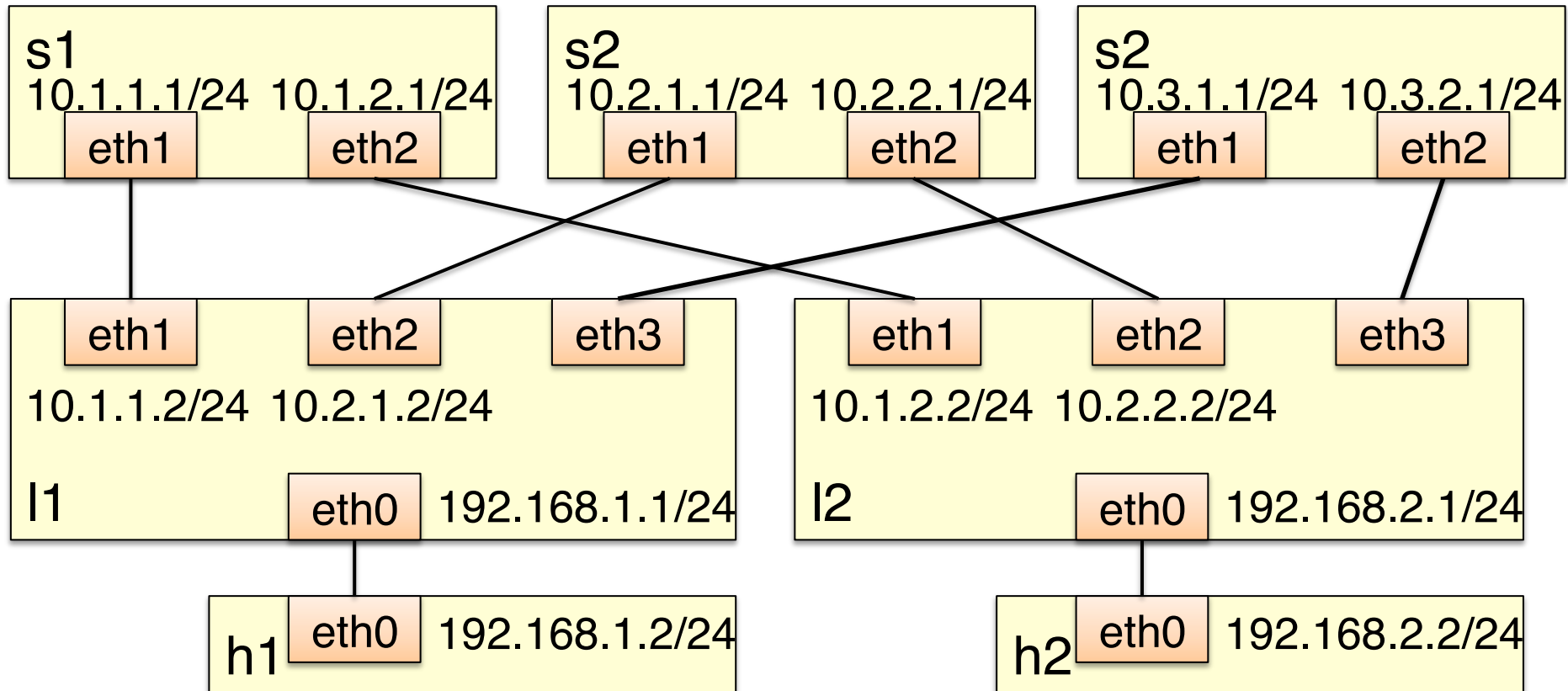


ネットワークトポロジ（再掲）



Quagga

今回のスクリプトを実行すると、Docker内でQuaggaというルーティングソフトウェアが自動で起動しています

Quagga : FreeBSD, NetBSD, Linux, Solaris などの UNIX で動作し、OSPFv2, OSPFv3, RIP v1, RIP v2, RIPng, BGP-4 などのルーティングプロトコルが実装されている。

