Enabling Segment Routing Flexible Algorithm

Segment Routing Flexible Algorithm allows operators to customize IGP shortest path computation according to their own needs. An operator can assign custom SR prefix-SIDs to realize forwarding beyond link-cost-based SPF. As a result, Flexible Algorithm provides a traffic engineered path automatically computed by the IGP to any destination reachable by the IGP.

The SR architecture associates prefix-SIDs to an algorithm which defines how the path is computed. Flexible Algorithm allows for user-defined algorithms where the IGP computes paths based on a user-defined combination of metric type and constraint.

This document describes the IS-IS and OSPF extensions to support Segment Routing Flexible Algorithm on an MPLS data-plane.

- Prerequisites for Flexible Algorithm, on page 1
- Building Blocks of Segment Routing Flexible Algorithm, on page 1
- Configuring Flexible Algorithm, on page 3
- Example: Configuring IS-IS Flexible Algorithm, on page 5
- Example: Configuring OSPF Flexible Algorithm, on page 5
- Example: Traffic Steering to Flexible Algorithm Paths, on page 6
- Delay Normalization, on page 10

Prerequisites for Flexible Algorithm

Segment routing must be enabled on the router before the Flexible Algorithm functionality is activated.

Building Blocks of Segment Routing Flexible Algorithm

This section describes the building blocks that are required to support the SR Flexible Algorithm functionality in IS-IS and OSPF.

Flexible Algorithm Definition

Many possible constraints may be used to compute a path over a network. Some networks are deployed with multiple planes. A simple form of constraint may be to use a particular plane. A more sophisticated form of constraint can include some extended metric, like delay, as described in [RFC7810]. Even more advanced case could be to restrict the path and avoid links with certain affinities. Combinations of these are also possible.
To provide a maximum flexibility, the mapping between the algorithm value and its meaning can be defined by the user. When all the routers in the domain have the common understanding what the particular algorithm value represents, the computation for such algorithm is consistent and the traffic is not subject to looping. Here, since the meaning of the algorithm is not defined by any standard, but is defined by the user, it is called a Flexible Algorithm.

**Flexible Algorithm Support Advertisement**

An algorithm defines how the best path is computed by IGP. Routers advertise the support for the algorithm as a node capability. Prefix-SIDs are also advertised with an algorithm value and are tightly coupled with the algorithm itself.

An algorithm is a one octet value. Values from 128 to 255 are reserved for user defined values and are used for Flexible Algorithm representation.

**Flexible Algorithm Definition Advertisement**

To guarantee the loop free forwarding for paths computed for a particular Flexible Algorithm, all routers in the network must share the same definition of the Flexible Algorithm. This is achieved by dedicated router(s) advertising the definition of each Flexible Algorithm. Such advertisement is associated with the priority to make sure that all routers will agree on a single and consistent definition for each Flexible Algorithm.

Definition of Flexible Algorithm includes:

- Metric type
- Affinity constraints

To enable the router to advertise the definition for the particular Flexible Algorithm, `advertise-definition` command is used. At least one router in the area, preferably two for redundancy, must advertise the Flexible Algorithm definition. Without the valid definition being advertised, the Flexible Algorithm will not be functional.

**Flexible Algorithm Prefix-SID Advertisement**

To be able to forward traffic on a Flexible Algorithm specific path, all routers participating in the Flexible Algorithm will install a MPLS labeled path for the Flexible Algorithm specific SID that is advertised for the prefix. Only prefixes for which the Flexible Algorithm specific Prefix-SID is advertised is subject to Flexible Algorithm specific forwarding.

**Calculation of Flexible Algorithm Path**

A router may compute path for multiple Flexible Algorithms. A router must be configured to support particular Flexible Algorithm before it can compute any path for such Flexible Algorithm. A router must have a valid definition of the Flexible Algorithm before such Flexible Algorithm is used.

