Internet Engineering 2019

Class2 ~Datalink Layer~

NOTICE(For non-macOS user)

We prepared a VM image for the assignment.

https://iplab.naist.jp/class/2019/materials/hands-on/01/vm.ova

To open this file, you need to install VirtualBox. For more details, please refer Guide.

After the launch the VM, **you can skip to page 15.**

Even if you're macOS user, you can use to save your time for preparing an environment.

NOTICE

Docker images has changed. If you have installed old images already, please follow the steps below.

1. Download the makefile.
   ```
   $ cd ~/Desktop/ieng
   $ curl -O https://iplab.naist.jp/class/2019/materials/hands-on/01/makefile
   ```
   If you have already makefiles, you don't need to download.
2. Execute `make` command on the same directory as the downloaded makefile.
   ```
   $ make clean -i
   ```
3. Remove old image
   ```
   $ docker rmi ieng
   ```
4. Download a docker image file.
   ```
   $ curl -O https://iplab.naist.jp/class/2019/materials/hands-on/01/ieng_fixed.tar
   ```
5. Install it.
   ```
   $ docker load < ieng_fixed.tar
   ```
6. Go to page. 10

Environment Setup (for macOS user)

- Docker
- XQuarts

Install Docker (for macOS user)

1. Download the docker.
    - from this URL: https://docs.docker.com/docker-for-mac/release-notes/
    - Unless there is some particular reason, you should use the latest version.
2. Follow the wizard.

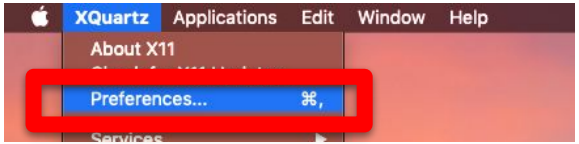
Install XQuartz (for macOS user)

XQuartz is X server for MacOS. It’s necessary for running Wireshark as GUI.

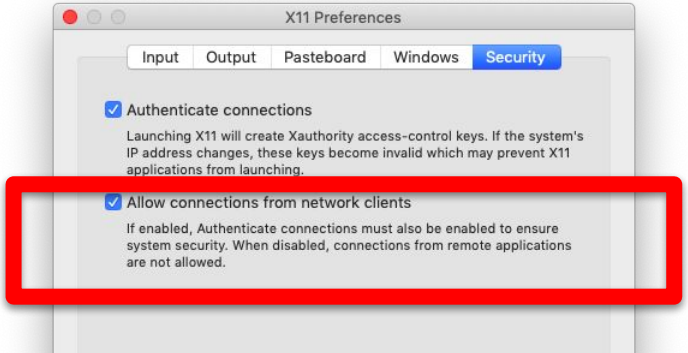
1. Download XQuartz.
    - https://www.xquartz.org/
    - Unless there is some particular reason, you should use the latest version.
2. Follow the wizard.
    - You need to restart PC.

Config XQuartz (for macOS user)

1. Run XQuartz
2. Open Preferences



3. Enable “Allow connections from network clients”



Config XQuartz (for macOS user)

4. Quit XQuartz

5. Reboot your PC

6. Execute XQueartz

Install a docker image (for macOS user)

If you installed fixed image file at 2 page, you don't need to do this step.

1. Create a working directory

   ```
   $ mkdir ~/Desktop/ieng && cd $_
   ```

2. Download a docker image.

   ```
   $ curl -O https://iplab.naist.jp/class/2019/materials/hands-on/01/ieng.tar
   PLEASE USE this one
   $ curl -O https://iplab.naist.jp/class/2019/materials/hands-on/01/ieng_fixed.tar
   ```

