NTT


# Lagopus Router
# ハンズオンセッション

NTT未来ねっと研究所　市川

2018年7月

# Agenda

- 本日の目標の共有
- ハンズオンサーバへのアクセス
- ハンズオン
  - セットアップ
    - DPDK環境のインストール
    - Lagopus Routerのインストール
  - 起動
    - DPDKの設定
    - Lagopus Routerの起動
  - テスト
    - スイッチとしての動作
    - ルータとしての動作
    - （時間が余ったらVLANなど）

**分からないことや知りたいことがあれば、いつでも聞いてください**

# 本日の目標

## Lagopus Routerを使って、pingを通すこと

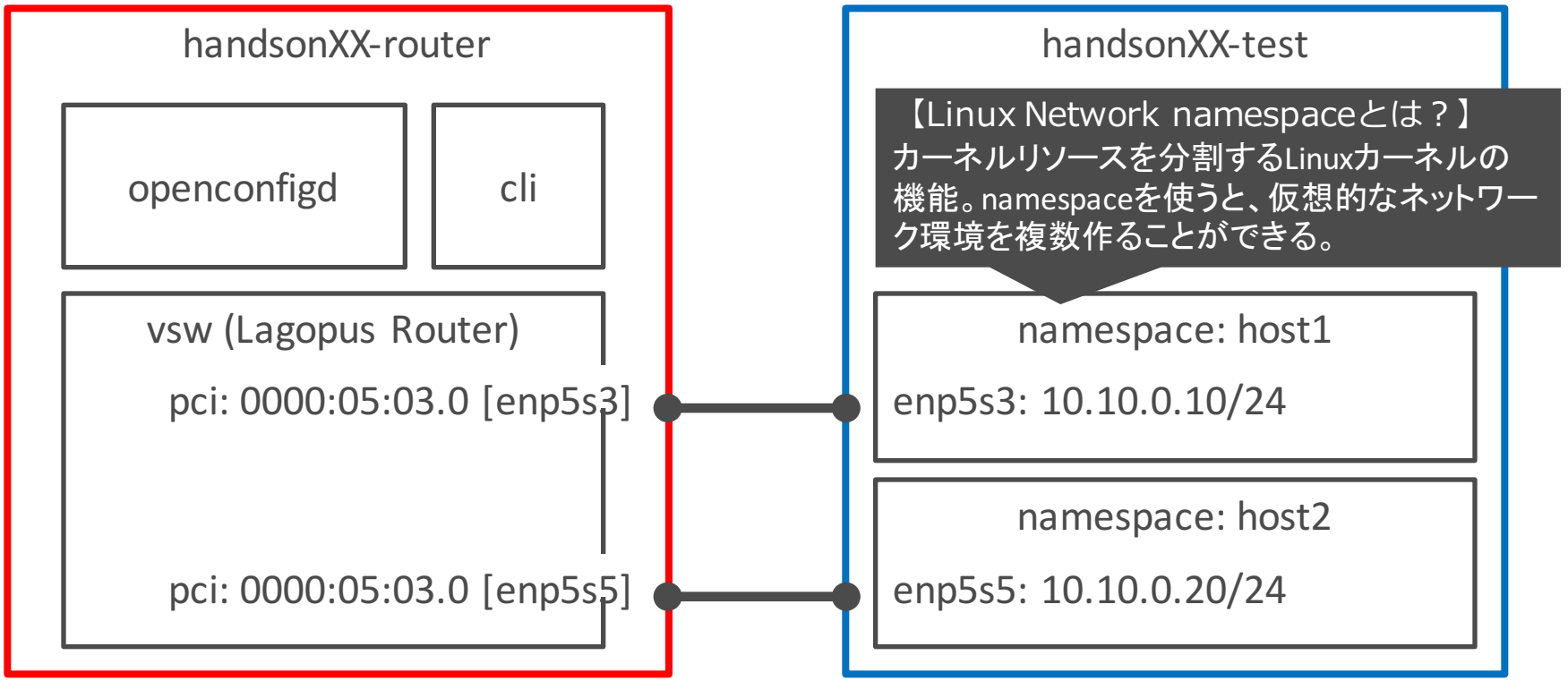
- 皆さんには、一人２台の仮想マシンを用意しました
- そのうちの１台をrouter(Lagopus Router), もう１台をtestとします
- 各仮想マシンはお互いに2本の仮想リンクで接続しています
- Testerの片方のインターフェースからpingを打ち、Lagopus Routerを経由させて、もう片方のインターフェースから応答が返ってくることを確認します
- Lagopus Routerがスイッチ設定の時/ルータ設定の時の２つの状態を試します


