セグメント ルーティング フレキシブル アルゴリズムの有効化

セグメントルーティングフレキシブルアルゴリズムを使用すると、オペレータは、独自のニーズに応じてIGP最短パス計算をカスタマイズできます。オペレータは、リンクコストベースのSPF よりも優れた転送を実現するために、カスタムの SR プレフィックス SID を割り当てることができます。結果として、フレキシブルアルゴリズムにより、IGPから到達可能なあらゆる宛先へのトラフィック エンジニアリングに基づくパスを IGP で自動的に計算できます。

SR アーキテクチャでは、パスの計算方法を定義するアルゴリズムにプレフィックス SID が関連付けられます。フレキシブルアルゴリズムにより、ユーザが定義したメトリックタイプと制約の組み合わせに基づいて IGP でパスを計算する、ユーザ定義のアルゴリズムを実現できます。

このマニュアルでは、MPLS データプレーンでセグメント ルーティング フレキシブル アルゴリズムをサポートするための IS-IS および OSPF 拡張機能について説明します。

フレキシブルアルゴリズムの前提条件

フレキシブルアルゴリズム機能をアクティブ化する前に、ルータでセグメントルーティングを有効にする必要があります。

セグメント ルーティング フレキシブル アルゴリズムの構成要素

このセクションでは、IS-IS および OSPF で SR フレキシブルアルゴリズム機能をサポートするために必要な構成要素について説明します。

フレキシブルアルゴリズムの定義

ネットワーク上のパスを計算するために、考えられる多くの制約が使用される可能性があります。一部のネットワークは複数のプレーンを使用して展開されます。単純な形の制約は、特定のプレーンを使用することである場合もあります。より洗練された形の制約には、「RFC7810」で説明されているように、遅延など、一部の拡張メトリックが含まれます。さらに高度なケースでは、パスを制限し、特定のアフィニティを持つリンクを回避することも考えられます。また、これらを組み合わせて使用することも可能です。最大限の柔軟性を得られるように、ユーザは、アルゴリズム値とその意味の間のマッピングを定義できます。ドメイン内のすべてのルータで、特定のアルゴリズム値が持つ意味について共通の認識が確立されている場合、アルゴリズムの計算は一貫性のあるものとなり、トラフィックがループすることもありません。つまり、アルゴリズムの意味が標準によってではなく、ユーザによって定義されるため、フレキシブルアルゴリズムと呼ばれます。

フレキシブルアルゴリズムのサポートのアドバタイズメント

アルゴリズムは、IGP によるベストパスの計算方法を定義します。ルータは、ノード機能としてアルゴリズムのサポートをアドバタイズします。プレフィックス SID もアルゴリズム値とともにアドバタイズされ、アルゴリズム自体と密接に結び付けられます。

アルゴリズムは 1 つのオクテット値です。128 ～ 255 までの値が、ユーザ定義の値用に予約されており、フレキシブルアルゴリズムの表現に使用されます。

フレキシブルアルゴリズムの定義のアドバタイズメント

特定のフレキシブルアルゴリズムで計算されたパスについてループフリーの転送を実現するためには、ネットワーク内のすべてのルータでフレキシブルアルゴリズムの同じ定義を共有する必要があります。これは、各フレキシブルアルゴリズムの定義をアドバタイズする専用ルータによって実現されます。このようなアドバタイズメントでは、優先度を設定して、フレキシブルアルゴリズムごとに一貫した 1 つの定義がすべてのルータで適用されるようにします。

フレキシブルアルゴリズムの定義には以下が含まれます。

- メトリックタイプ
- アフィニティ制約

特定のフレキシブルアルゴリズムの定義をルータからアドバタイズできるようにするには、**advertise-definition** コマンドを使用します。エリア内の少なくとも 1 つのルータ、または可能であれば冗長性を確保するために2つのルータで、フレキシブルアルゴリズム定義をアドバタイズする必要があります。有効な定義がアドバタイズされない場合、フレキシブルアルゴリズムは機能しません。

フレキシブルアルゴリズムのプレフィックス SID のアドバタイズメント

フレキシブルアルゴリズム固有のパスでトラフィックを転送できるように、フレキシブルアルゴリズムに参加するすべてのルータは、プレフィックスに対してアドバタイズされるフレキシブルアルゴリズム固有のSIDのMPLSラベル付きパスを組み込みます。フレキシブルアルゴリズム固有のプレフィックスSIDがアドバタイズされるプレフィックスだけが、フレキシブルアルゴリズム固有の転送の対象となります。

フレキシブル アルゴリズム パスの計算

ルータは、複数のフレキシブルアルゴリズムのパスを計算できます。このようなフレキシブルアルゴリズムのパスを計算する前に、特定のフレキシブルアルゴリズムをサポートするようにルータを設定する必要があります。このようなフレキシブルアルゴリズムを使用する場合は、あらかじめ、フレキシブルアルゴリズムの有効な定義をルータで確立しておく必要があります。

特定のフレキシブルアルゴリズムの最短パスツリーを計算する場合は、次のようなプロセスになります。

- このようなフレキシブルアルゴリズムのサポートをアドバタイズしないすべてのノードは、トポロジからプルーニングされます。
- 除外されるアフィニティがフレキシブルアルゴリズム定義に含まれている場合、そのようなアフィニティのいずれかがアドバタイズされるすべてのリンクは、トポロジからプルーニングされます。
- ルータは、フレキシブルアルゴリズム定義の一部であるメトリックを使用します。特定のリンクに対してメトリックがアドバタイズされていない場合、そのリンクはトポロジからプルーニングされます。

