Lecture 4 Mininet

Introduction

- How to maintain connection between two hosts without high cost?
- Easily Utilized on Laptop
- linux命令行 + 进入root用户 (su)

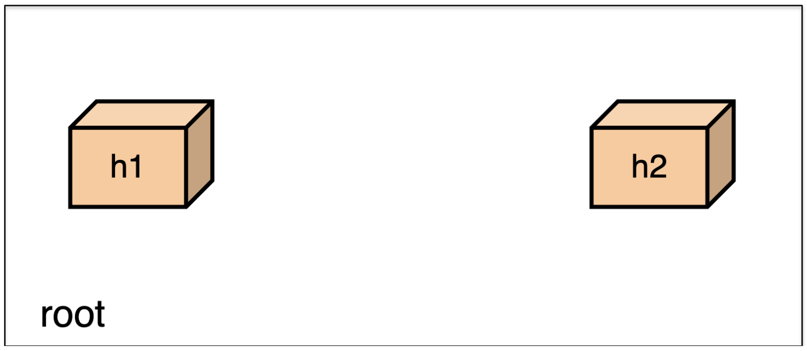
Create Network Namespaces

```bash
ip netns add h1   # 前三个是reserved，h1是自定义的namespace ⇒ linux自建立轻量“虚拟机”
ip netns add h2

ip netns show     # 显示当前所有的名字空间 ⇒ h1 / h2 / root (不会显示)
```

Hence, there are 3 network namespaces in system now

- h1
- h2
- root



Create Virtual Ethernet Pair

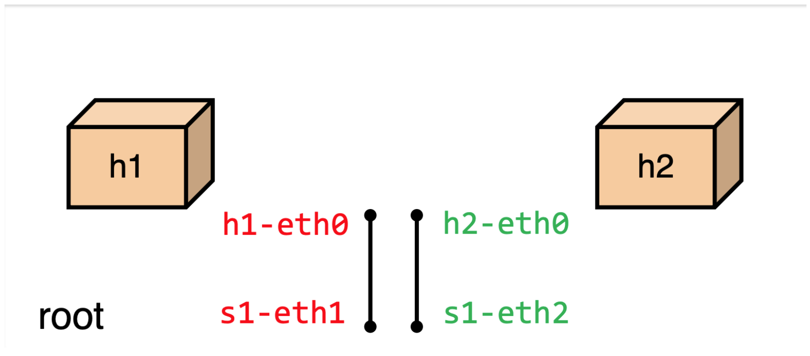
```shell
ip link add h1-eth0 type veth peer name s1-eth1  # h1 → h1-eth0 ⟶-- s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2 # h2 → h2-eth0 ⟶-- s1-eth2

ip link show # 显示在root名字空间内的链路
```

Hence, we created 2 links now in the root namespace

- h1-eth0 ⇒ s1-eth1

- h2-eth0 ⇒ s1-eth2



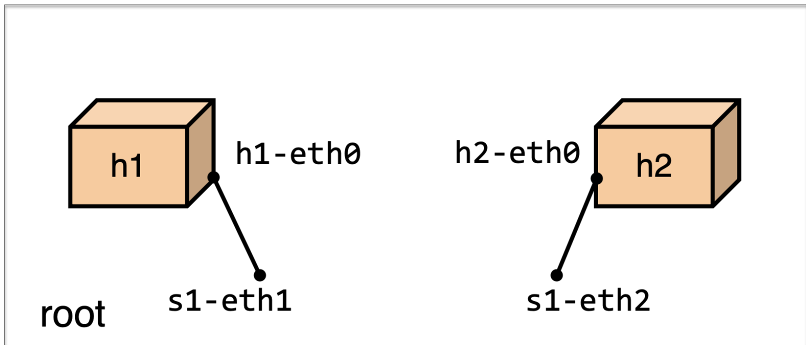
Move Ports into Host Namespaces

```shell
ip link set h1-eth0 netns h1   # h1-eth0 从 root 名字空间移动到 h1 名字空间下
ip link set h2-eth0 netns h2  # h2-eth0 从 root 名字空间移动到 h2 名字空间下

ip netns exec h1 ip link show # 在h1名字空间下执行ip link show ⇒ 可见h1-eth0
ip netns exec h2 ip link show # 在h2名字空间下执行ip link show ⇒ 可见h2-eth0
```