【pingとは？】
ネットワークの疎通確認をするために、よく用いられるLinuxコマンド。ピン、ピング
詳しくは、ICMP Echo Request で検索

# テスト１：スイッチ

## testから、同一ネットワークのインターフェースへのping

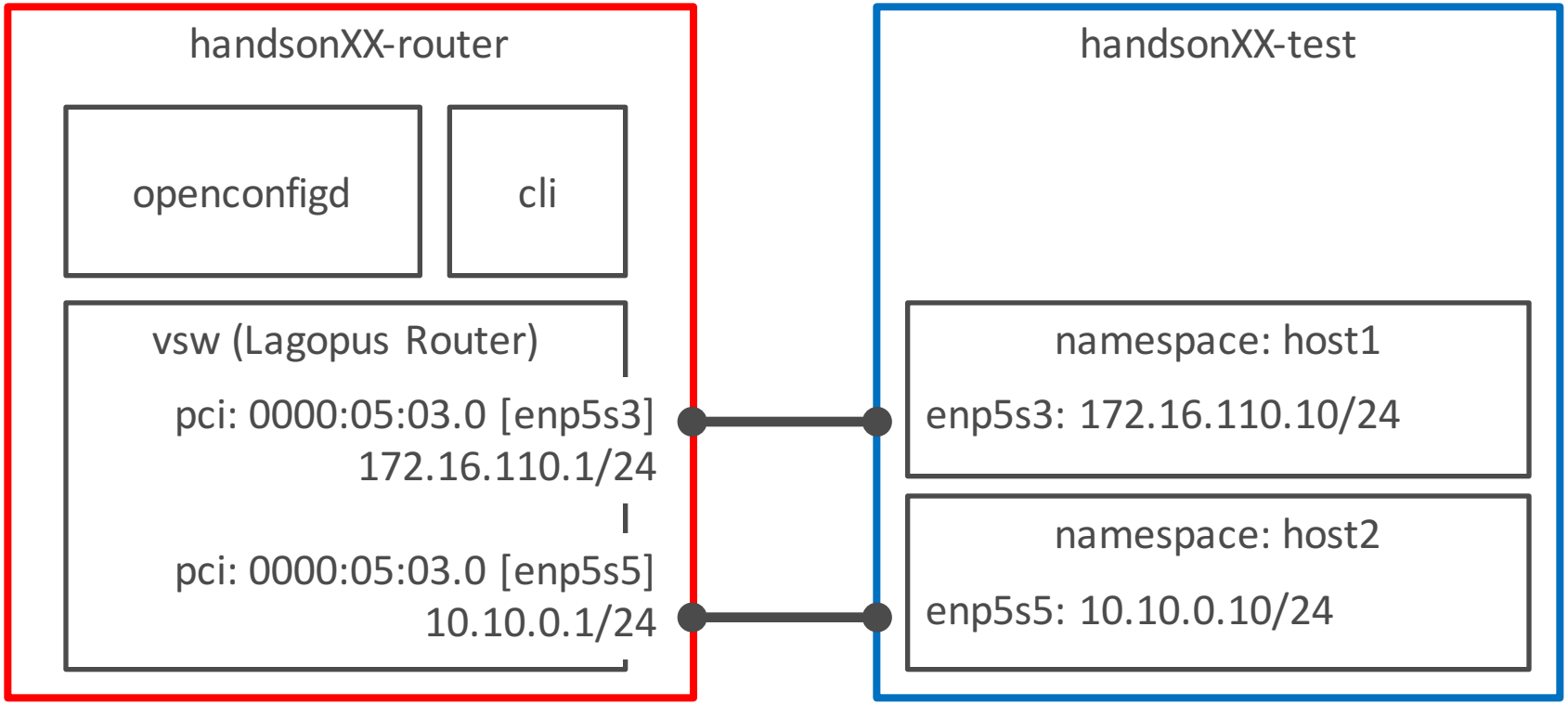
- testにnamespaceを２つ作成し、同一ネットワークのアドレス付与
- 各namespaceからpingを打ち、応答が返ることで疎通確認