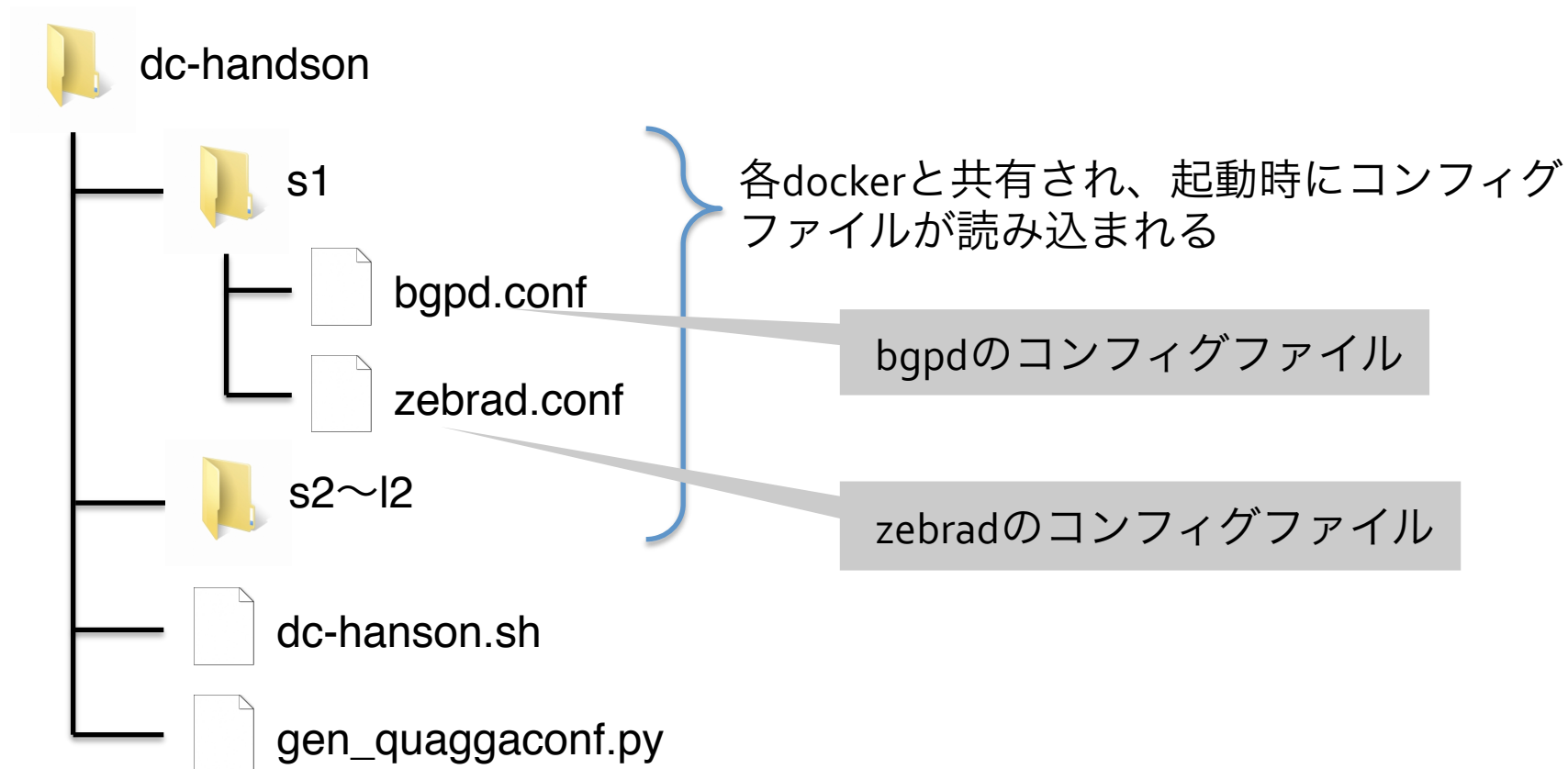
[<http://www.nongnu.org/quagga/>]