When computing the shortest path tree for particular Flexible Algorithm:

- All nodes that do not advertise support for such Flexible Algorithm will be pruned from the topology.
- If the Flexible Algorithm definition includes affinities that are excluded, then all links for which any of such affinities are advertised will be pruned from the topology.
- Router uses the metric that is part of the Flexible Algorithm definition. If the metric is not advertised for the particular link, such link will be pruned from the topology.

For IS-IS, Loop Free Alternate (LFA) paths, TI-LFA backup paths, and Microloop Avoidance paths for particular Flexible Algorithm are computed using the same constraints as the calculation of the primary paths for such Flexible Algorithm. These paths use Prefix-SIDs advertised specifically for such Flexible Algorithm in order to enforce a backup or microloop avoidance path.

---

**Note**

LFA, TI-LFA, and Microloop Avoidance for Flexible Algorithm routes are not supported in OSPF.

---

### Installation of Forwarding Entries for Flexible Algorithm Paths

Flexible Algorithm path to any prefix must be installed in the forwarding using the Prefix-SID that was advertised for such Flexible Algorithm. If the Prefix-SID for Flexible Algorithm is not known, such Flexible Algorithm path is not installed in forwarding for such prefix.

Only MPLS to MPLS entries are installed for a Flexible Algorithm path. No IP to IP or IP to MPLS entries are installed. These follow the native IPG paths computed based on the default algorithm and regular IGP metrics.

### Flexible Algorithm Prefix-SID Redistribution

Previously, prefix redistribution from IS-IS to another IS-IS instance or protocol was limited to SR algorithm 0 (regular SPF) prefix SIDs; SR algorithm 1 (Strict SPF) and SR algorithms 128-255 (Flexible Algorithm) prefix SIDs were not redistributed along with the prefix. The Segment Routing IS-IS Flexible Algorithm Prefix SID Redistribution feature allows redistribution of strict and flexible algorithms prefix SIDs from IS-IS to another IS-IS instance or protocols. This feature is enabled automatically when you configure redistribution of IS-IS Routes with strict or Flexible Algorithm SIDs.

### Configuring Flexible Algorithm

**Note**

For information about the commands usage, see the *Segment Routing Command Reference for Cisco NCS 5500 Series, Cisco NCS 540 Series, and Cisco NCS 560 Series Routers*.

The following ISIS and OSPF configuration sub-mode is used to configure Flexible Algorithm:

```
flex-algo algorithm number
```

*algorithm number* — value from 128 to 255

**Commands under Flexible Algorithm Configuration Mode**

The following commands are used to configure Flexible Algorithm definition under the flex-algo sub-mode:

- IS-IS
**metric-type delay**

*Note* By default the regular IGP metric is used. If delay metric is enabled, the advertised delay on the link is used as a metric for Flexible Algorithm computation.

**OSPF**

*metric-type* {delay | te-metric}

*Note* By default the regular IGP metric is used. If delay or TE metric is enabled, the advertised delay or TE metric on the link is used as a metric for Flexible Algorithm computation.

*affinity* {include-any | include-all | exclude-any} name, name2, ...

*name*—name of the affinity map

*priority* priority value

*priority value*—priority used during the Flexible Algorithm definition election.

The following command is used to include the Flexible Algorithm prefix metric in the advertised Flexible Algorithm definition in IS-IS:

prefix-metric

The following command is used to enable advertisement of the Flexible Algorithm definition in IS-IS:

advertise-definition

**Commands for Affinity Configuration**

The following command is used for defining the affinity-map. Affinity-map associates the name with the particular bit positions in the Extended Admin Group bitmask.

affinity-map name bit-position bit number

• *name*—name of the affinity-map.

• *bit number*—bit position in the Extended Admin Group bitmask.

The following command is used to associate the affinity with an interface:

affinity flex-algo name 1, name 2, ...