# テスト２：ルータ

## testから、異なるネットワークのインターフェースへのping

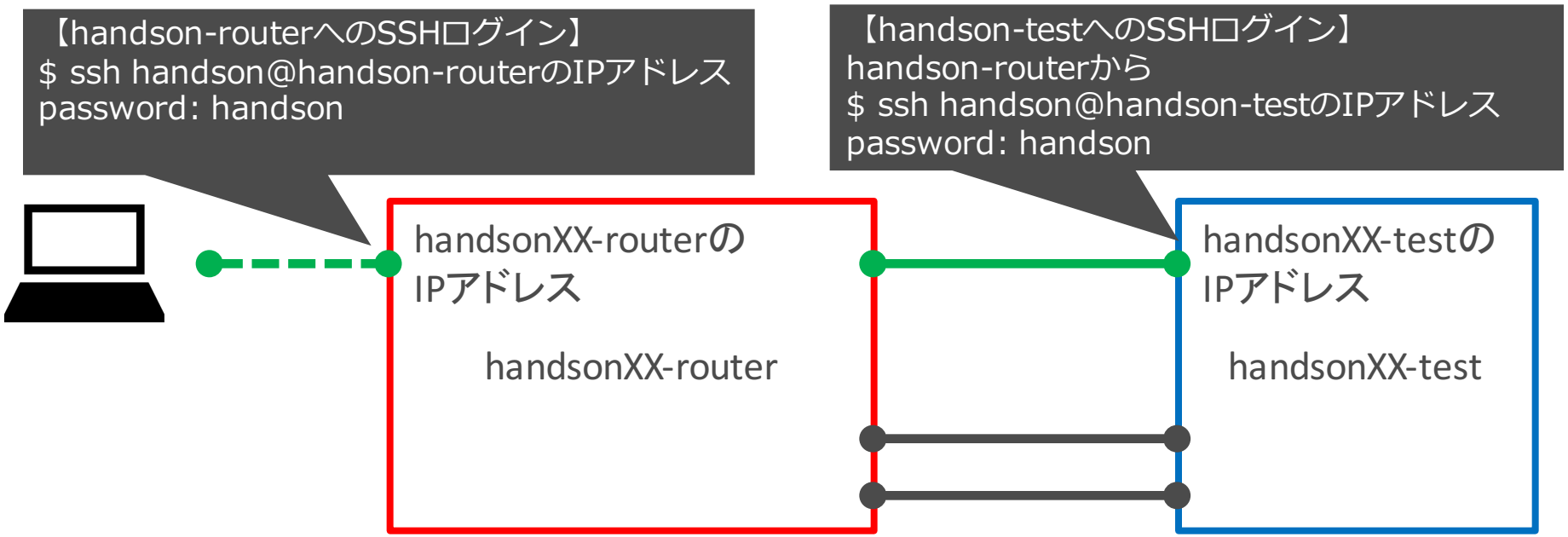
- testにnamespaceを２つ作成し、異なるネットワークのアドレスを付与
- まず、それぞれのdefault-gwからのping応答を試し、さらに対向のnamespaceからのping応答を確認



# ハンズオンサーバへのアクセス

## 2台の仮想マシンにSSHでログインしてみる

- handsonXX-router
  - リストに記載のホスト名@IPアドレス(163.138.X.X)にsshログイン
- handsonXX-test
  - handsonXX-routerにログイン後、handsonXX-router内から、リストに記載のホスト名@IPアドレス(192.168.122.X)にsshログイン



# ハンズオン

# はじめに

- proxyを設定
  - 今回のハンズオン環境では、パッケージの取得などのインターネット接続にプロキシを使います。
  - ただし、Lagopus Routerの実行時には、下記のスクリプトで設定したプロキシの情報は邪魔になるため、無効にしておくことを忘れずに

```
$ cd
$ sudo ./proxy_sw.sh on
$ source .bashrc
```

- インターネット接続性を確認

```
$ curl www.google.com
~
Googleのページがずらずらっと
~
```

# セットアップ：DPDK

- dpdk-stable-17.11.1 を解凍

```
$ cd
$ tar xvf dpdk-stable-17.11.1
```

- configを編集

```
$ cd dpdk-stable-17.11.1/
$ cp config/common_base config/common_base.original
$ vi config/common_base
...
- CONFIG_RTE_BUILD_SHARED_LIB=n
+ CONFIG_RTE_BUILD_SHARED_LIB=y
...
```

- make, install (もしかしたら時間がかかるかも…)

```
$ make T=x86_64-native-linuxapp-gcc config
$ make
$ sudo make install
```