Now, we put NIC h1-eth0 and h2-eth0 into namespace h1 and h2 independently

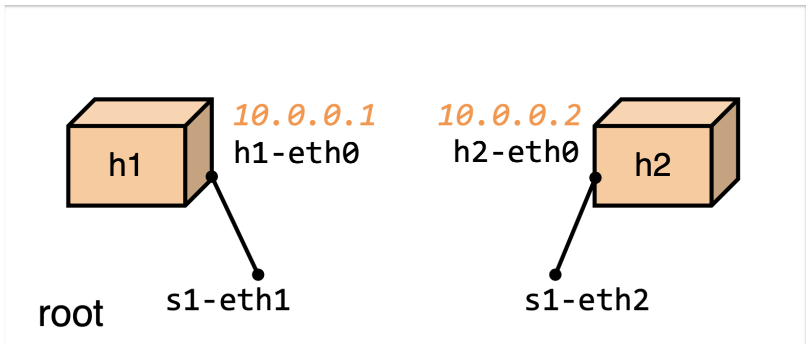
Hence, if we try to input "ip link show" in the CLI now:
We will not see the link h1-eth0→s1-eth1 and h2-eth0→s1-eth2



Configure Host Interfaces

```shell
ip netns exec h1 ifconfig h1-eth0 10.1 # offer IP to the NIC
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up # 在该网络命名空间中将 `lo` 接口（本地回环接口）启用
ip netns exec h1 ifconfig lo up
```

In fact, `ip netns exec h1 ifconfig lo up` and `ip netns exec h2 ifconfig lo up` can be ignored practically

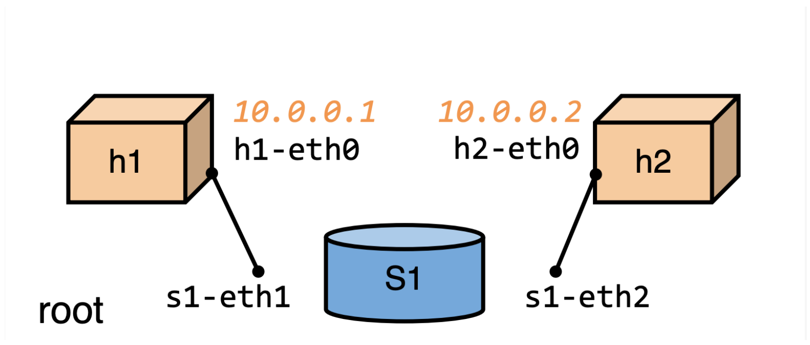


Create Virtual Switch

```shell
ovs-vsctl show
ovs-vsctl add-br s1 # 添加bridge(switch)，这个软件交换机called: S1
ovs-vsctl show
```

- **`ovs-vsctl`** 是 Open vSwitch 的管理工具之一，用于配置和管理 OVS。
- **`show`** 参数指示该命令显示当前 OVS 的配置信息，包括网桥、端口、控制器、流表等。

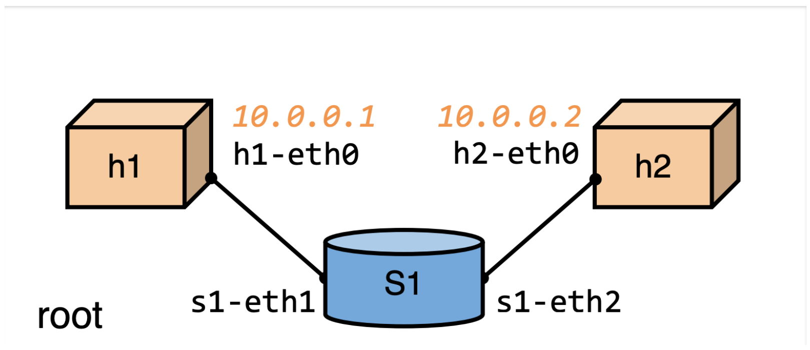
运行 **`ovs-vsctl show`** 命令将列出当前 OVS 实例中的各种配置信息