*name*—name of the affinity-map

**Command for Prefix-SID Configuration**

The following command is used to advertise prefix-SID for default and strict-SPF algorithm:
prefix-sid [strict-spf | algorithm algorithm-number] [index | absolute] sid value

- algorithm-number—Flexible Algorithm number
- sid value—SID value

Example: Configuring IS-IS Flexible Algorithm

```sh
router isis 1
  affinity-map red bit-position 65
  affinity-map blue bit-position 8
  affinity-map green bit-position 201

  flex-algo 128
  advertise-definition
  affinity exclude-any red
  affinity include-any blue
! flex-algo 129
  affinity exclude-any green
!

  address-family ipv4 unicast
    segment-routing mpls
!
  interface Loopback0
    address-family ipv4 unicast
      prefix-sid algorithm 128 index 100
      prefix-sid algorithm 129 index 101
!

  interface GigabitEthernet0/0/0/0
    affinity flex-algo red
!
  interface GigabitEthernet0/0/0/1
    affinity flex-algo blue red
!
  interface GigabitEthernet0/0/0/2
    affinity flex-algo blue
!
```

Example: Configuring OSPF Flexible Algorithm

```sh
router ospf 1
  flex-algo 130
  priority 200
  affinity exclude-any red
  blue
! metric-type delay
! flex-algo 140
  affinity include-all green
!
  affinity include-any red
```
Example: Traffic Steering to Flexible Algorithm Paths

BGP Routes on PE – Color Based Steering

SR-TE On Demand Next-Hop (ODN) feature can be used to steer the BGP traffic towards the Flexible Algorithm paths.

The following example configuration shows how to setup BGP steering local policy, assuming two router: R1 (2.2.2.2) and R2 (4.4.4.4), in the topology.

Configuration on router R1:

vrf Test
address-family ipv4 unicast
import route-target
1:150
!
export route-policy SET_COLOR_RED_HI_BW
export route-target
1:150
!
!
interface Loopback0
ipv4 address 2.2.2.2 255.255.255.255
!
interface Loopback150
vrf Test
ipv4 address 2.2.2.222 255.255.255.255
!
interface TenGigE0/1/0/3/0
description exr1 to cxr1
ipv4 address 10.0.20.2 255.255.255.0
!
extcommunity-set opaque color129-red-igp
129
derive-policy PASS
pass
derive-policy
!
derive-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
derive-policy
!
router isis 1
  is-type level-2-only
  net 49.0001.0000.0000.0002.00
  log adjacency changes
  affinity-map RED bit-position 28
  flex-algo 128
  priority 228

address-family ipv4 unicast
  metric-style wide
  advertise link attributes
  router-id 2.2.2.2
  segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
    prefix-sid index 2
    prefix-sid algorithm 128 index 282
!
interface TenGigE0/1/0/3/0
  point-to-point
  address-family ipv4 unicast
!
!
router bgp 65000
bgp router-id 2.2.2.2
address-family ipv4 unicast

address-family vpnv4 unicast
  retain route-target all

neighbor-group RR-services-group
remote-as 65000
update-source Loopback0
address-family ipv4 unicast

address-family vpnv4 unicast
!
neighbor 4.4.4.4
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
    redistribute connected
!
segment-routing
traffic-eng
logging
policy status
!
segment-list sl-cxr1
  index 10 mpls label 16294
!
policy pol-foo
  color 129 end-point ipv4 4.4.4.4
  candidate-paths
  preference 100
  explicit segment-list sl-cxr1
!
!

Configuration on router R2:

vrf Test
address-family ipv4 unicast
  import route-target
  1:150
!
  export route-policy SET_COLOR_RED_HI_BW
  export route-target
  1:150
!
!
interface TenGigE0/1/0/1
description cxr1 to exr1
ipv4 address 10.0.20.1 255.255.255.0
!
extcommunity-set opaque color129-red-igp
  129
end-set
!
route-policy PASS
  pass
end-policy
!
route-policy SET_COLOR_RED_HI_BW
  set extcommunity color color129-red-igp
  pass
end-policy
!
router isis 1
is-type level-2-only
net 49.0001.0000.0000.0004.00
log adjacency changes
affinity-map RED bit-position 28
affinity-map BLUE bit-position 29
affinity-map GREEN bit-position 30
flex-algo 128
  priority 228
! flex-algo 129
  priority 229
! flex-algo 130
  priority 230
!
address-family ipv4 unicast
metric-style wide
advertise link attributes
router-id 4.4.4.4
segment-routing mpls
!
interface Loopback0
  address-family ipv4 unicast
    prefix-sid index 4
    prefix-sid algorithm 128 index 284
    prefix-sid algorithm 129 index 294
    prefix-sid algorithm 130 index 304
!
interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
!
interface TenGigE0/1/0/1
  point-to-point
  address-family ipv4 unicast
!
router bgp 65000
  bgp router-id 4.4.4.4
  address-family ipv4 unicast
  address-family vpnv4 unicast
!
neighbor-group RR-services-group
  remote-as 65000
  update-source Loopback0
  address-family ipv4 unicast
  address-family vpnv4 unicast
!
neighbor 1.1.1.1
  use neighbor-group RR-services-group
!
neighbor 2.2.2.2
  use neighbor-group RR-services-group
!
vrf Test
  rd auto
  address-family ipv4 unicast
    redistribute connected
!
neighbor 25.1.1.2
  remote-as 4
  address-family ipv4 unicast
  route-policy PASS in
  route-policy PASS out
!
!
segment-routing
!
end
Delay Normalization

Performance measurement (PM) measures various link characteristics like packet loss and delay. Such characteristics can be used by IS-IS as a metric for Flexible Algorithm computation. Low latency routing using dynamic delay measurement is one of the primary use cases for Flexible Algorithm technology.

Delay is measured in microseconds. If delay values are taken as measured and used as link metrics during the IS-IS topology computation, some valid ECMP paths might be unused because of the negligible difference in the link delay.

The Delay Normalization feature computes a normalized delay value and uses the normalized value instead. This value is advertised and used as a metric during the Flexible Algorithm computation.

The normalization is performed when the delay is received from the delay measurement component. When the next value is received, it is normalized and compared to the previous saved normalized value. If the values are different, then the LSP generation is triggered.

The following formula is used to calculate the normalized value:

- \( D_m \) – measured Delay
- \( Int \) – configured normalized Interval
- \( Off \) – configured normalized Offset (must be less than the normalized interval \( Int \))
- \( D_n \) – normalized Delay
- \( a = \frac{D_m}{Int} \) (rounded down)
- \( b = a \times Int + Off \)

If the measured delay (\( D_m \)) is less than or equal to \( b \), then the normalized delay (\( D_n \)) is equal to \( b \). Otherwise, \( D_n \) is \( b + Int \).

Example

The following example shows a low-latency service. The intent is to avoid high-latency links (1-6, 5-2). Links 1-2 and 5-6 are both low-latency links. The measured latency is not equal, but the difference is insignificant.

We can normalize the measured latency before it is advertised and used by IS-IS. Consider a scenario with the following:

- Interval = 10
- Offset = 3
The measured delays will be normalized as follows:

- \( D_m = 29 \)
  - \( a = \frac{29}{10} = 2.9 \) (rounded down to 2)
  - \( b = 2 \times 10 + 3 = 23 \)

  In this case, \( D_m \) (29) is greater than \( b \) (23); so \( D_n \) is equal to \( b + 10 = 33 \)

- \( D_m = 31 \)
  - \( a = \frac{31}{10} = 3.1 \) (rounded down to 3)
  - \( b = 3 \times 10 + 3 = 33 \)

  In this case, \( D_m \) (31) is less than \( b \) (33); so \( D_n \) is \( b = 33 \)

The link delay between 1-2 and 5-6 is normalized to 33.
Configuration

Delay normalization is disabled by default. To enable and configure delay normalization, use the `delay normalize interval interval [offset offset]` command.

- `interval` – The value of the normalize interval in microseconds.
- `offset` – The value of the normalized offset in microseconds. This value must be smaller than the value of normalized interval.

```plaintext
router isis 1
interface GigEth 0/0/0/0
    delay normalize interval 10 offset 3
    address-family ipv4 unicast
    metric 77
```