3. Install it.

   ```
   $ docker load < ieng_fixed.tar
   ```

Create an environment for assignment

1. Download the makefile.

   ```
   $ cd ~/Desktop/ieng
   ```

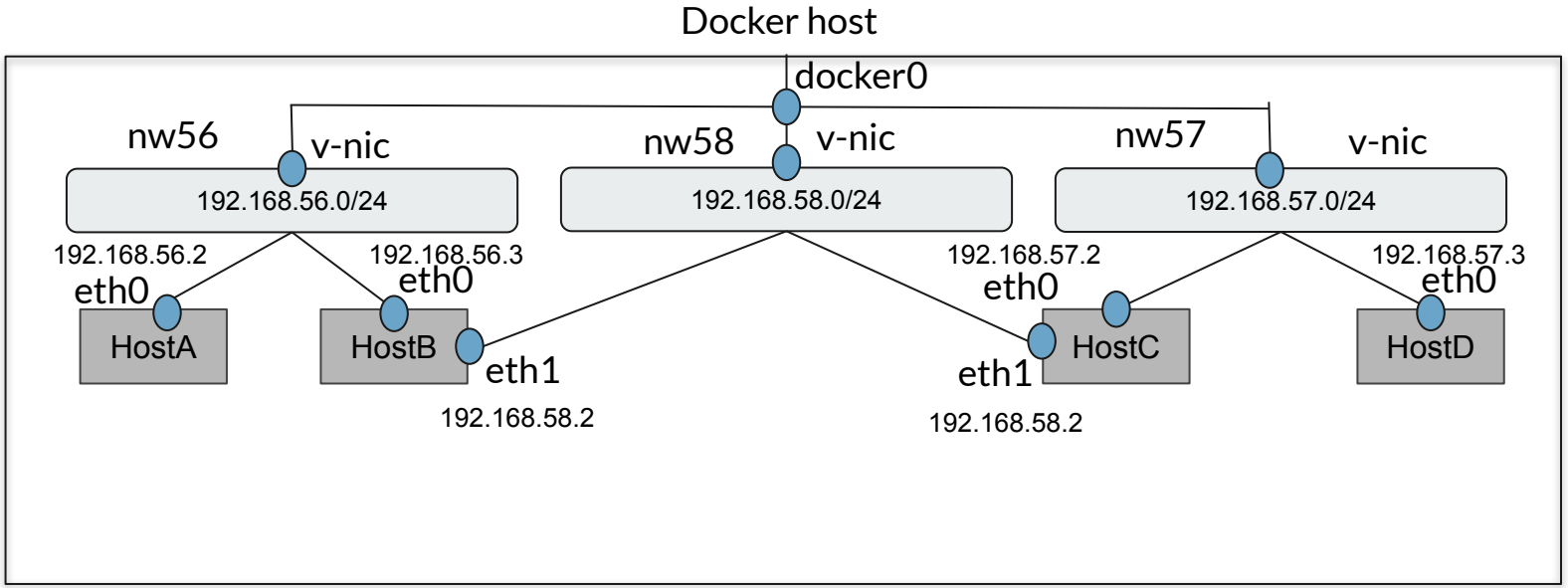
   ```
   $ curl -O https://iplab.naist.jp/class/2019/materials/hands-on/01/makefile
   ```

2. Execute `make` command on **the same directory** as the downloaded makefile.

   ```
   $ make
   ```

The network topology

After the executing make, the network topology will be configured as shown below.



Add 127.0.0.1 to xhost (for MacOS User)

We need to run wireshark by container. In this time, wireshark uses X server.
Therefore, you need to add your host to accepted list.

In the docker host, **not in the container**

```
$ xhost +127.0.0.1
```

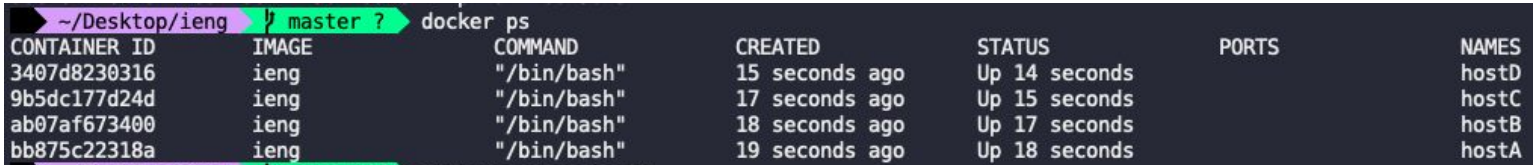
But it is said that this way is insecure.
If you afraid of it, it's better to use VM. (refer page. 2)

The way to reset is as follows:

```
$ xhost -127.0.0.1
```

Check whether all containers are started

From now, we assume that **ALL CONTAINERS ARE STARTED**.
You can check that started containers by `docker ps`.
If the output is like below, it's OK. Go to page 15.



* The value of container ID depends on the environment.

If not, you have to start containers that is not displayed. Go to page 14.

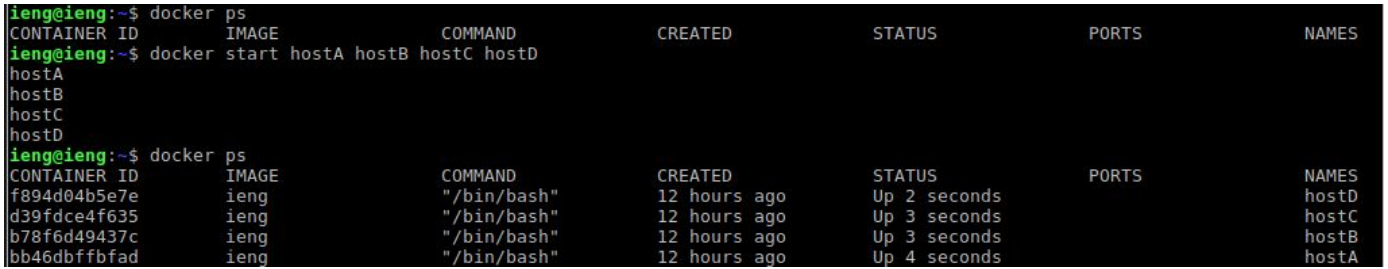
VM user will not displayed all containers. Therefore, you have to start all containers.

