Dockerで箱庭実験ネットワークを作る

NTTソフトウェアイノベーションセンタ
石井久治

事前準備

https://github.com/osrg/ryu-handson
の説明に従い、必要なソフトをインストールして下さい

```
$ sudo apt-get install -y --force-yes git
$ git clone https://github.com/osrg/ryu-handson.git
$ cd ryu-handson/dc-handson
$ ./dc-handson.sh install
```

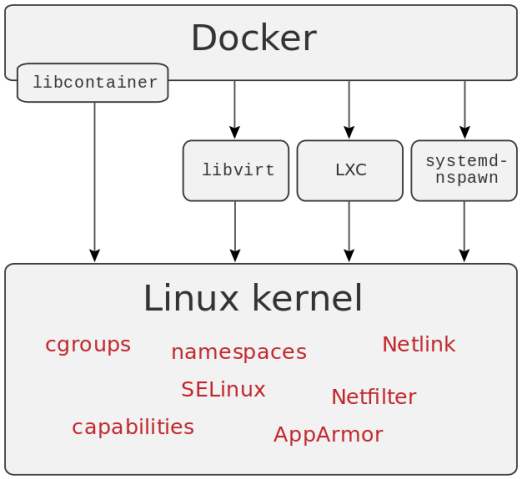
インストール後は、一度ログアウトして再ログインして下さい。
（dockerグループへの所属を反映させるため）

```
$ docker version
Client version: 1.3.2
Client API version: 1.15
Go version (client): go1.3.3
Git commit (client): 39fa2fa
OS/Arch (client): linux/amd64
Server version: 1.3.2
Server API version: 1.15
Go version (server): go1.3.3
Git commit (server): 39fa2fa
```

"docker version"を実行して
"Server version"が
表示されればOK

Dockerとは

Docker:
 LXCと同等のコンテナを簡単にデプロイできるツール



Linuxコンテナ（LXC）:
 Linuxカーネルの機能を用いたOSレベル仮想化技術
関係するカーネル機能（の一部）:
 cgroups:
  プロセスグループのリソース（CPU,メモリ,I/Oなど）利用量を制限する機能
 namespace（名前空間）:
  独立した空間内にリソースを隔離し、外からは見えないようにする機能
  名前空間の種類: IPC, Network, Mount, PID, User, UTS
 capability:
  root権限の一部だけをプロセスに与える機能

dockerのコマンド

docker images	イメージの一覧表示
docker pull	イメージをダウンロード
docker ps	コンテナの一覧表示
docker run	コンテナを起動
docker attach	コンテナの擬似端末に接続
docker exec	コンテナ内でプログラム実行
docker kill	コンテナを停止
docker rm	コンテナを削除

docker images

Innovative R&D by NTT

ローカルマシン上のdockerイメージを一覧表示する

```
ishii@localhost:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED      VIRTUAL SIZE
osrg/quagga   latest    0bbfdd10be15   3 days ago   250.1 MB
ubuntu        trusty    5ba9dab47459   4 days ago   192.7 MB
ubuntu        14.04     5ba9dab47459   4 days ago   192.7 MB
ubuntu        14.04.1   5ba9dab47459   4 days ago   192.7 MB
ubuntu        latest    5ba9dab47459   4 days ago   192.7 MB
```

NTT

docker pull

DockerHubからイメージをダウンロードする

$ docker pull イメージ名

```
$ docker pull osrg/quagga
Pulling repository osrg/quagga
0bbfdd10be15: Download complete
511136ea3c5a: Download complete
27d47432a69b: Download complete
5f92234dcf1e: Download complete
51a9c7c1f8bb: Download complete
5ba9dab47459: Download complete
63e5db713a57: Download complete
4de04be4a9d5: Download complete
086031e5b0fd: Download complete
b5dd50cd3e34: Download complete
Status: Downloaded newer image for osrg/quagga:latest
```

