NASA Homework #3

Student Name: 林楷恩
Student ID: b07902075

Network Administration

1 Set up another Cisco Switch (11%)

1. **advantage**: useful when the network is unavailable
   disadvantage: requiring physical access, which can be troublesome for an administrator
2. He can use **Express Setup** to configure the switch. After the switch enter the Express Setup mode, he can connect it by a Ethernet cable. In the process, the switch will act as a DHCP server and give the connected computer a IP, so he can use HTTP graphical interface to configure it.
3. `No_TypE_7`
4. Pros of switch stacking:
   (a) *stacking* is more convenient. With combining multiple switches into single unit, the management becomes simpler because we can configure multiple switches with single interface.
   (b) *stacking* provides higher bandwidth compared with *trunk*.
5. Cei-ba need **two** stack cables to connect the two stacks in **ring topology** (e.g. a loop), which provides additional reliability. It means that if one stack link fails, they can still maintain the functionality by the rest link.

*** References**

- Online documents of Cisco
- http://www.fiberopticshare.com/switch-stacking-vs-trunking-whats-difference.html

2 Cisco Packet Tracer (14%)

-
  ```
  Switch(config)# hostname CiscoLab
  ```

-
  ```
  CiscoLab(config)# no ip domain-lookup
  ```

-
  ```
  CiscoLab(config)# enable password CISCO
  CiscoLab(config)# service password-encryption
  ```

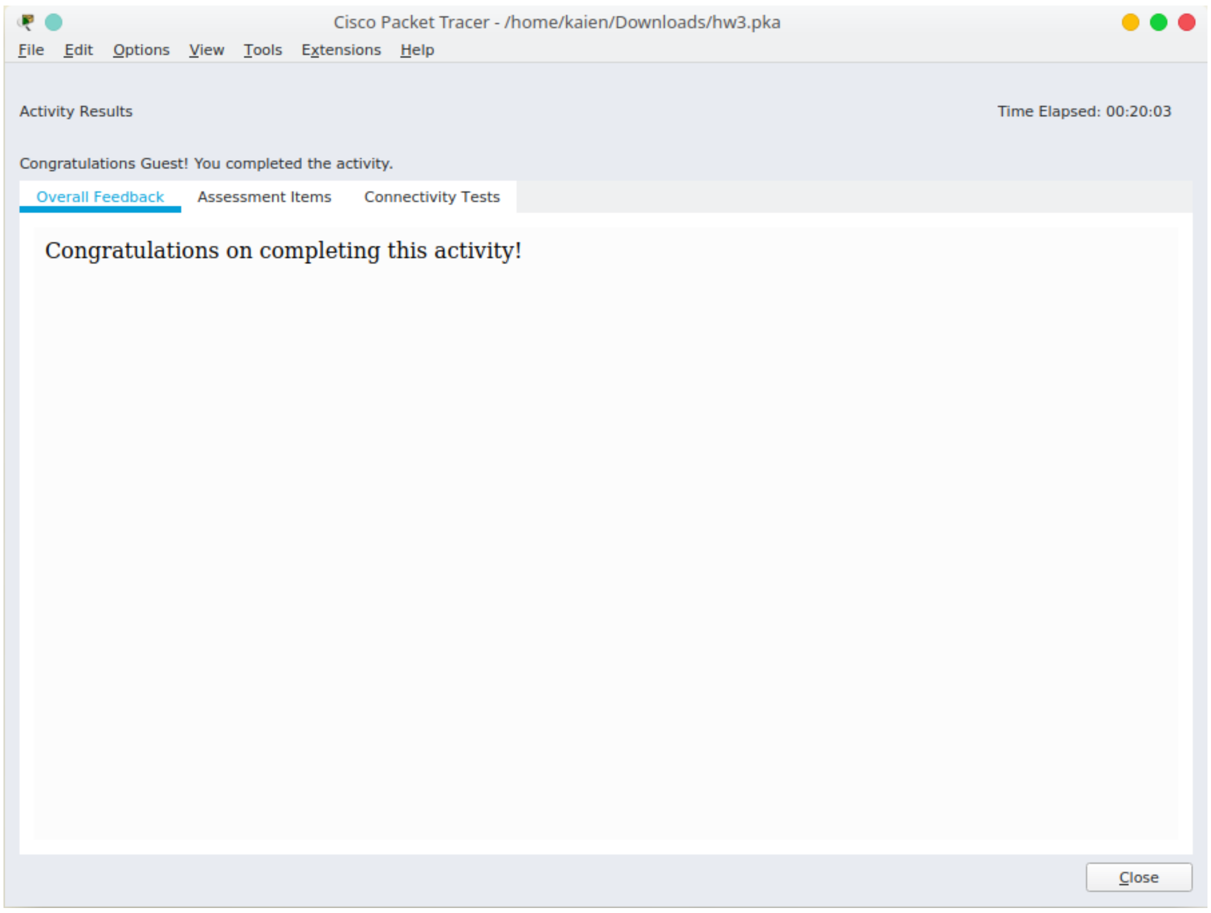
-
  ```
  CiscoLab(config)# vlan 10
  CiscoLab(config-vlan)# exit
  CiscoLab(config)# vlan 20
  CiscoLab(config-vlan)# exit
  CiscoLab(config)# vlan 99
  ```

-
  ```
  CiscoLab(config)# interface Fa0/1
  CiscoLab(config-if)# switchport access vlan 10
  CiscoLab(config-if)# exit
  CiscoLab(config)# interface Fa0/2
  CiscoLab(config-if)# switchport access vlan 10
  CiscoLab(config-if)# exit
  CiscoLab(config)# interface Fa0/3
  CiscoLab(config-if)# switchport access vlan 20
  CiscoLab(config-if)# exit
  CiscoLab(config)# interface Fa0/4
  ```

```
CiscoLab(config-if)# switchport access vlan 20
CiscoLab(config-if)# exit
```

-
  ```
  CiscoLab(config)# interface Fa0/5
  CiscoLab(config-if)# switchport access vlan 99
  CiscoLab(config-if)# exit
  CiscoLab(config)# interface vlan 99
  CiscoLab(config-if)# ip address 192.168.99.1 255.255.255.0
  CiscoLab(config-if)# no shutdown
  CiscoLab(config-if)# exit
  ```

-
  ```
  CiscoLab(config-if)# line vty 0 4
  CiscoLab(config-line)# password cisco
  CiscoLab(config-line)# login
  CiscoLab(config-line)# exit
  ```