Quaggaの内部にzebraデーモンとbgpデーモンがある

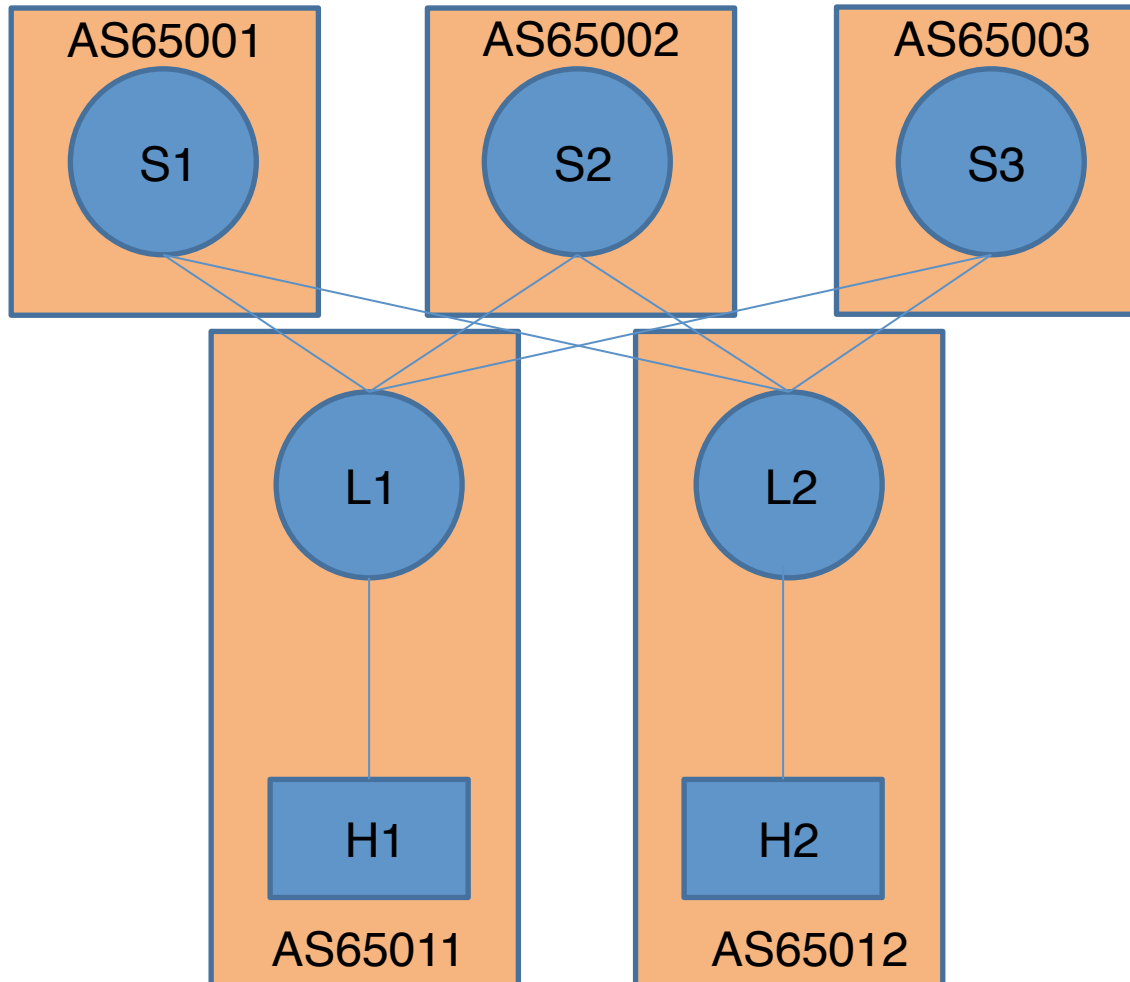
bgpデーモン: 実際にbgpプロトコルを扱うデーモン

zebraデーモン: Quaggaのコアデーモン。bgpデーモンによって取得された経路に基づいてLinuxのカーネルにルーティング情報を書き込む

bgpdのコンフィグファイル



BGPのAS構成



bgpd.confの確認

s1内のbgpd.conf

```
1  hostname spine1
2  password zebra
3  log file /var/log/quagga/bgpd.log
4  !
5  !
6  router bgp 65001
7  bgp router-id 10.1.1.1
8  network 10.1.1.0/24
9  network 10.1.2.0/24
10 !
11 !
12 neighbor 10.1.1.2 remote-as 65011
13 neighbor 10.1.1.2 timers 1 4
14 neighbor 10.1.1.2 version 4
15 neighbor 10.1.1.2 timers connect 1
16 neighbor 10.1.1.2 route-map LEAF_ROUTE0 in
17 !
18 !
19 neighbor 10.1.2.2 remote-as 65012
20 neighbor 10.1.2.2 timers 1 4
21 neighbor 10.1.2.2 version 4
22 neighbor 10.1.2.2 timers connect 1
23 neighbor 10.1.2.2 route-map LEAF_ROUTE1 in
24 !
25 !
```

自分のAS番号
所属するネットワーク

隣接するBGPルータのAS番号
ルートの名前などなど

BGP経路情報を確認

1. ルーターにログイン

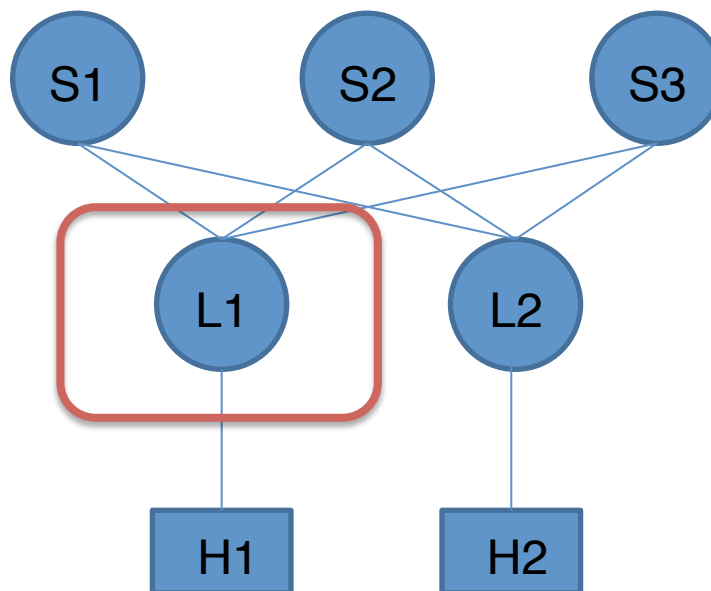
`% docker exec -it <container-name> telnet localhost bgpd`