# セットアップ：Go

- golangをインストールし、バージョン確認

```
$ cd
$ sudo apt update
$ sudo apt install golang-go
go version go1.10.1 linux/amd64
```

- dep(Goのバージョン管理ツール)をインストール

```
$ sudo apt install go-dep
```

- パスの設定

```
$ vi .bashrc
...
export GOPATH=$HOME/go
export PATH=$GOPATH/bin:$PATH
$ source .bashrc
```

# セットアップ：openconfigd, cli

- Goパッケージのインストール（今回はスキップ）

```
$ cd
$ go get github.com/coreswitch/openconfigd/openconfigd
$ go get github.com/coreswitch/openconfigd/cli_command
```

- 取得済みパッケージのディレクトリへ移動（既にある）

```
$ cd go/src/github.com/coreswitch/openconfigd/cli
```

- cliのインストール

```
$ ./configure
$ make
$ sudo make install
```

- cliのコピー

```
$ cd ../bash_completion.d
$ sudo cp cli /etc/bash_completion.d/
```

# セットアップ：vsw

- Goのlagopusパッケージを取得（今回はスキップ）

```
$ cd
$ mkdir -p go/src/github.com/lagopus
$ cd go/src/github.com/lagopus
$ git clone https://github.com/lagopus/vsw
```

- vswのインストール

```
$ cd go/src/github.com/lagopus/vsw
$ dep ensure
$ go install
```

– dep ensure すると、vendor/ ディレクトリができることを確認

# セットアップ：lagopus-router

- lagopus-routerパッケージを取得（今回はスキップ）

```
$ cd go/src/github.com/lagopus
$ git clone https://github.com/lagopus/lagopus-router
```

- lagopus-routerパッケージは、現状、make install すべき実行ファイル等が存在せず、設定ファイル(yang)のサンプルのみとなっている

# セットアップ：.conf ファイルの設置

- openconfigd.conf の設置

```
$ sudo touch /usr/local/etc/openconfigd.conf
```

- vsw.conf の設置

```
$ cd ~/go/src/github.com/lagopus/vsw
$ sudo cp vsw.conf /usr/local/etc/
```

- vsw.conf の設定
  - 今回はCPUを４個しか使わないため、cpu_mask を変更
  - 0xfe：1111 1110 … 7, 6, 5, 4, 3, 2, 1番のCPUを使用、という意味
  - 3,2,1番(0番は使わない)のCPUを使用したい場合は？ … 0xe：1110

```
$ sudo vi /usr/local/etc/vsw.conf
...
  [dpdk]
- core_mask = 0xfe
+ core_mask = 0xe
```

# セットアップ：proxyの無効化, reboot

- 今回のハンズオンでのプロキシ設定方法では、openconfigdの実行でerrorを起こしてしまうため、プロキシを無効化する

```
$ cd
$ sudo ./proxy_sw.sh off
```