DockerHubから
イメージ"osrg/quagga"
をダウンロード

docker ps

Innovative R&D by NTT

起動中のコンテナを一覧表示
（-a オプション: 停止中のコンテナも表示）

```
$ docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS   NAMES
fbfffddc0f16   osrg/quagga:latest    "/usr/bin/supervisor     43 seconds ago   Up 42 seconds           s2
cf6e521c0b56   osrg/quagga:latest    "/usr/bin/supervisor     43 seconds ago   Up 42 seconds           s1
94f632059bb3   osrg/quagga:latest    "/usr/bin/supervisor     43 seconds ago   Up 42 seconds           l2
c09f0077a905   osrg/quagga:latest    "/usr/bin/supervisor     44 seconds ago   Up 42 seconds           l1
5e71b67ad840   ubuntu:14.04          "/bin/bash"              44 seconds ago   Up 43 seconds           h2
78768b135577   ubuntu:14.04          "/bin/bash"              44 seconds ago   Up 43 seconds           h1
```

NTT

docker run

コンテナを起動

docker run [オプション] イメージ名 [コマンド]

オプション:

--name="コンテナ名"	コンテナ名を設定
--privileged=true	（ネットワーク設定を含む） 全ての操作をコンテナに許可
--net=none	dockerによるveth接続を無効化
-it	擬似端末を作成
-d	デタッチ状態で起動

docker run（続き）

```
$ docker run --name c1 --privileged=true --net=none -it ubuntu:14.04 bash
root@7adb95899650:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 07:49 ?        00:00:00 bash
root        16     1  0 07:50 ?        00:00:00 ps -ef
root@7adb95899650:/# exit
exit

$ docker ps -a
CONTAINER ID  IMAGE         COMMAND  CREATED         STATUS                     PORTS  NAMES
7adb95899650  ubuntu:14.04  "bash"   51 seconds ago  Exited (0) 4 seconds ago          c1
$ docker rm c1
c1
```

コンテナを起動して擬似端末にアタッチ

擬似端末上でコマンド実行

コンテナを終了

コンテナを削除

コンテナはExited(終了)状態

docker attach

```
$ docker run --name c1 --privileged=true --net=none -itd ubuntu:14.04 bash
58ea82db23e5b9e57308fc7906dc03471ef759a7ece4814517f63840cdadff42
$ docker ps
CONTAINER ID  IMAGE         COMMAND  CREATED         STATUS         PORTS  NAMES
58ea82db23e5  ubuntu:14.04  "bash"   12 seconds ago  Up 11 seconds         c1
$ docker attach c1

root@58ea82db23e5:/# echo hoge
hoge
root@58ea82db23e5:/# $
$
$ docker ps
CONTAINER ID  IMAGE         COMMAND  CREATED         STATUS         PORTS  NAMES
58ea82db23e5  ubuntu:14.04  "bash"   21 seconds ago  Up 20 seconds         c1
$ docker attach c1

root@58ea82db23e5:/# echo fuga
fuga
root@58ea82db23e5:/# exit
exit
$ docker ps -a
CONTAINER ID  IMAGE         COMMAND  CREATED         STATUS                     PORTS  NAMES
58ea82db23e5  ubuntu:14.04  "bash"   54 seconds ago  Exited (0) 4 seconds ago         c1
$ docker rm c1
c1
```

アタッチせずにコンテナを起動

コンテナはUp(起動中)状態

コンテナにアタッチ

「CTRL+P CTRL+Q」を押してコンテナからデタッチ

コンテナに再アタッチ

コンテナを終了

コンテナを削除

docker exec

起動中のコンテナ内でプログラムを実行

$ docker exec [オプション] コンテナ名 コマンド [引数]

オプション:　-it　　擬似端末を作成

```
$ docker run --name c1 --privileged=true --net=none -itd ubuntu:14.04 bash
812222ea1e7864d86cfbb50580423cc73a82a35fad9d45ae42f79bb5199e3640

$ docker exec c1 touch /tmp/hoge
$
$ docker exec -it c1 bash
root@812222ea1e78:/# ls /tmp
hoge
root@812222ea1e78:/# exit
exit
$ docker ps
CONTAINER ID   IMAGE          COMMAND   CREATED              STATUS              PORTS   NAMES
812222ea1e78   ubuntu:14.04   "bash"    About a minute ago   Up About a minute           c1
$ docker kill c1
c1
$ docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED              STATUS                      PORTS   NAMES
812222ea1e78   ubuntu:14.04   "bash"    About a minute ago   Exited (-1) 9 seconds ago           c1
$ docker rm c1
c1
```