Start the container

If you can't find like below, you can start the containers by `docker start`.

```
$ docker start hostA hostB hostC hostD
```

For VM user, you need to follow the below's flow.



Attach the container(Cont.)

To enter inside of the container, attach option is required.

E.g. if you want to enter the hostA, the way is as follows:

```
$ docker attach hostA
```

The way to leave the container : `Ctrl-P` and `Ctrl-Q`

* When you use `exit` command or `Ctrl-D`, you need to restart the conteiners again with `start` option.

<Corpus>

If you get the error like “You cannot attach to a stopped container, start it first”, you are required to start the container. Check page14.

About wireshark

Wireshark is the world’s foremost and widely-used network protocol analyzer.

This is the filtering cheat sheet.

http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf

Run wireshark

Run wireshark like this in the container, then select the suitable interface.

```
$ wireshark
```

Apply the filter `not tcp.port 6000`. If you want to add more clause, please type `and` at the end.

This port(6000) is used by *X server* to show the GUI in your host computer.

* Actually We setted the alias as follows, it means the STDERR won't show and to execute in background.

```
alias wireshark='wireshark 2> /dev/null &'
```

Test the connection each other with pyng

pyng is reachability check tool using **Ethernet**. Before try this, you have to ENSURE all containers is started.

Server

```
$ sudo pyngd $INTERFACE
```

Client

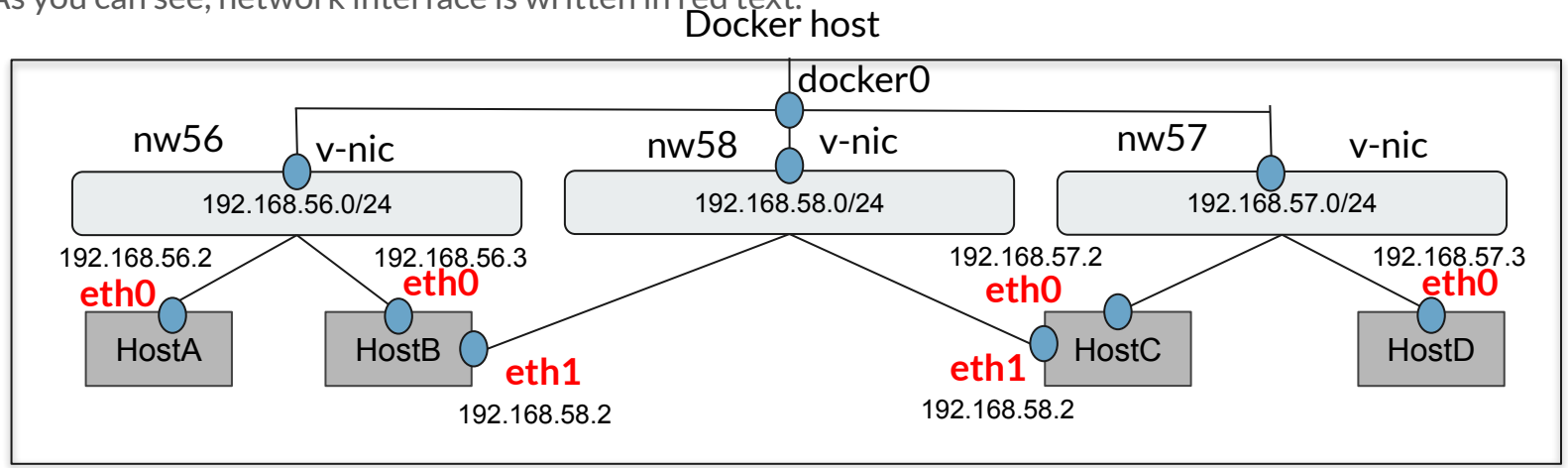
```
$ sudo pyng $DESTINATION_MAC_ADDRESS $SOURCE_INTERFACE
```

You need to change `$INTERFACE`, `$DESTINATION_MAC_ADDRESS`, `$SOURCE_INTERFACE`.
We'll explain about them.

Look p.19, we conduct a demo.

