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

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 Membership

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

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
OSPF: Microloop Avoidance for Flexible Algorithm	Release 7.3.2	This feature extends the current OSPF Flexible Algorithm functionality to support Microloop Avoidance.

Feature Name	Release Information	Feature Description
OSPF: TI-LFA for Flexible Algorithm	Release 7.3.1	This feature extends the current OSPF Flexible Algorithm functionality to support TI-LFA.

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 Flexible Algorithm is used.

The router uses the following rules to prune links from the topology during the Flexible Algorithm computation:

- All nodes that don't advertise support for Flexible Algorithm are pruned from the topology.
- Affinities:
  - Check if any exclude affinity rule is part of the Flexible Algorithm Definition. If such exclude rule exists, check if any color that is part of the exclude rule is also set on the link. If such a color is set, the link must be pruned from the computation.
  - Check if any include-any affinity rule is part of the Flexible Algorithm Definition. If such include-any rule exists, check if any color that is part of the include-any rule is also set on the link. If no such color is set, the link must be pruned from the computation.
  - Check if any include-all affinity rule is part of the Flexible Algorithm Definition. If such include-all rule exists, check if all colors that are part of the include-all rule are also set on the link. If all such colors are not set on the link, the link must be pruned from the computation

Note See Flexible Algorithm Affinity Constraint.

- Router uses the metric that is part of the Flexible Algorithm definition. If the metric isn't advertised for the particular link, the link is pruned from the topology.

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.

Configuring Microloop Avoidance for Flexible Algorithm

By default, Microloop Avoidance per Flexible Algorithm instance follows Microloop Avoidance configuration for algo-0. For information about configuring Microloop Avoidance, see Configure Segment Routing Microloop Avoidance.

You can disable Microloop Avoidance for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo microloop avoidance disable

router ospf process flex-algo algo microloop avoidance disable
```

Configuring LFA / TI-LFA for Flexible Algorithm

By default, LFA/TI-LFA per Flexible Algorithm instance follows LFA/TI-LFA configuration for algo-0. For information about configuring TI-LFA, see Configure Topology-Independent Loop-Free Alternate (TI-LFA).

You can disable TI-LFA for Flexible Algorithm using the following commands:

```
router isis instance flex-algo algo fast-reroute disable

router ospf process flex-algo algo fast-reroute disable
```

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

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.

Flexible Algorithm Prefix Metric

A limitation of the existing Flexible Algorithm functionality in IS-IS is the inability to compute the best path to a prefix in a remote area or remote IGP domain. Prefixes are advertised between IS-IS areas or between protocol domains, but the existing prefix metric does not reflect any of the constraints used for Flexible Algorithm path. Although the best Flexible Algorithm path can be computed to the inter-area or redistributed prefix inside the area, the path may not represent the overall best path through multiple areas or IGP domains.

The Flexible Algorithm Prefix Metric feature introduces a Flexible Algorithm-specific prefix-metric in the IS-IS prefix advertisement. The prefix-metric provides a way to compute the best end-to-end Flexible Algorithm optimized paths across multiple areas or domains.

Note The Flexible Algorithm definition must be consistent between domains or areas. Refer to section 8 in IETF draft https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/.

Configuring Flexible Algorithm

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

```
router isis instance flex-algo algo
```

```
router ospf process flex-algo algo
```

algo—value from 128 to 255

Configuring Flexible Algorithm Definitions

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**} *name1, 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 to include the Flexible Algorithm prefix metric in the advertised Flexible Algorithm definition in IS-IS :

```
router isis instance flex-algo algo prefix-metric
```

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

```
router isis instance flex-algo algo advertise-definition
```

Configuring Affinity

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.

```
router isis instance flex-algo algo affinity-map name bit-position bit number
```

```
router ospf process flex-algo algo affinity-map name bit-position bit number
```

name—name of the affinity-map

Configuring Prefix-SID Advertisement

The following command is used to advertise prefix-SID for default and strict-SPF algorithm:

```
router isis instance interface type interface-path-id address-family {ipv4 | ipv6} [unicast]
 prefix-sid [strict-spf | algorithm algorithm-number]  [index | absolute] sid value

router ospf process area area interface Loopback interface-instance 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

```
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

```
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
  !
 !

  interface Loopback0
   prefix-sid index 10
   prefix-sid strict-spf index 40
   prefix-sid algorithm 128 absolute 16128
   prefix-sid algorithm 129 index 129
   prefix-sid algorithm 200 index 20
   prefix-sid algorithm 210 index 30
  !
 !

  interface GigabitEthernet0/0/0/0
   flex-algo affinity
    color red
    color blue
   !
  !

 affinity-map
  color red bit-position 10
  color blue bit-position 11
 !
```

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
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 49.0001.0000.0000.0002.00
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 10.1.1.1
  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
```