or combinations of attributes).
- Granular traffic engineering and network slicing: Flex-Algo allows for precise traffic steering tailored to service requirements. Each network slice can be associated with a unique locator and a user-defined Flex-Algo instance, effectively distributing multiple, parallel routing topologies through a standard IGP protocol such as IS-IS.
- Transport SLA assurance: By incorporating specified metrics, Flex-Algo steers traffic over paths that meet specific transport Service Level Agreements (SLAs), such as minimum delay or minimum cost, rather than relying solely on default shortest path metrics.
- Simplified control plane operations: Flex-Algo reduces operational complexity by enabling direct traffic steering without the need for explicit Segment Routing Traffic Engineering (SR-TE) policies or route coloring.
- Enhanced forwarding efficiency: Flex-Algo improves forwarding efficiency by allowing packets to be steered directly along desired paths, optimizing network resource utilization.

Restrictions to configure SRv6 Flex-Algo with IS-IS

When configuring SRv6 Flex-Algo with IS-IS, ensure that you follow these restrictions:

- You can configure up to eight locators to support SRv6 Flex-Algo.
- The locator algorithm value must be in the range of 128 to 255.
- At least one router in the area, preferably two for redundancy, must advertise the Flex-Algo definition. Without the valid definition being advertised, the Flex-Algo will not function.
- To ensure loop-free forwarding for paths computed for a specific Flex-Algo, all routers in the network must share the same Flex-Algo definition.
- Flex-Algo paths to any prefix must be installed in forwarding using the Prefix-SID advertised for that Flex-Algo. If the Prefix-SID is unknown, the path will not be installed in forwarding for that prefix.

How Flex-Algo prefix-SIDs are advertised

Summary

The key components involved in the process are:

- Routers: Devices that compute paths for Flex-Algos and install IPv6 forwarding entries with SRv6-specific SIDs.
- Flex-Algo SIDs: IPv6 addresses advertised for prefixes to enable Flex-Algo-specific forwarding.
- Shortest path tree computation: A method that determines the optimal path for packet forwarding according to algorithm constraints.

The process of advertising Flex-Algo SIDs in SRv6 involves configuring routers to support and compute paths for Flex-Algo. The IS-IS protocol is used by network nodes to announce their support and advertise SRv6 locators, making all routers aware of available algorithms. Routers then compute paths based on Flex-Algo definitions and install corresponding IPv6 forwarding entries. This ensures that only advertised paths are used for forwarding within the network.

Workflow

These stages describe the process of advertising and forwarding using Flex-Algos in SRv6 with IS-IS:

1. SRv6 locators and slices: Each network slice is assigned a distinct SRv6 locator block. This segmentation enables specific traffic engineering and policy requirements per slice.
2. Advertising Flex-Algo participation: Network nodes announce their support for Flex-Algo instances by advertising their associated SRv6 locators via the IS-IS protocol. This process ensures all routers are aware of which algorithms are available and how they map to specific locators.
3. Shortest Path Tree Computation: Routers compute paths for each Flex-Algo by:
   - Pruning nodes that do not support the Flex-Algo
   - Excluding links with affinities that are not allowed by the algorithm definition
   - Using only links that advertise the metric specified in the Flex-Algo's definition.
4. Forwarding entry installation : Routers install IPv6 forwarding entries for Flex-Algo-computed paths using the advertised SRv6 Locator or End SIDs. If the SRv6 SID for a Flex-Algo is not advertised or known for a given prefix, the corresponding path cannot be installed in the forwarding table.

   This table shows how the system behaves under different conditions during the Flex-Algo advertisement and forwarding process:

When...	And...	Then...	And...
all routers are configured with the same Flex-Algo ID	the Flex-Algo definition is consistently advertised,	all routers can compute and agree on Flex-Algo paths	loop-free and consistent routing is ensured.
a Prefix-SID is advertised for a prefix and Flex-Algo	the prefix meets all Flex-Algo constraints,	the prefix is included in Flex-Algo-specific forwarding	traffic is forwarded along Flex-Algo-computed paths.