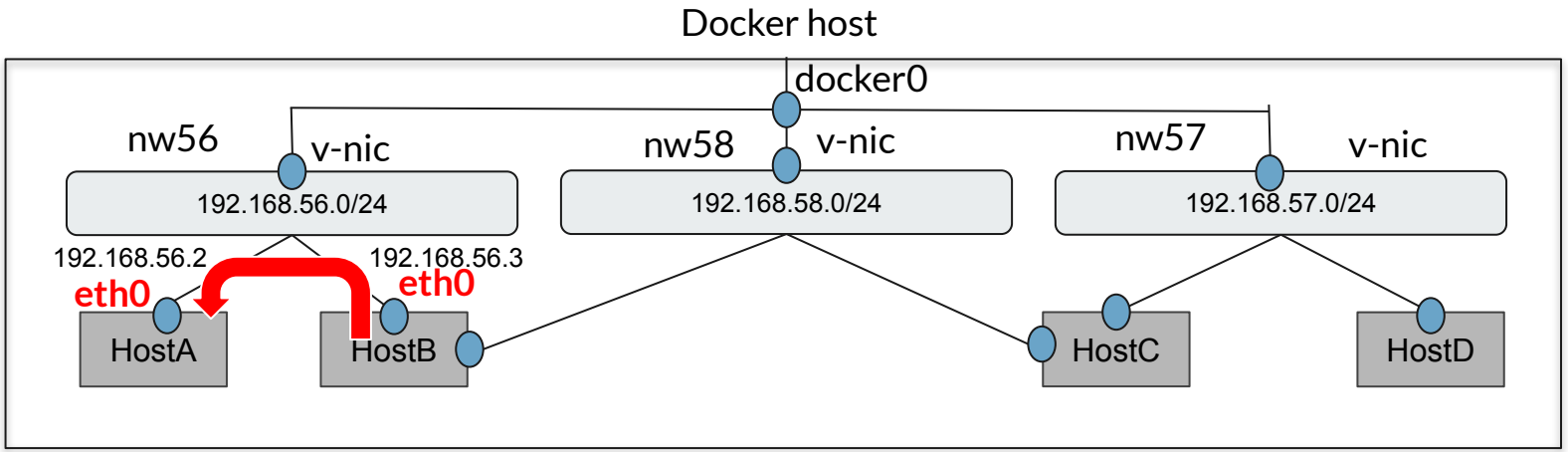
What's $INTERFACE?

$INTERFACE is network interface. To see it, use `ip link` command on some **container** (E.g. HostA, HostB…).
As you can see, network interface is written in red text.



Example test between hostA to hostB

We want to connect betwreen hostA(client) to hostB(server) in 192.168.56.0/24.
We can know that **hostA is eth0 and hostB is eth0** by following topology.



Example(Cont. ~ server ~)

We want to connect betwreen hostA(client) to hostB(server) in 192.168.56.0/24.
At first, let's check the interface of hostA, so specify hostA as server

```
$ docker attach hostA
```

Here is the hostA's MAC address as follows:

```
63: eth0@if64: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:38:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.56.2/24 brd 192.168.56.255 scope global eth0
       valid_lft forever preferred_lft forever
```

You don't need specify MAC Address when you run server(pyngd). But this is your address, so you need it when you send a packet to hostA.

```
$sudo pyngd eth0
```

Example (Cont. ~Client~)

Let's launch the another window of the terminal, after that attach hostB like bolow.

```
$ docker attach hostB # Make hostB a server, not hostA or host machine.
```

We already know the hostB's interface name(eth0), because we use nw56 network. We'll send to hostA. Therefore, you have to specify hostA's MAC address. That's why you have to remember before.

```
$ sudo pyng 02:42:c0:a8:38:02 eth0
```

```
ieng@ab07af673400:~$ sudo pyng 02:42:c0:a8:38:02 eth0
Sent!
```

Then, on the server side… . You can see receiving the frame from hostA. It says 'Hi' to server :)

```
ieng@bb875c22318a:~$ sudo pyngd eth0
dst: 02:42:c0:a8:38:02, src: 02:42:c0:a8:38:03, type: 0x88b5, payload: b'Hi'...
```

Test the connection each other with ping

`ping` uses the ICMP protocol's mandatory ECHO_ReQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway.

Ping is used to test the reachability a host on a Internet Protocol(IP)

$ ping <host> -c <the number of packets>

* If you want to stop ping, use `CTRL-C`

Cheat sheet

hostA	192.168.56.2
hostB	192.168.56.3
hostC	192.168.57.2
hostD	192.168.57.3

E.g.

```
$ docker ps   #Check the container
$ docker attach host1   #Attach the container
$ wireshark   #Capture the traffic in the container since this command
$ ping 192.168.56.2 -c 3   #Test the reachability to hostB from hostA
```

The way to check the information each nodes.

```
$ ip a
```