- その後、いったん再起動

```
$ sudo reboot
```

# 余談：HugePages（スキップでOK）

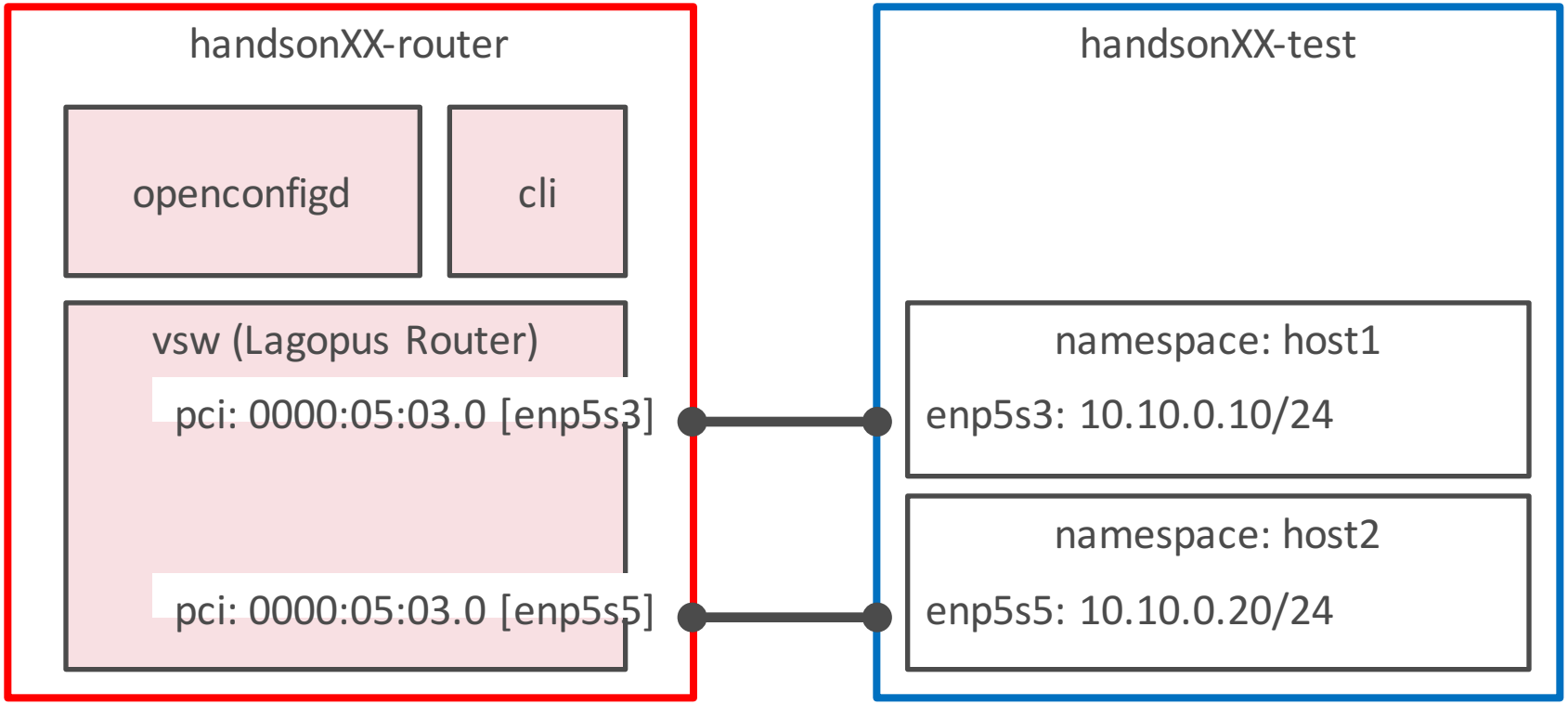
- 今回のハンズオンでは、あらかじめ設定済みですが、DPDKを利用するアプリケーションの実行には、HugePagesというオプションの設定が必要です
- 色々な設定方法がありますが、ここではgrubに記述しておく簡単な手順を紹介します
  - GRUB_CMDLINE_LINUX_DEFAULT, もしくはGRUB_CMDLINE_LINUXへ次のように記述
    - 1GBのHugePageを10page設定する場合

```
$ sudo vi /etc/default/grub
...
GRUB_CMDLINE_LINUX_DEFAULT="default_hugepagesz=1GB
hugepagesz=1GB hugepages=10”
...
$ sudo update-grub
$ sudo reboot
```

# ここまで確認

## Openconfigd, vsw, cli が使える環境の構築が完了

- 各パッケージのインストール
- Hugepageの設定
  - cat /proc/meminfo



# 起動：DPDKの設定（１）

- DPDKで利用するモジュールのロード

```
$ cd ~/dpdk-stable-17.11.1
$ sudo modprobe uio
$ sudo insmod build/kmod/igb_uio.ko
```

- Lagopus Routerで制御したいインターフェースの確認
  - 今回は、handson-testと接続している2本のリンクを用います
  - おそらく、enp5s3，enp5s5 が、link-down状態で存在するはずです
  - これらの存在とPCI番号を確認します
  - 下記の例では、0000:05:03.0，0000:05:05.0 だと確認できます

```
$ ip a
$ sudo lshw –class network –businfo
...
pci@0000:05:03.0  enp5s3  network  ...
pci@0000:05:05.0  enp5s5  network  ...
...
```

# 起動：DPDKの設定（２）

- DPDKによるインターフェースのBind

```
$ sudo usertools/dpdk-devbind.py --bind=igb_uio enp5s3
enp5s5
```

  - 上記コマンドで、enp5s3，enp5s5 をLinux管理下から外し、DPDKにBindすることができます（Linuxからunbind）
  - Lagopus Routerでは、ここでBindしたインターフェースを利用します