IS-IS では、特定のフレキシブルアルゴリズムのループフリー代替（LFA）パス、TI-LFA バックアップパス、およびマイクロループ回避パスは、このようなフレキシブルアルゴリズムのプライマリパスの計算と同じ制約を使用して計算されます。これらのパスでは、バックアップパスまたはマイクロループ回避パスを適用するために、フレキシブルアルゴリズム用にアドバタイズされたプレフィックス SID が使用されます。

（注） フレキシブル アルゴリズム ルートの LFA、TI-LFA、およびマイクロループ回避は、OSPF ではサポートされていません。

フレキシブル アルゴリズム パスの転送エントリの組み込み

フレキシブルアルゴリズム用にアドバタイズされたプレフィックス SID を使用して、あらゆるプレフィックスに対するフレキシブル アルゴリズム パスを転送に組み込む必要があります。フレキシブルアルゴリズムのプレフィックス SID が不明な場合、そのようなプレフィックスの転送にフレキシブル アルゴリズム パスは組み込まれません。

フレキシブル アルゴリズム パスの MPLS から MPLS へのエントリのみが組み込まれます。IP から IP へのエントリまたは IP から MPLS へのエントリは組み込まれません。これらは、デフォルトのアルゴリズムと通常の IGP メトリックに基づいて計算されたネイティブ IPG パスに従います。

フレキシブルアルゴリズムのプレフィックス SID の再配布

これまで、IS-IS インスタンスまたは IS-IS プロトコル間のプレフィックスの再配布は、SR アルゴリズム 0（通常の SPF）のプレフィックス SID に制限されていました。SR アルゴリズム 1（厳格な SPF）および SR アルゴリズム 128-255（フレキシブルアルゴリズム）のプレフィックス SID がプレフィックスとともに再配布されることはありませんでした。セグメントルーティング IS-IS フレキシブルアルゴリズムのプレフィックス SID の再配布機能により、IS-IS インスタンスまたは IS-IS プロトコル間で厳格な SPF およびフレキシブルアルゴリズムのプレフィックス SID を再配布できます。この機能は、厳格な SPF またはフレキシブルアルゴリズムの SID を使用する IS-IS ルートの再配布を設定すると、自動的に有効になります。

フレキシブルアルゴリズムの設定

（注） コマンドの使用方法については、『』を参照してください。

フレキシブルアルゴリズムを設定するには、次の ISIS および OSPF コンフィギュレーション サブモードを使用します。

flex-algo *algorithm number*

algorithm number：128 ～ 255 の値

フレキシブル アルゴリズム コンフィギュレーション モードでのコマンド

フレキシブル アルゴリズム サブモードでフレキシブルアルゴリズム定義を設定するには、次のコマンドを使用します。

- IS-IS

  metric-type delay

  （注） デフォルトでは、通常の IGP メトリックが使用されます。遅延メトリックが有効になっている場合、リンク上でアドバタイズされた遅延が、フレキシブルアルゴリズム計算のメトリックとして使用されます。

  OSPF

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

  （注） デフォルトでは、通常の IGP メトリックが使用されます。遅延または TE メトリックが有効になっている場合、リンク上でアドバタイズされた遅延または TE メトリックが、フレキシブルアルゴリズム計算のメトリックとして使用されます。

- **affinity** { **include-any** | **include-all** | **exclude-any**} *name1, name2, …*

  name：アフィニティマップの名前

- **priority** *priority value*

  priority value：フレキシブルアルゴリズム定義の選択時に使用される優先度

IS-IS でのフレキシブルアルゴリズム定義のアドバタイズメントを有効にするには、次のコマンドを使用します。

advertise-definition

アフィニティ設定用のコマンド

アフィニティマップを定義する際は、次のコマンドを使用します。アフィニティマップは、拡張管理者グループのビットマスク内の特定のビット位置に名前を関連付けます。

affinity-map *name* **bit-position** *bit number*

- *name*：アフィニティマップの名前
- *bit number*：拡張管理者グループのビットマスク内のビット位置

アフィニティをインターフェイスに関連付けるには、次のコマンドを使用します。

affinity flex-algo *name 1, name 2, …*

name：アフィニティマップの名前

プレフィックス SID 設定用のコマンド

デフォルトおよび厳格なSPFのアルゴリズムのプレフィックスSIDをアドバタイズするには、次のコマンドを使用します。

prefix-sid [**strict-spf** | **algorithm** *algorithm-number*] [**index** | **absolute**] *sid value*

- *algorithm-number*：フレキシブルアルゴリズム番号
- *sid value*：SID 値

例：IS-IS フレキシブルアルゴリズムの設定

```
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
!
```

例：OSPF フレキシブルアルゴリズムの設定

```
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
!
```

例：フレキシブルアルゴリズムパスへのトラフィックのステアリング

PE 上の BGP ルート：カラーベースのステアリング

SR-TE オンデマンドネクストホップ（ODN）機能を使用すると、BGP トラフィックをフレキシブル アルゴリズム パスに誘導できます。

次の設定例は、トポロジ内の 2 つのルータ、R1（2.2.2.2）と R2（4.4.4.4）を前提として、BGP ステアリング ローカル ポリシーを設定する方法を示しています。

ルータ R1 での設定

```
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
     !
    !
   !
  !
!
!
```

ルータ R2 での設定

```
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
```