アタッチせずにコンテナ内でtouchコマンドを実行

シェルを起動してアタッチ

コンテナは
Up(起動中)状態

コンテナを停止

コンテナはExited(終了)状態

docker kill / docker rm

$ docker kill [オプション] コンテナ名

コンテナ起動時の実行プログラムをkill

オプション:　　-s シグナル名　　送信するシグナルを指定

$docker rm [オプション] コンテナ名

コンテナを削除

オプション:　　-f　　コンテナが起動中の場合も強制削除

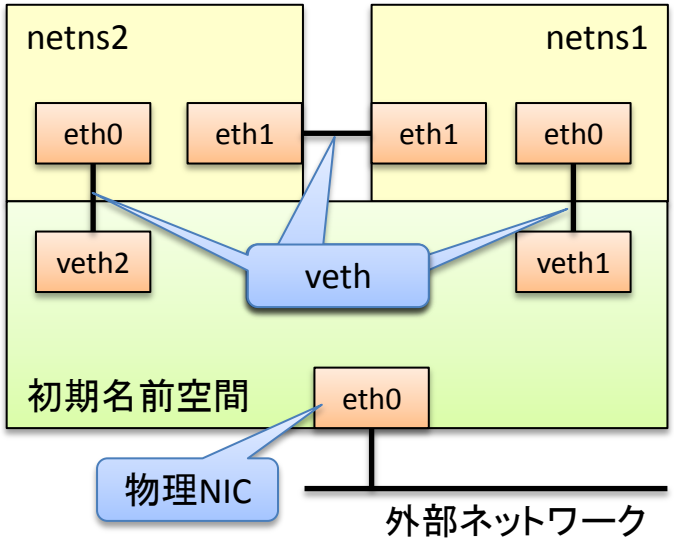
ネットワーク名前空間とveth

ネットワーク名前空間（netns）:

ネットワークに関連するリソースを分離する名前空間

分離されるリソース:

- ネットワークデバイス（NIC）
- IPv4/IPv6プロトコルスタック
- IPルーティングテーブル
- ファイアウォール（iptables）
- TCP/UDPポート



veth（virtual ether）:

EthernetのNICを２枚作成し、クロスケーブルで接続**したかのように**エミュレートする仮想デバイス

iproute2のコマンド

Innovative R&D by NTT

ネットワークデバイス	
ip link show	NICを一覧表示
ip link set	NICの設定を変更
ip link add / ip link delete	NIC(veth)の作成・削除
IPアドレス	
ip addr show	IPアドレスの表示
ip addr add / ip addr delete	IPアドレスの設定・削除
ネットワーク名前空間	
ip netns list	名前空間の一覧表示
ip netns add / ip netns delete	名前空間の作成・削除
ip netns exec	名前空間内でプログラム実行

NTT

ip link set

ネットワークデバイス(NIC)の各種設定変更

ip link set dev デバイス名 [各種設定]

各種設定 :

up	up状態にする
down	down状態にする
name デバイス名	デバイス名を変更する
netns 名前空間名	名前空間に移す

例 :

```
$ sudo ip link set dev eth0 up
$ sudo ip link set dev eth1 name eth2
$ sudo ip link set dev eth0 netns ns1
```

ip link add / ip link delete

vethの作成:

ip link add name デバイス名1 type veth peer name デバイス名2

vethの削除:

ip link delete デバイス名

例:

$ sudo ip link add name eth-v1 type veth peer name eth-v2

$ sudo ip link delete eth-v1

ip addr add / ip addr delete

IPアドレスの設定（追加）:

ip addr add IPアドレス/プレフィックス長 dev デバイス名

IPアドレスの削除:

ip addr delete IPアドレス/プレフィックス長 dev デバイス名

例:

$ sudo ip addr add 192.168.0.1/24 dev eth-v1

$ sudo ip addr delete 192.168.0.1/24 dev eth-v1

ip netns add / ip netns delete

名前空間の作成:

ip netns add 名前空間名

名前空間の削除:

ip netns delete 名前空間名

例:

$ sudo ip netns add ns1

$ sudo ip netns delete ns1

ip netns exec

名前空間内で任意のプログラムを実行

ip netns exec 名前空間名 実行プログラム [引数]

例:

$ ip netns exec ns1 ip addr add 10.0.0.1/24 dev eth-v1

netnsを2個作成してvethで接続する

```
$ sudo ip netns add netns1
$ sudo ip netns add netns2
$ sudo ip link add name veth1 type veth peer name veth2
$ sudo ip link set dev veth1 netns netns1
$ sudo ip link set dev veth2 netns netns2
$ sudo ip netns exec netns1 ip link set dev veth1 up

$ sudo ip netns exec netns1 ip addr add 10.0.0.1/24 dev veth1

$ sudo ip netns exec netns2 ip link set dev veth2 up
$ sudo ip netns exec netns2 ip addr add 10.0.0.2/24 dev veth2
```

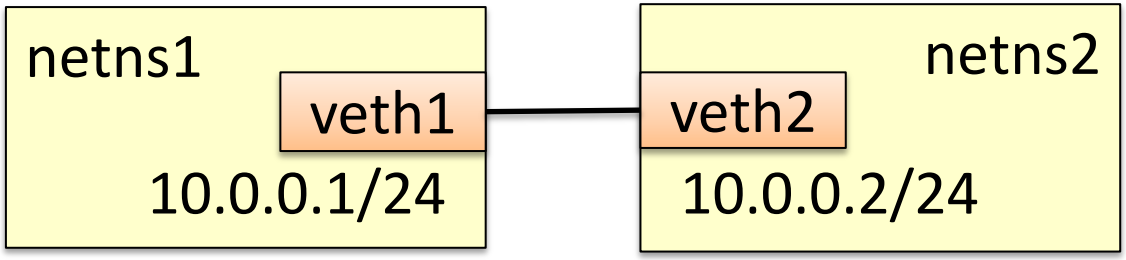
netns1を作成

vethを作成

veth1をnetns1に移動

netns1の中でveth1をup状態に設定

netns1の中でveth1にIPアドレス10.0.0.1/24を設定



（続き）netnsを2個作成してvethで接続する

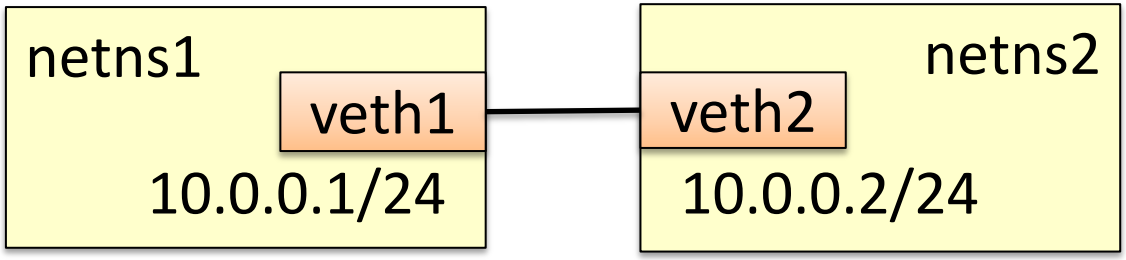
```
$ sudo ip netns exec netns1 ping -c 3 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.116 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.045 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.045/0.076/0.116/0.029 ms
$ sudo ip netns exec netns1 ip link delete veth1
$ sudo ip netns delete netns1
$ sudo ip netns delete netns2
```