例) `%docker exec -it l1 telnet localhost bgpd`

Password = zebra

2. 経路情報を表示

`leaf1> show ip bgp`



BGP経路情報を確認

I1の経路

Network	Next Hop	Metric	LocPrf	Weight	Path
* 10.1.1.0/24	10.1.1.1	0		0	65001 i
*>	0.0.0.0	0		32768	i
* 10.1.2.0/24	10.3.1.1			0	65003 65012 i
*>	10.1.1.1	0		0	65001 i
* 10.2.1.0/24	10.2.1.1			0	65002 65012 i
*>	0.0.0.0	0		32768	i
* 10.2.2.0/24	10.3.1.1			0	65003 65012 i
*>	10.1.1.1			0	65001 65012 i
* 10.3.1.0/24	10.2.1.1	0		0	65002 i
*>	10.3.1.1	0		0	65003 i
*>	0.0.0.0	0		32768	i
*> 10.3.2.0/24	10.3.1.1	0		0	65003 i
*>	10.1.1.1			0	65001 65012 i
*>	10.2.1.1			0	65002 65012 i
*> 192.168.1.0	0.0.0.0	0		32768	i
* 192.168.2.0	10.3.1.1		100	0	65003 65012 i
*>	10.1.1.1		300	0	65001 65012 i
*>	10.2.1.1		200	0	65002 65012 i

BGP経路情報を確認

I1の経路

ベストパスの証

h2(192.168.2.2)へベストパスとして
s1(10.1.1.1)経由のパスが選択

Network	Next Hop	Metric	LocPrf	Weight	Path
* 10.1.1.0/24	10.1.1.1	0		0	65001 i
*> 10.1.1.0/24	0.0.0.0	0		32768	i
* 10.1.2.0/24	10.3.1.1			0	65003 65012 i
*> 10.1.2.0/24	10.1.1.1	0		0	65001 i
* 10.1.2.0/24	10.2.1.1			0	65002 65012 i
* 10.2.1.0/24					
*> 10.2.1.0/24					
* 10.2.2.0/24					
*> 10.2.2.0/24					
* 10.3.1.0/24					
*> 10.3.1.0/24					
*> 10.3.2.0/24	10.3.1.1	0		0	65003 i
* 10.3.2.0/24	10.1.1.1			0	65001 65012 i
* 10.3.2.0/24	10.2.1.1			0	65002 65012 i
*> 192.168.1.0	0.0.0.0	0		32768	i
* 192.168.2.0	10.3.1.1		100	0	65003 65012 i
*> 192.168.2.0	10.1.1.1	300		0	65001 65012 i
* 192.168.2.0	10.2.1.1	200		0	65002 65012 i

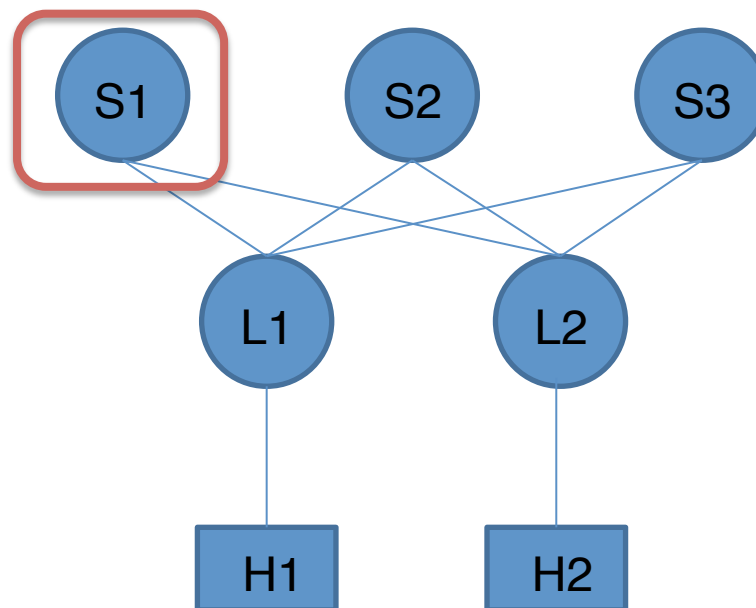
BGP経路情報を確認

1. ルーターにログイン

```
% docker exec -it s1 telnet localhost bgpd  
Password = zebra
```