When...	And...	Then...	And...
a router does not receive a Flex-Algo definition	or the definition is inconsistent	the router does not compute or install Flex-Algo paths	those paths are excluded from forwarding.
a link is marked with an excluded affinity	the Flex-Algo definition excludes that affinity,	the link is pruned from Flex-Algo path computation	traffic is automatically rerouted over eligible links.
A prefix-SID is not advertised for a prefix,	--	Flex-Algo forwarding entry for that prefix is not installed	prefix follows default IGP path instead.

Configure SRv6 Flex-Algo with IS-IS

This task involves configuring SRv6 locators and integrating them with IS-IS using Flex-Algo to optimize routing decisions based on various metrics.

Follow these steps to configure SRv6 Flex-Algo with IS-IS:

Procedure

Step 1 Configure SRv6 locators on the router.

Each locator is assigned a name, a prefix, and a micro-segment behavior. Optionally, you can associate a locator with a specific flexible algorithm.

In this example, two locators are created. The first locator uses Algo 0 (best-effort), and the second uses Algo 128 (low-latency).

Example:

```
Router(config)#segment-routing srv6
Router(config-srv6)#locators
Router(config-srv6-locators)#locator myLocBestEffort // best-effort locator
Router(config-srv6-locator)#micro-segment behavior unode psp-usd
Router(config-srv6-locator)#prefix 2001:0:1::/48
Router(config-srv6-locator)#exit

Router(config-srv6-locators)#locator myLocLowLat // low-latency (flex algo 128) locator
Router(config-srv6-locator)#micro-segment behavior unode psp-usd
Router(config-srv6-locator)#prefix 2001:0:2::/48
Router(config-srv6-locator)#algorithm 128
Router(config-srv6-locator)#exit
Router(config-srv6)#exit
```

Step 2 Assign SRv6 locators to IS-IS on the router to advertise the defined SRv6 locator prefixes throughout the network

Example:

```
Router(config)#router isis core
Router(config-isis)#address-family ipv6 unicast
Router(config-isis-af)#segment-routing srv6
Router(config-isis-srv6)#locator myLocBestEffort
```

```
Router(config-isis-srv6-loc)#exit
Router(config-isis-srv6)#locator myLocLowLat
Router(config-isis-srv6-loc)#exit
```

Step 3 Define the Flex-Algo with the desired metric on the router to enable path computation.

Example:

```
Router(config)#router isis core
Router(config-isis)#flex-algo 128
Router(config-isis-flex-algo)#metric-type delay
Router(config-isis-flex-algo)#exit
Router(config-isis)#interface HundredGigE0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
```

Step 4 Configure the delay probe on the interface on the router to measure and monitor network latency.

Example:

```
Router(config)# performance-measurement
Router(config-perf-meas)# interface HundredGigE0/0/0/0
Router(config-pm-intf)# delay-measurement
Router(config-pm-intf-dm)# commit
```

Step 5 Verify the SRv6 locators and their associated algorithms on the router to confirm their correct configuration and advertisement within the network.

Example:

```
Router# show segment-routing srv6 locator

Name                  ID       Algo  Prefix                    Status   Flags
--------------------  -------  ----  ------------------------  -------  ---------
myLoc1                3        0     2001:0:8::/48             Up       U
myLocBestEffort       5        0     2001:0:1::/48             Up       U
myLocLowLat           4        128   2001:0:2::/48             Up       U
```

Step 6 Verify the IS-IS Flex-Algo configuration and status for Level-1 and Level-2.

Example:

```
Router# show isis flex-algo 128
IS-IS core Flex-Algo Database

Flex-Algo 128:

Level-2:
        Definition Priority: 128
        Definition Source: Router.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


Level-1:
        Definition Priority: 128
        Definition Source: Router.00, (Local)
        Definition Equal to Local: Yes
        Disabled: No


    Local Priority: 128
```

```
FRR Disabled: No
Microloop Avoidance Disabled: No
```