netns1の中から veth2に向かってpingを打つ

veth1を削除（veth2も削除される）

netns1を削除

netns2を削除



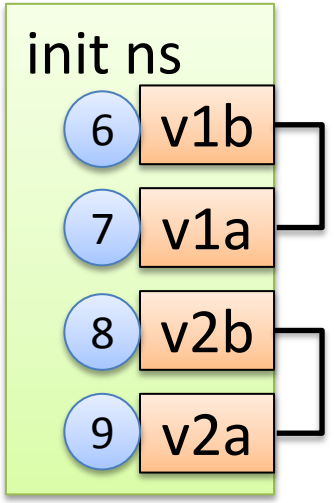
vethの対応関係を調べる方法

```
$ sudo ip link add name v1a type veth peer name v1b
$ sudo ip link add name v2a type veth peer name v2b
$ sudo ethtool -S v1a
NIC statistics:
     peer_ifindex: 6
$ ip link show
...(snip)...
6: v1b: <BROADCAST,MULTICAST> mtu 1500 ...(snip)
    link/ether 16:f3:f2:3f:25:44 brd ff:ff:ff:ff:ff:ff
7: v1a: <BROADCAST,MULTICAST> mtu 1500 ...(snip)
    link/ether 8a:e2:6f:de:69:91 brd ff:ff:ff:ff:ff:ff
8: v2b: <BROADCAST,MULTICAST> mtu 1500 ...(snip)
    link/ether ca:52:73:59:0d:68 brd ff:ff:ff:ff:ff:ff
9: v2a: <BROADCAST,MULTICAST> mtu 1500 ...(snip)
    link/ether 8a:e1:51:e6:d3:d1 brd ff:ff:ff:ff:ff:ff
```

vethを2組作成

v1aの反対側を調べる

対応するデバイスのindexが分かる



（続き）vethの対応関係を調べる方法

```
$ sudo ip netns add ns1b
$ sudo ip link set dev v1b netns ns1b
$ sudo ethtool -S v1a
NIC statistics:
     peer_ifindex: 6
$ sudo ip netns exec ns1b ip link show
...(snip)...
6: v1b: <BROADCAST,MULTICAST> mtu 1500 ...(snip)
    link/ether 16:f3:f2:3f:25:44 brd ff:ff:ff:ff:ff:ff

$ sudo ip link delete v1a
$ sudo ip link delete v2a
$ sudo ip netns delete ns1b
```

ns1bを作成

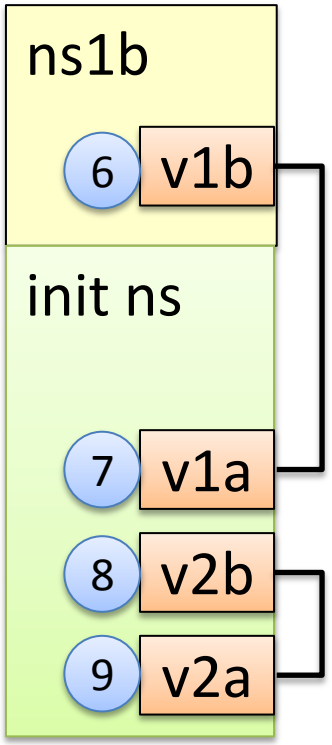
v1bをns1bに移動

デバイスをnetnsに移動しても
indexは変わらない

v1a/v1bを削除

v2a/v2bを削除

ns1bを削除



dockerの名前空間をip netnsで使う方法

```
$ docker run --name c1 --privileged=true -itd ubuntu:14.04 bash
2f8fcfee8947c64d9b7bbcaee58c1c58cfa4d3137cc562117a8080c0
$ ip netns list
$
$ docker inspect -f '{{.State.Pid}}' c1
28765
$ sudo ln -s /proc/28765/ns/net /var/run/netns/c1
$ ip netns list
c1
$ sudo ip netns exec c1 ip link
...(snip)...
12: eth0: <BROADCAST,UP,LOWER_UP> mtu 1460 ...(snip)
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
$ docker rm -f c1
c1
$ sudo rm /var/run/netns/c1
```