# 起動：Openconfigdの実行

- Lagopus Routerでは複数のプロセスを起動します
- 色々な方法があるが、ここではtmuxで簡単に複数のプロセスを起動して、Lagopus Routerを動かしてみましょう
  - ターミナルを複数起動でもOK。その場合はtmux関連を読み飛ばす
- tmuxを起動

```
$ tmux
```

- Openconfigdを起動（tmuxの中で）
  - openconfigd –y 以降のmodulesからtypesまではスペースなしの文字列

```
$ openconfigd -y
modules:modules/policy:modules/bgp:modules/interfaces:mo
dules/local-
routing:modules/vlan:modules/rib:modules/network-
instance:modules/types lagopus-router.yang
```

# 起動：vswの実行

- tmuxでvswのウィンドウを作成

```
Ctrl-B, C
```

– ウィンドウの切替：次 Ctrl-B, N / 前 Ctrl-B, P / ジャンプ Ctrl-B, 番号

- vswを実行

```
$ sudo LD_LIBRARY_PATH=$DPDKDIR/build/lib go/bin/vsw -v
```

# 起動：cliの実行

- tmuxでcliのウィンドウを作成

```
Ctrl-B, C
```

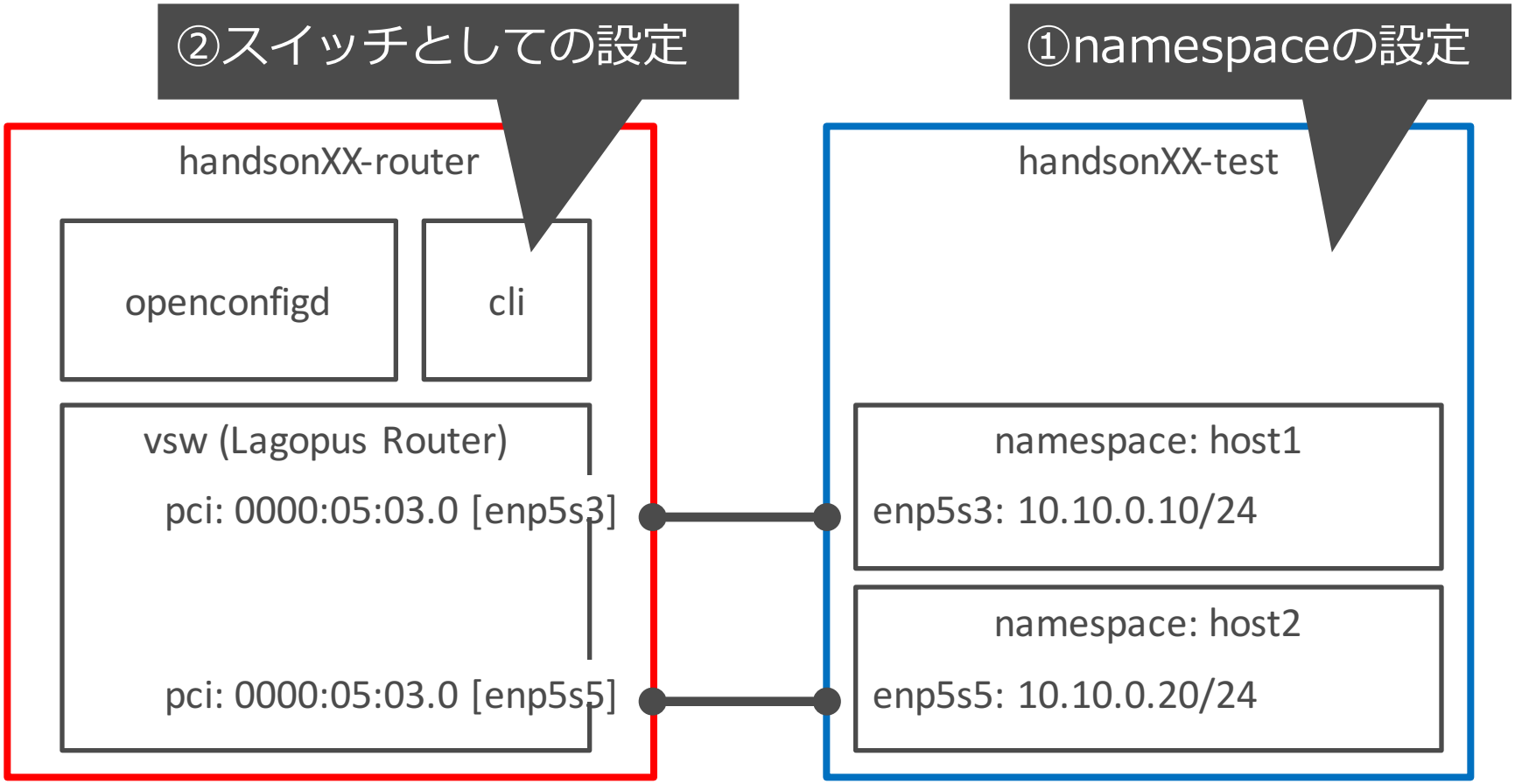
– ウィンドウの切替：次 Ctrl-B, N / 前 Ctrl-B, P / ジャンプ Ctrl-B, 番号