2. 経路情報を表示

```
spine1> show ip bgp
```



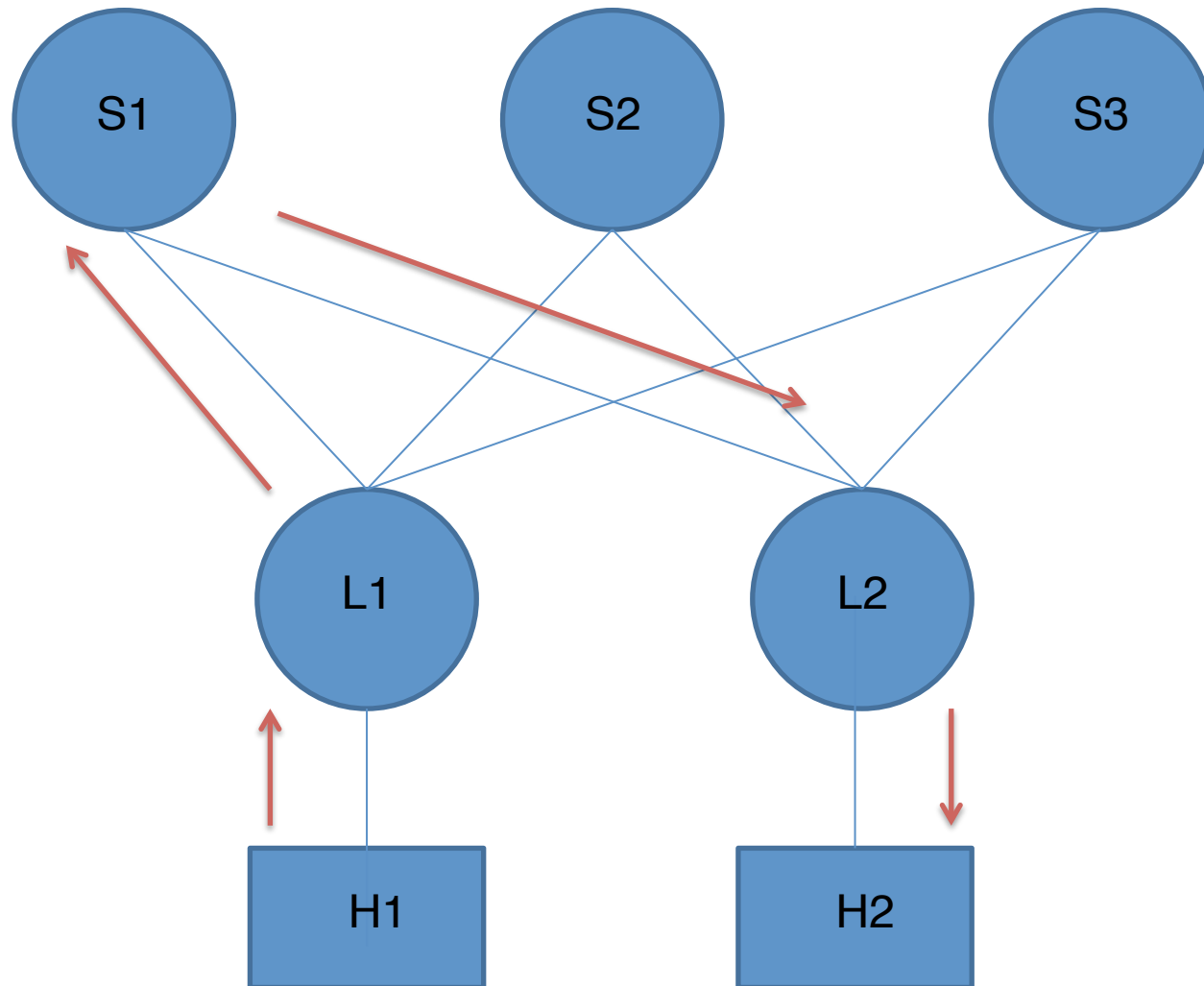
BGP経路情報を確認

s1の経路

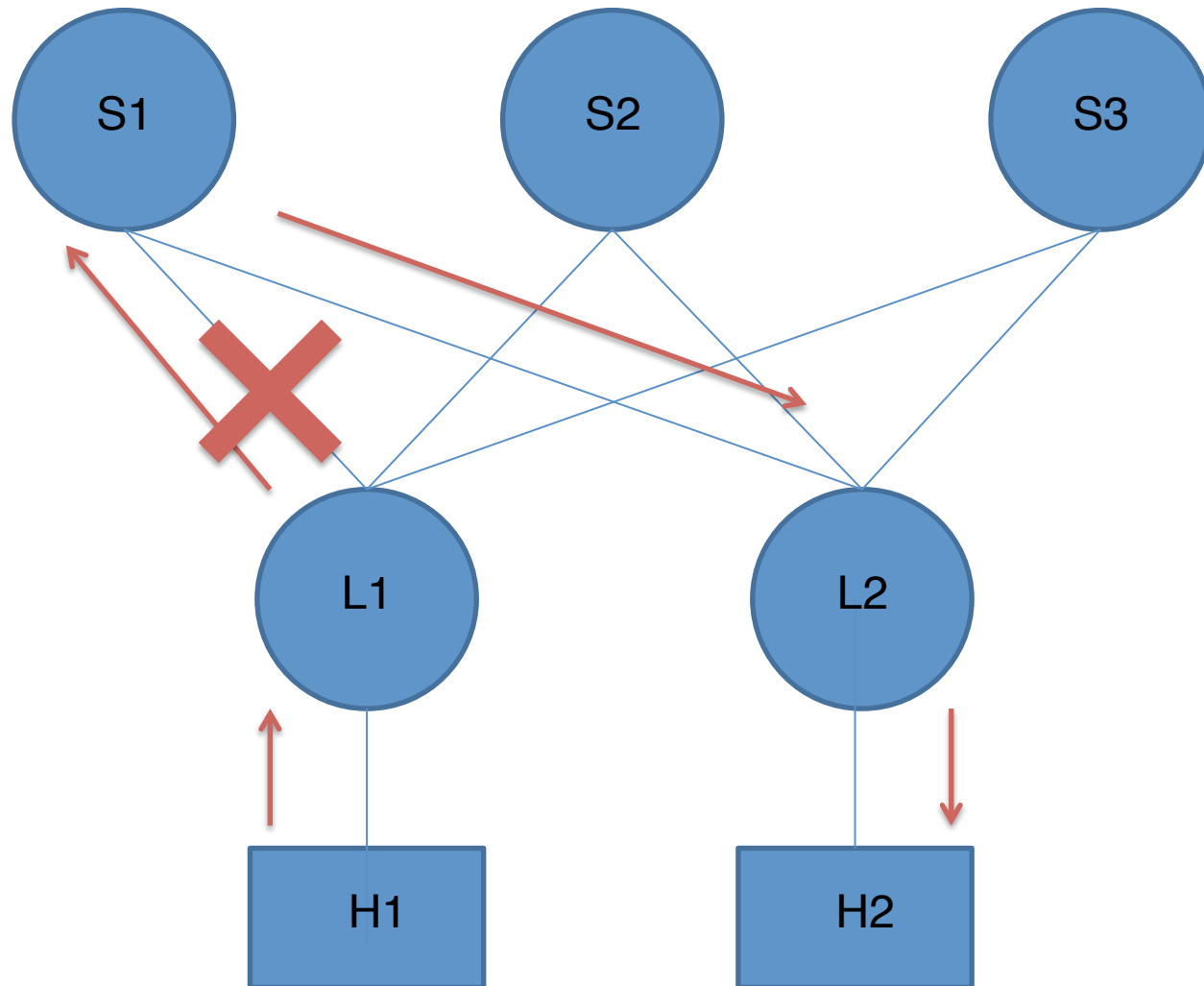
h2(192.168.2.2)へのベストパスとして
l2(10.1.2.2)経由のパスが選択

Network			
* 10.1.1.0/24			
*>			
* 10.1.2.0/24			
*>			
*> 10.2.1.0/24		0	0 65011 i
* 10.2.2.0/24	10.1.1.2		0 65012 65002 i
*>	10.1.1.2		0 65011 65002 i
* 10.3.1.0/24	10.1.1.2	0	0 65012 i
*>	10.1.1.2	0	0 65011 i
* 10.3.2.0/24	10.1.1.2		0 65012 65003 i
*>	10.1.1.2		0 65011 65003 i
*> 192.168.1.0	10.1.1.2	0	0 65012 i
*> 192.168.2.0	10.1.2.2	0	0 65011 i
		0	0 65012 i

ホスト間の経路



経路の切断



経路の切断

1. 経路を切断

```
%sudo ip netns exec s1 ip link set down dev eth1
```

2. 切断されている(ping が通らない)ことを確認

```
%docker exec -it l1 ping 10.1.1.1
```

~ちょっと待つ~

3. ホスト間の疎通を確認

```
%docker exec -it h1 ping 192.168.2.2
```

4. l1の経路を再確認

```
%docker exec -it l1 telnet localhost bgpd  
leaf1>show ip bgp
```