コンテナc1を起動

コンテナを起動しただけでは netnsは見えない

コンテナc1のプロセスIDを調べる

コンテナc1の netnsを表すinodeへの シンボリックリンクを /var/run/netnsに作成する

コンテナc1のnetnsが 見えるようになる

コンテナc1のnetns内で 任意のコマンドが実行できる

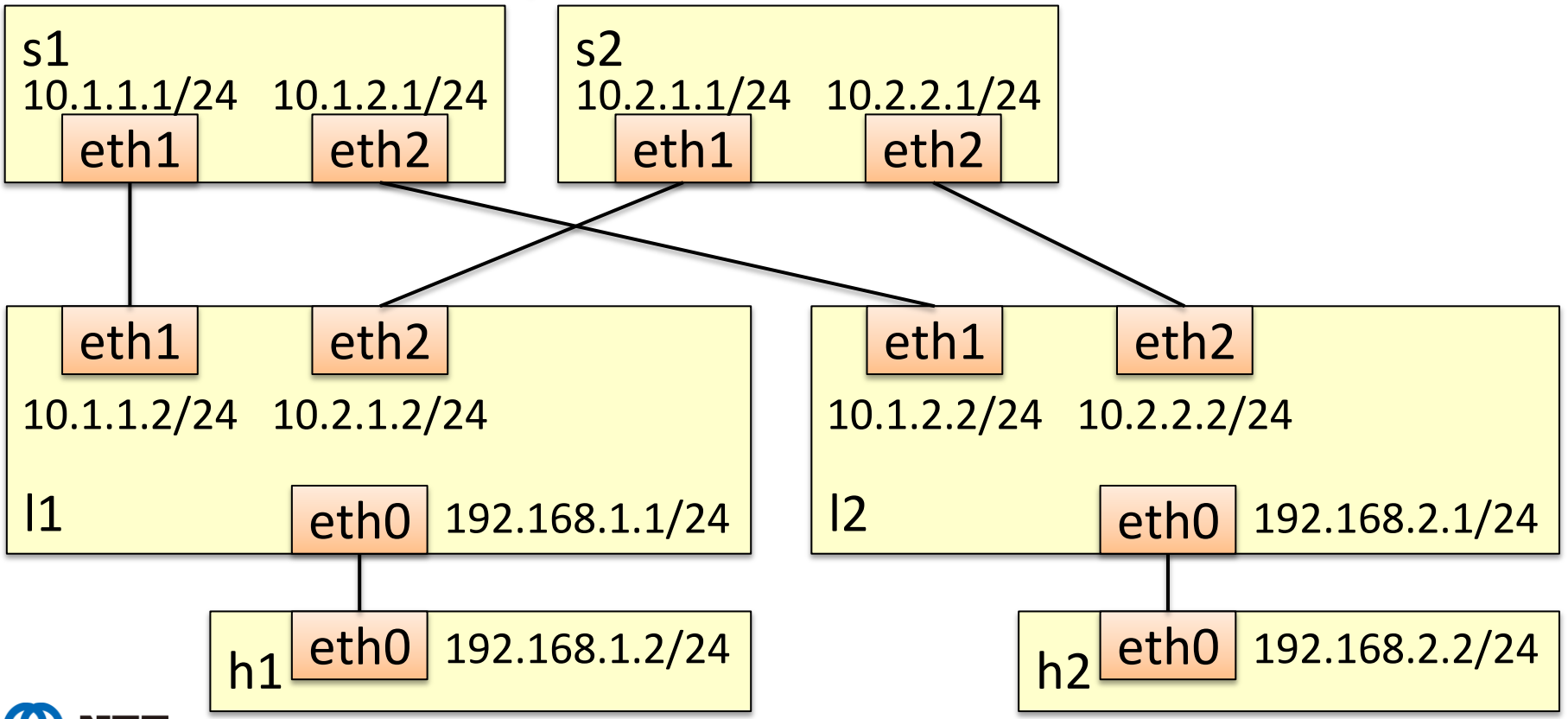