- cliを実行

```
$ cli
handsonXX-router>
```

# テスト１：スイッチ

- 環境は構築できたので、ここからはテストごとの設定と、実際のping疎通実験を行います



# テスト１：スイッチ　設定①

- namespaceの設定
  - handsonXX-testへsshログイン後、下記コマンドで作成

```
$ sudo ip netns add host1
$ sudo ip netns exec host1 ip link set lo up
$ sudo ip link set enp5s3 netns host1 up
$ sudo ip netns exec host1 ip a add 10.0.0.10/24 dev enp5s3

$ sudo ip netns add host2
$ sudo ip netns exec host2 ip link set lo up
$ sudo ip link set enp5s5 netns host2 up
$ sudo ip netns exec host2 ip a add 10.0.0.20/24 dev enp5s5
```

- この時点ではpingを打っても届かない

```
$ sudo ip netns exec host1 ping 10.0.0.20
```

# テスト１：スイッチ　設定②

- Lagopus Routerへのスイッチの設定の投入
  - cliのウィンドウを開く

```
handsonXX-router>configure
handsonXX-router#
```

- コンフィグを手打ちする場合
  - ここで、set interfaces などと設定を投入することで、市販ルータやスイッチのようなコンフィグ作業が可能
  - コンフィグ設定は、下記の場所にある（set～の行がコンフィグ）

```
$ less ~/sample/test_1_l2vsi.sh
```