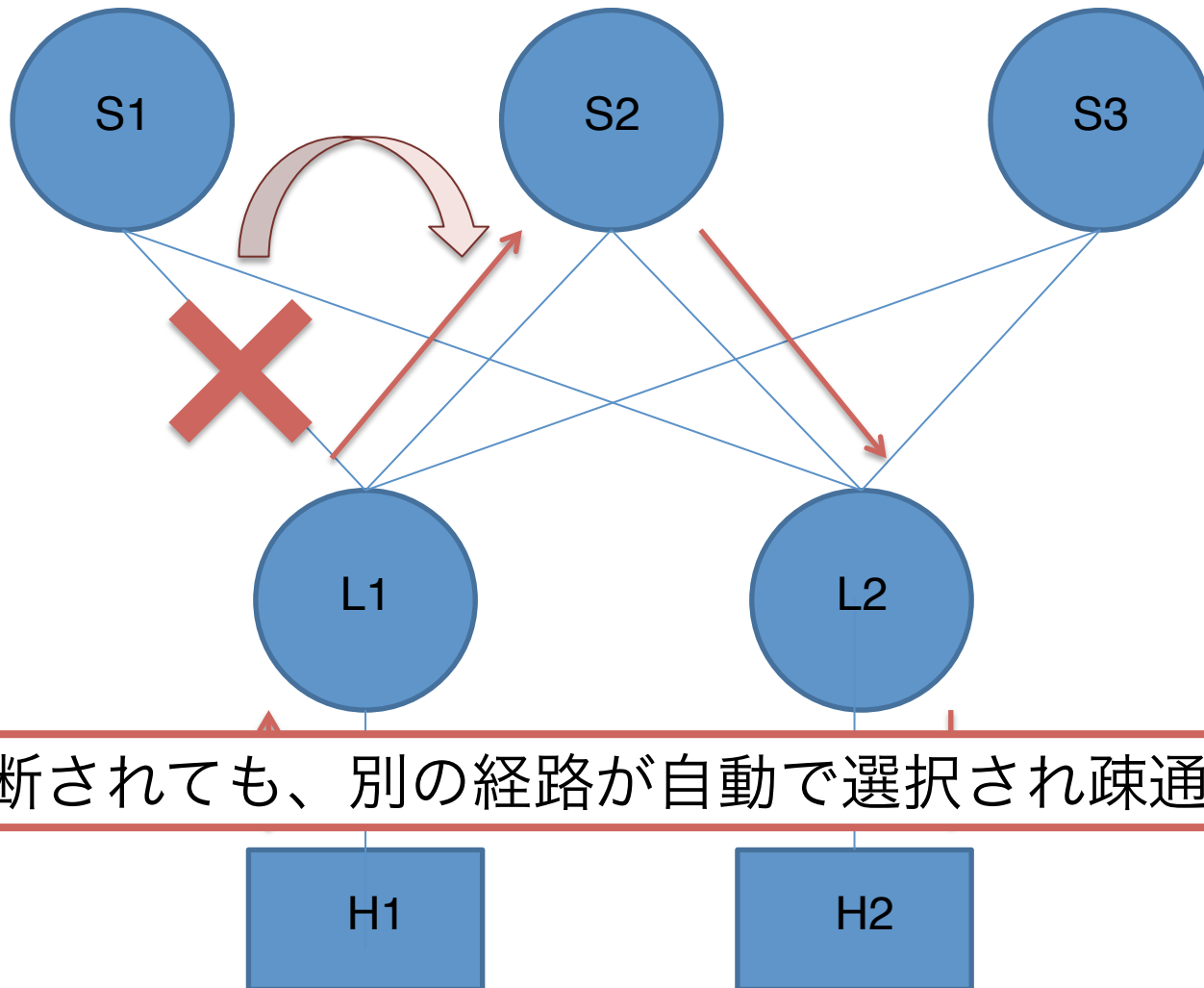
経路の切断

I1の経路

h2(192.168.2.2)への経路として
s1(10.1.1.1)が削除され、
s2(10.2.1.1)が選択

Network					
*>	10.1.1.0/24				
*	10.1.2.0/24				
*>	10.2.1.0/24				
*	10.2.2.0/24	10.1.1.1	0	65003	65012 i
*>	10.3.1.0/24	10.1.1.1	0	65002	i
*	10.3.2.0/24	10.1.1.1	0	65003	i
*>	192.168.1.0	0.0.0.0	0	32768	i
*	192.168.2.0	10.3.1.1	100	0	65003 65012 i
*>		10.2.1.1	200	0	65002 65012 i

経路の切断



経路の復旧

4. 経路を復旧

```
%sudo ip netns exec s1 ip link set up dev eth1  
~ちょっと待つ~
```

5. I1の経路を再確認

```
%docker exec -it I1 telnet localhost bgpd
```

```
*> 192.168.1.0      0.0.0.0      0      32768 i  
*> 192.168.2.0      10.1.1.1      300      0 65001 65012 i  
*      10.3.1.1      100      0 65003 65012 i  
*      10.2.1.1      200      0 65002 65012 i
```

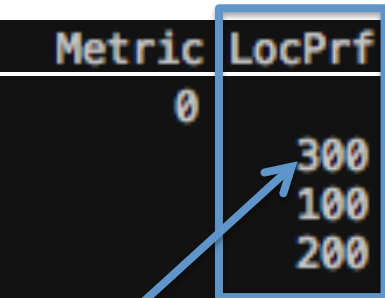
s1(10.1.1.1)への経路が復活しベストパスも戻る

経路の変更

s1をメンテナンスしたいがホスト間の通信を止めたくない...
一旦ベストパスをs2にしてからs1を止めたい！

実は・・・

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.1.0	0.0.0.0	0		32768	i
*> 192.168.2.0	10.1.1.1		300	0	65001 65012 i
*	10.3.1.1		100	0	65003 65012 i
*	10.2.1.1		200	0	65002 65012 i



LOCAL_PREF属性の値によってベストパスを選択している
s1のLOCAL_PREF値をs2より小さくすることでベストパスを
s2経由に！

経路の変更

1. 設定ファイル編集

bgpの設定ファイル(bgpd.conf)がl1~s3フォルダ以下にある
l1/bgpd.confを以下のように編集

```
@@ -37,7 +37,7 @@  
!  
route-map SPINE_ROUTE1 permit 10  
    match ip address 1  
-    set local-preference 300  
+    set local-preference 100  
!  
!  
route-map SPINE_ROUTE1 permit 20
```