コンテナc1を停止・削除

手動で作成した シンボリックリンクは削除しておく

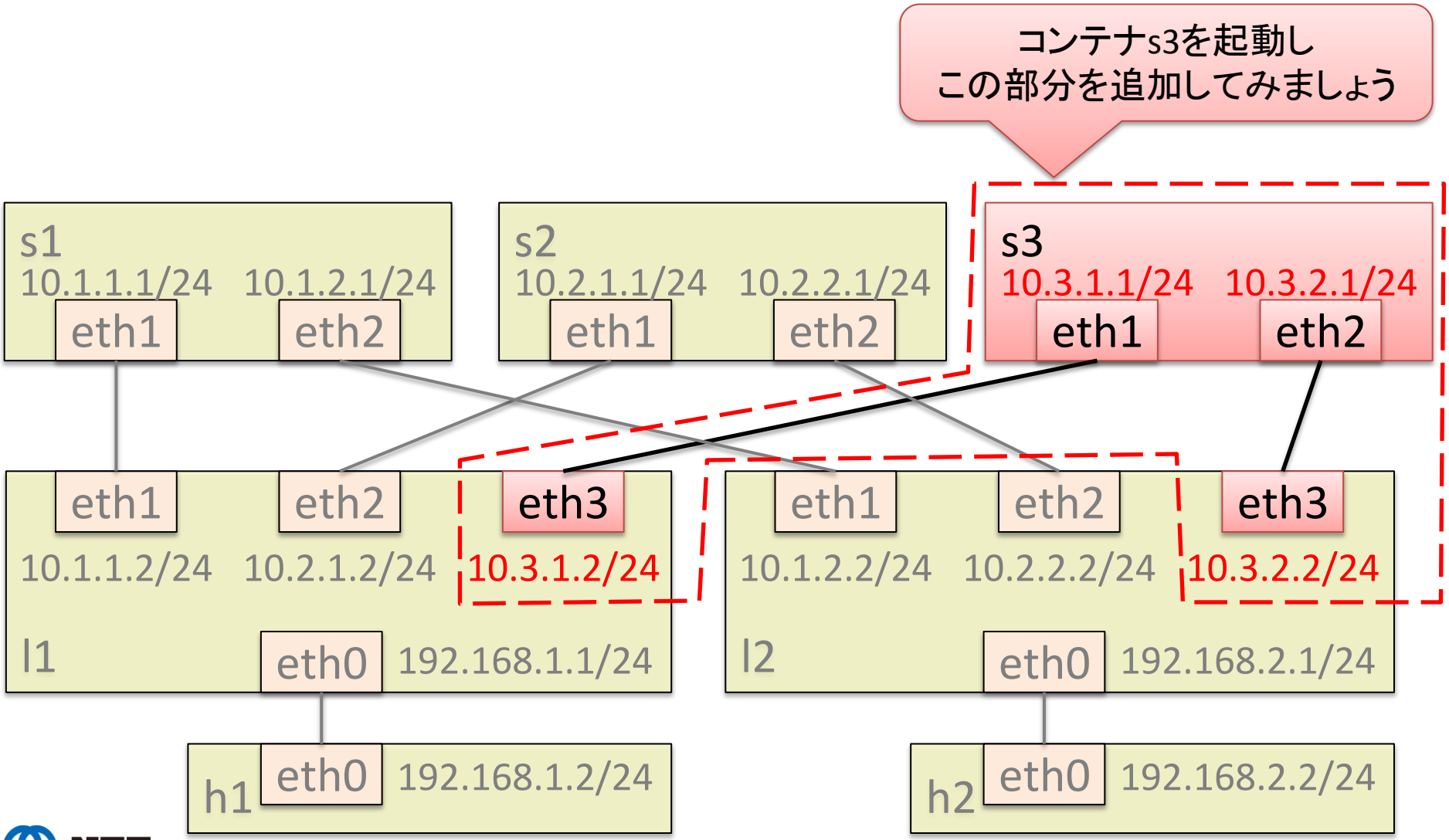
leaf-spine型のDCネットワークを構築

```
$ ./dc-handson.sh start
f47b9fee66d6a33e2d441f8d3dafb3579afbf945a05bffb88023030012e7de10
...(snip)
```