  - setした後、commitすることで設定を確定する

```
handsonXX-router#commit
```

- 手打ち以外に、このシェルを実行することでも設定可能

```
$ bash ~/sample/test_1_l2vsi.sh
```

# テスト１：スイッチ　テスト

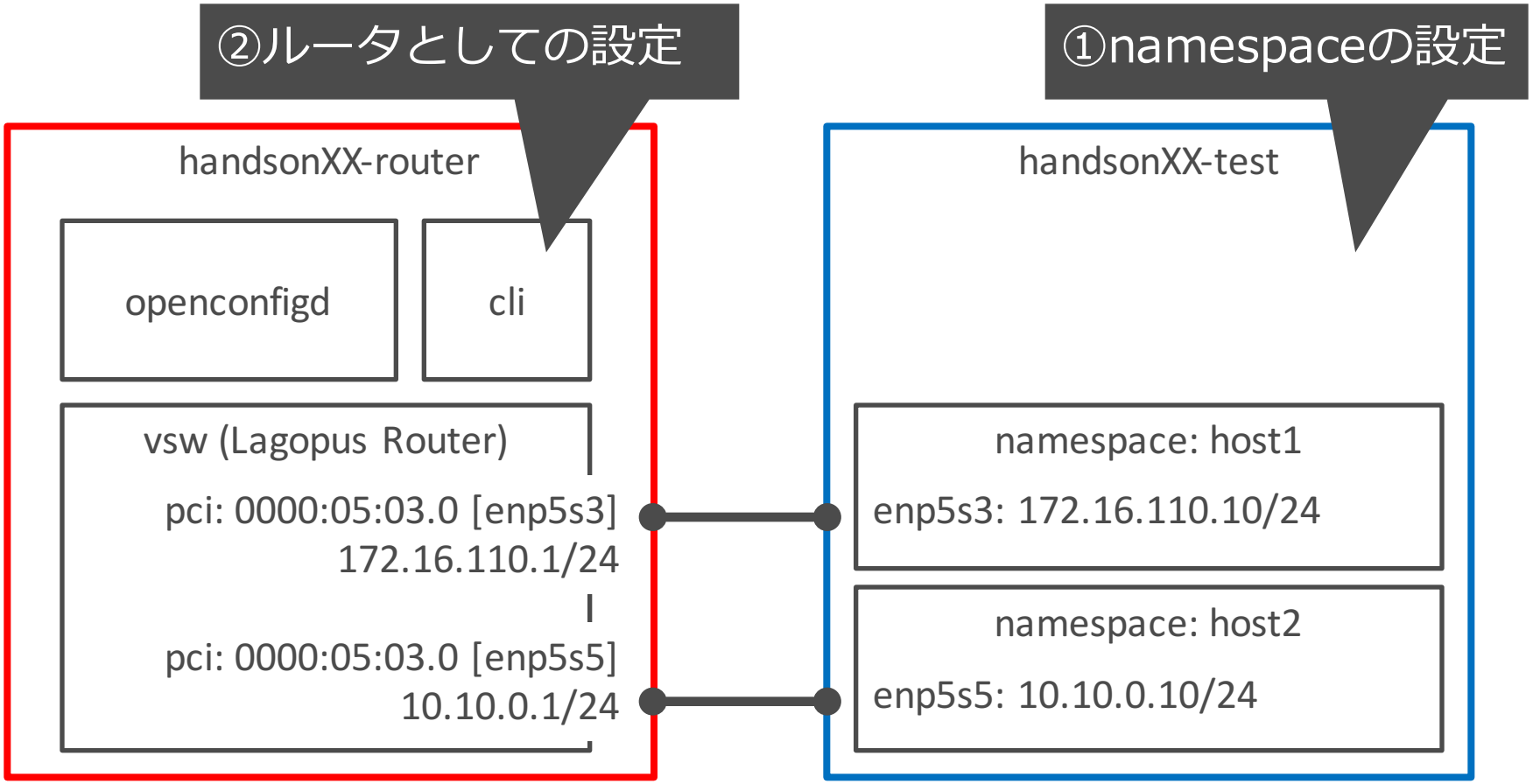
- handsonXX-testのnamespaceからpingを打ってみる

```
$ sudo ip netns exec host1 ping 10.0.0.20

# 反対側へ
$ sudo ip netns exec host2 ping 10.0.0.10
```

- どちらも届くはず

# テスト２：ルータ



# テスト２：ルータ　設定①

- namespaceの設定
  - handsonXX-testへsshログイン後、下記コマンドで作成

```
$ sudo ip netns exec host1 ip a del 10.0.0.10/24 dev enp5s3
$ sudo ip netns exec host1 ip a add 172.16.110.10/24 dev
enp5s3
$ sudo ip netns exec host1 ip r add default via 172.16.110.1

$ sudo ip netns exec host2 ip a del 10.0.0.20/24 dev enp5s5
$ sudo ip netns exec host2 ip a add 10.10.0.10/24 dev enp5s5
$ sudo ip netns exec host2 ip r add default via 10.10.0.1
```

- この時点ではpingを打っても届かない

```
$ sudo ip netns exec host1 ping 172.16.110.1
$ sudo ip netns exec host1 ping 10.10.0.1
$ sudo ip netns exec host1 ping 10.10.0.2
```

# テスト２：ルータ　設定②

- Lagopus Routerへのルータの設定の投入
  - いったん、先ほどのスイッチの設定を一回消去した後、コンフィグを投入

```
handsonXX-router>configure
handsonXX-router#delete interfaces
handsonXX-router#delete network-interfaces
handsonXX-router#commit
handsonXX-router#
```

- コンフィグを手打ちする場合
  - コンフィグ設定は、下記の場所にある（set～の行がコンフィグ）

```
$ less ~/sample/test_2_l3vrf.sh
```

  - setした後、commitすることで設定を確定する

```
handsonXX-router#commit
```

- 手打ち以外に、このシェルを実行することでも設定可能

```
$ bash ~/sample/test_2_l3vrf.sh
```

# テスト２：ルータ　テスト

- handsonXX-testのnamespaceからpingを打ってみる

```
# ルータのhost1側のinterfaceへ
$ sudo ip netns exec host1 ping 172.16.110.1
# ルータのhost2側のinterfaceへ
$ sudo ip netns exec host1 ping 10.10.0.1
# host2のinterfaceへ
$ sudo ip netns exec host1 ping 10.10.0.2

# 反対側も
$ sudo ip netns exec host2 ping 10.10.0.1

$ sudo ip netns exec host2 ping 172.16.110.1

$ sudo ip netns exec host2 ping 172.16.110.2
```

# Q & A