2. 設定の反映

%docker kill -s SIGHUP l1

経路の変更

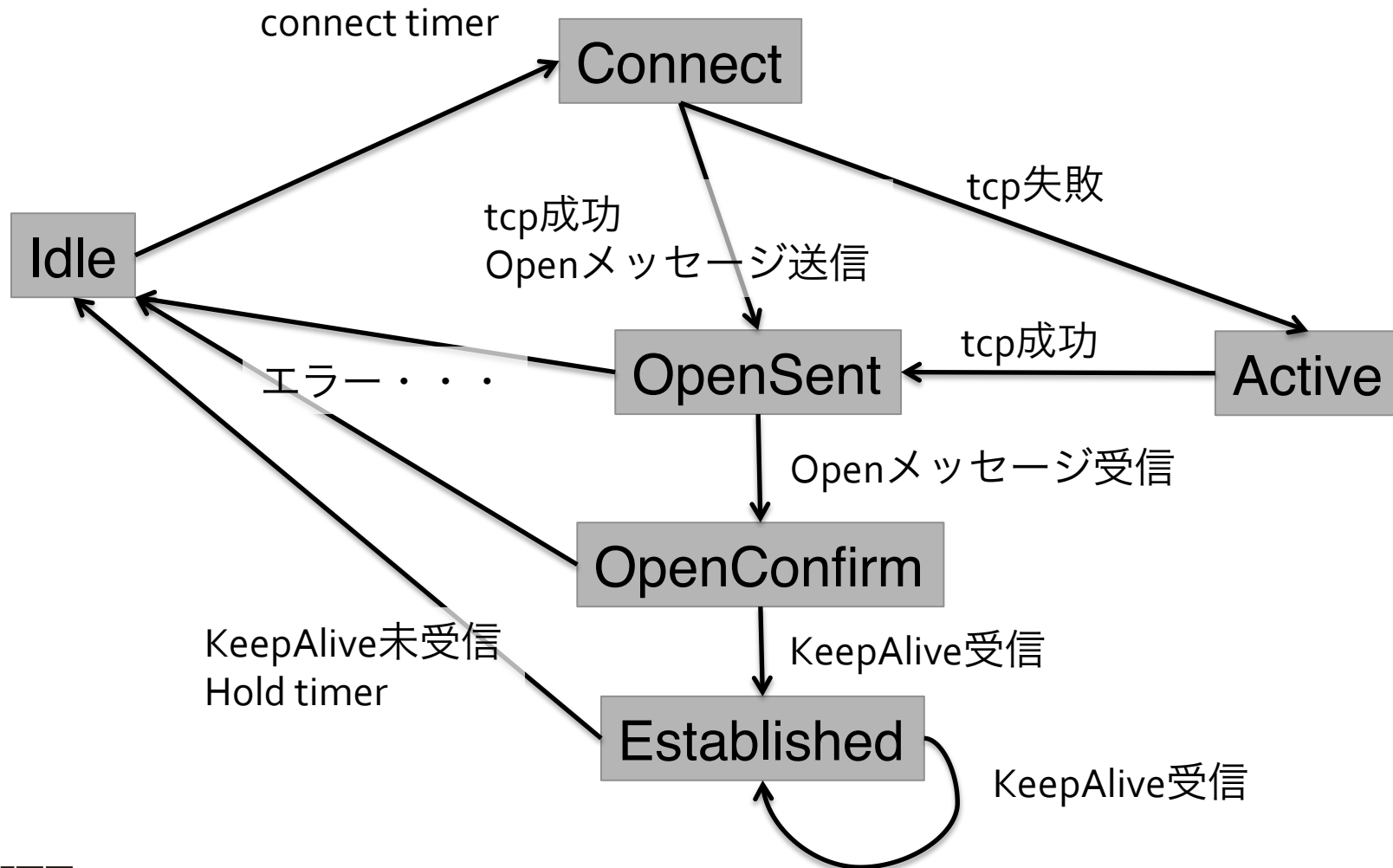
3. 経路確認

```
*> 192.168.1.0      0.0.0.0      0      32768 i
* 192.168.2.0      10.1.1.1      100     0 65001 65012 i
*>      10.2.1.1      200     0 65002 65012 i
*      10.3.1.1      100     0 65003 65012 i
```

s1のLOCAL_PREF値が100になり、ベストパスがs2経由になっている

I2でも同様に設定すればホスト間通信ではs1を通らなくなる
LOCAL_PREF値を元に戻すとベストパスは再びs1経由になる

【参考】 BGPの状態遷移



【参考】経路の変更（手動）

1. ルーターにログイン

```
% docker exec -it l1 telnet localhost bgpd
```

2. 特権モード→設定モードへ移行

```
leaf1>enable           : 特権モードへ
```

```
leaf1#configure t       : 設定モードへ
```

3. 経路属性の変更

```
leaf1(config)#route-map SPINE_ROUTE1 permit 10
```

```
leaf1(config-route-map)# match ip address 1
```

```
leaf1(config-route-map)# set local-preference 100
```

```
leaf1(config-route-map)# exit （2回）
```

4. 経路属性更新の反映

```
leaf1#clear ip bgp * soft in
```