このようなネットワークができます
ip netns exec で確認してみましょう



3台目のspineルータを足す

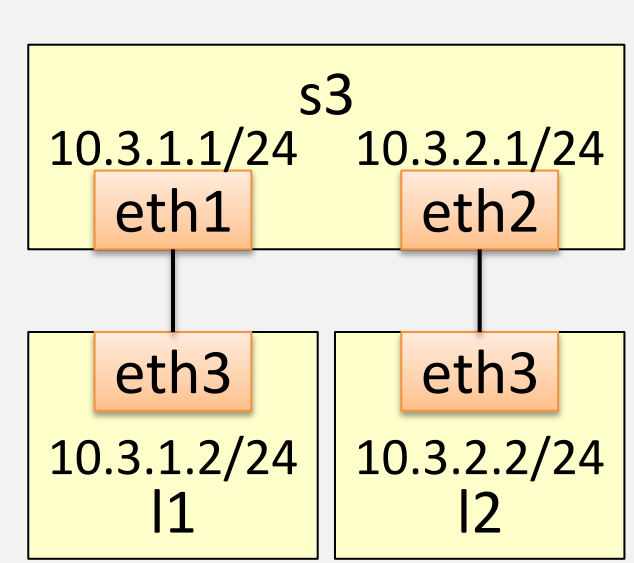


（実行例）3台目のspineルータを足す

```
$ docker run --name s3 --privileged=true --net none -v $PWD/s3:/etc/quagga -itd osrg/quagga
0c87e1adcc559f3ad25637bf5c8a05e07d65e24cfc9c9f9d2df183e333eba2b9
$ docker inspect -f '{{.State.Pid}}' s3
31812
$ sudo ln -s /proc/31812/ns/net /var/run/netns/s3
$ sudo ip link add name s3-eth1 type veth peer name l1-eth3
$ sudo ip link add name s3-eth2 type veth peer name l2-eth3
$ sudo ip link set dev s3-eth1 netns s3
$ sudo ip link set dev s3-eth2 netns s3
$ sudo ip link set dev l1-eth3 netns l1
$ sudo ip link set dev l2-eth3 netns l2
$ sudo ip netns exec s3 ip link set dev s3-eth1 name eth1
$ sudo ip netns exec s3 ip link set dev s3-eth2 name eth2
$ sudo ip netns exec l1 ip link set dev l1-eth3 name eth3
$ sudo ip netns exec l2 ip link set dev l2-eth3 name eth3
$ sudo ip netns exec s3 ip link set dev eth1 up
$ sudo ip netns exec s3 ip link set dev eth2 up
$ sudo ip netns exec l1 ip link set dev eth3 up
$ sudo ip netns exec l2 ip link set dev eth3 up
$ sudo ip netns exec s3 ip addr add 10.3.1.1/24 dev eth1
$ sudo ip netns exec s3 ip addr add 10.3.2.1/24 dev eth2
$ sudo ip netns exec l1 ip addr add 10.3.1.2/24 dev eth3
$ sudo ip netns exec l2 ip addr add 10.3.2.2/24 dev eth3
```

デバイス名(eth3)が
衝突するので
一旦別名で作成する

netns内で
デバイス名を変更する



（補足）3台目のspineルータを足す

```
$ ./dc-handson.sh stop
f1fb37460666
996abb939cdb
beb68e57f1a1
280ae1564fa5
92fb42be570a
7ffe613172f4
$ ./dc-handson.sh start --s3
47b87f5244c91bd7d2c95df8796ce79e7764d46b2ef4f8e89
a2af04882d12ff9decd697152a59fb384707eb45e0365522d
b7c0c236b2225b49da18a9a69bd7421db4eed783c3db85e7ccaa
144ca07de7ddadf8da59dfbb37301a691dd109cdde63c508e5d6f45390bf3367
ac1c4a0b103fb2bb4a8687af3586056b4352fca0146bec3f5b77fcaa1f3f0f12
4e49d61795ccab04099473e4457e77ef689b62b3f9ea9734141e5fb4919914af
2f1d8ac5de0c090705edee5d7aa4beae5a2997e55f2a278f90719f67880a7f7c
$ ip netns
s3
s2
s1
l2
l1
h2
h1
```

s3の作成が上手くいかなかった場合は
次のBGPパートが始まる前に
一度 "./dc-handson stop" で
コンテナを削除した後
"./dc-handson start --s3" で
環境構築を行って下さい

参考資料

Innovative R&D by NTT

Docker　https://www.docker.com/

　Command Line　https://docs.docker.com/reference/commandline/cli/

Dockerを支える技術　http://www.slideshare.net/enakai/docker-34668707

LinuxContainers　https://linuxcontainers.org/ja/

namespaces - Linux 名前空間の概要

http://linuxjm.sourceforge.jp/html/LDP_man-pages/man7/namespaces.7.html

IPROUTE2 Utility Suite Howto

http://www.policyrouting.org/iproute2.doc.html

NTT

https://github.com/osrg/ryu-handson