Create Switch Ports to OVS

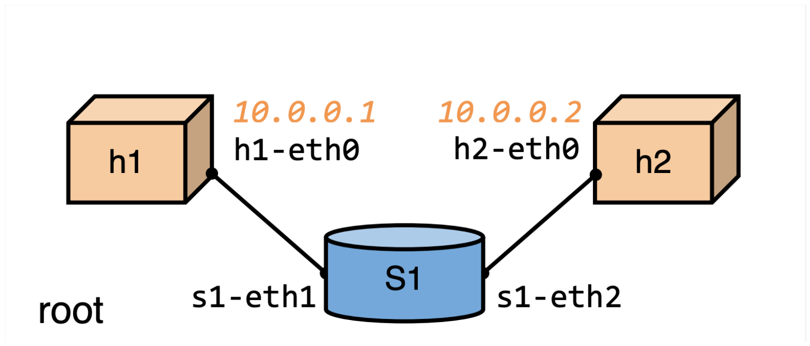
```shell
ovs-vsctl add-port s1 s1-eth1 # port→link in root namespace
ovs-vsctl add-port s1 s1-eth2
ovs-vsctl show
```

The ports of S1 Switch are connected with links



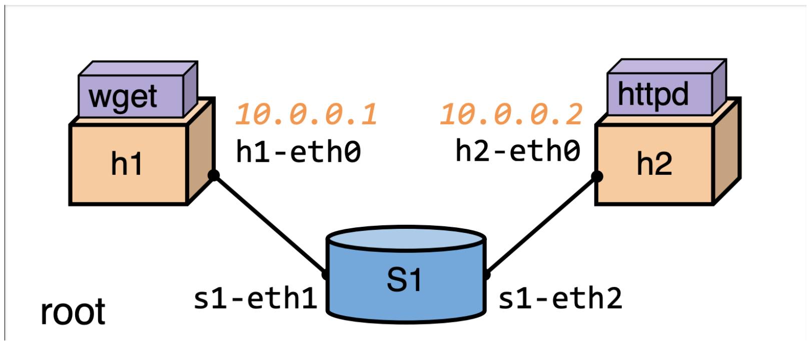
Test the Network

```
SHELL
ip netns exec h1 ping -c10 10.2 # 在h1名字空间中; -c10表示ping的次数 => 此时NIC仍处于关闭态
ifconfig s1-eth1 up                # s1-eth1的NIC开启
ifconfig s1-eth2 up                # s1-eth2的NIC开启
ip netns exec h1 ping -c10 10.2 # 此时可以ping通了
```



Test Client and Server

```
SHELL
ip netns exec h1 wget -O - 10.2
ip netns exec h2 python -m SimpleHTTPServer 80 &
```



Mininet

> Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM/cloud/native), in seconds, with a single command

Traits:

- A simple API that did this for us automatically
- Easily create topologies of varying size
- You must make orders in `su`

A summary for the listed orders above:

```
# Create host namespaces
ip netns add h1
ip netns add h2
# Create switch
ovs-vsctl add-br s1
# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h2 ifconfig h2-eth0 10.2
# Connect switch ports to OVS
ovs-vsctl add-port s1 s1-eth1
ovs-vsctl add-port s1 s1-eth2
```

They can be easily transmitted as below:

```shell
net = Mininet()
# Create hosts
h1 = net.addHost('h1')
h2 = net.addHost('h2')
# Create switch
s1 = net.addSwitch('s1')
# Create links
net.addLink(h1,s1)
net.addLink(h2,s1)
```

Show Nodes and Links

```shell
mininet> nodes
- Display nodes
```

```shell
mininet> net
- Display links
```

```shell
mininet> dump
- Dump ß(倾泻/详细显示) information about all nodes
```

Show Interface Configurations

```shell
mininet> h1 ifconfig -a
    - h1-eth0
    - lo

== 在主机 h1 上查看所有接口的配置信息
```

```shell
mininet> si ifconfig -a
    - eth0
    - lo

== 在交换机s1上查看所有接口的配置信息
```

Show the Process

```shell
mininet> h1 ps -a
    - show the process seen by h1...
mininet> h2 ps -a
    - show the process seen by h2...
```

Conclusion

- Only the network is virtualized
- Each host process sees the same set of processes and directories

Test Connectivity

```shell
mininet> h1 ping -c5 h2
   - Test connectivity between h1 and h2
   - The first ping takes a much longer time usually
mininet> pingall
   - Test all-pair connectivity
```

Simple Web Server and Client

```shell
mininet> xterm h1 h2
h1> python -m SimpleHTTPServer 80 &
h2> wget -O 10.0.0.1
```

Customize Topologies

详见Lab1 Tutorial