Cisco Packet Tracer - /home/kaien/Downloads/hw3.pka
File Edit Options View Tools Extensions Help
Overall Feedback
Assessment Items
Connectivity Tests
Close

Activity Results

Time Elapsed: 00:20:03

Congratulations Guest! You completed the activity.

Congratulations on completing this activity!

3 Malicious User (6%)

We can first take advantage of the ip-arp table of the L3 core switch, by executing:

```
    CoreSwitch# show ip arp
```

we can find the MAC address and port of 140.112.30.250 in the output. Now the target is to find where is the MAC address. To trace down it, we type: (we may need to temporarily change to l2 mode to take use of the following commands on a L3 switch)

```
    CoreSwitch# show cdp neighbors detail
```

This command help us find the ip of the next switch on the path to the target. Then, we can telnet into that switch and type:

```
Switch1# show mac address-table
```

Find a MAC address equal to our target again, if the corresponded port is connected to an end device, then that it is. (if it is a cisco switch, it will appear in the output of `show cdp neighbors detail`, so we can use this to determine whether it is out target) Otherwise, we need to keep tracing down by telneting the next switch with ip address found by `# show cdp neighbors detail`.

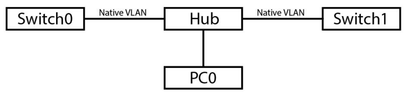
4 More on Link Aggregation (8%)

1. No, all the member links of a link aggregation group should operate at the same speed. So in this case, the Cat.6 cable will be restricted to the speed of Cat.5, and we get a 100Mbps * 2 bandwidth. This is because the controller handling the distribution of incoming traffic can only operate according to one configuration. Otherwise, it will be "confused" by two different types of links and thus lead to error.
2. The port-channel does not work because both of them are configured with **passive** mode. In this mode, a port only responds to LCAP data units which is used on negotiation. That is, it won't work as a port-channel member until it receives negotiation requests from other port. So what we need to do is change one of the ports's mode to active. For example:
   ```
   Switch(config)# interface Gi1/0/1
   Switch(config-if)# channel-group 1 mode active
   ```

5 The Evil VLAN, Access, and Trunk (11%)

1. The packet sent out via port Gi1/0/1 is tagged VLAN 424, while the one from port Gi1/0/2 has no 802.1q header. The difference is because Gi1/0/1 is configured on **trunk** mode, so the receiver, which may be a switch, still needs to know which VLAN it belongs to. As for Gi1/0/2, it is on **access** mode, since the switch has handled the separation, the header can be removed before being sent out.
2.
   - For the packet sent to port `Gi1/0/3`, since it operates on access mode, the packet will be forwarded to all the trunk port that allowed VLAN 307, which are Gi1/0/4 and Gi1/0/5. For Gi1/0/4, the 307 VLAN tag will remain there, while for Gi1/0/5, the VLAN tag will be removed due to the native VLAN configuration.
   - For the packet sent to port `Gi1/0/4`, it will first pass through the switch connected by port `Gi1/0/4`. After that, it can be tagged with VLAN 307 or VLAN 511, depends on which VLAN the original end user is in.
   - For the packet sent to port `Gi1/0/5`, it will first pass through the switch connected by port `Gi1/0/5`. If the original end user is in VLAN 307, the packet will not be tagged, since VLAN 307 is a **native VLAN** in this trunk link. then, the switch will determine where to forward this packet. If the destination MAC address matches the one on `Gi1/0/3`, it will be sent to `Gi1/0/3`, or it will be tagged with VLAN 307 and sent to `Gi1/0/4`.

     If the original end user is in VLAN 511, the packet will be tagged with VLAN 511, then sent to port `Gi1/0/5`. After receiving it, the switch will forward it to `Gi1/0/4` with the same VLAN 511 tag.
3. The native VLAN can provide *backward compatibility*. For example, there is a hub between switch, with a PC doesn't support 802.1q:

   In this scenario, we must configure the two switch with a native VLAN, since the hub cannot do tagging and just forwards all the traffic to all the links. If there is 802.1q tagged traffic, *PC0* will receive tagged traffic, which the pc has no capability of dealing with.

*** References**

- Discuss with *b07902064*

System Administration

1 Install a VM host running CentOS 7 (10%)

...After installation

- Install required packages `$ sudo yum install -y virt-install qemu-kvm libvirt`
- Start `libvirtd` service: `$ sudo systemctl start libvirtd`
- Make it automatically start on boot: `$ sudo systemctl enable libvirtd`

2 Create a Virtual Machine (guest) on VM host (18%)

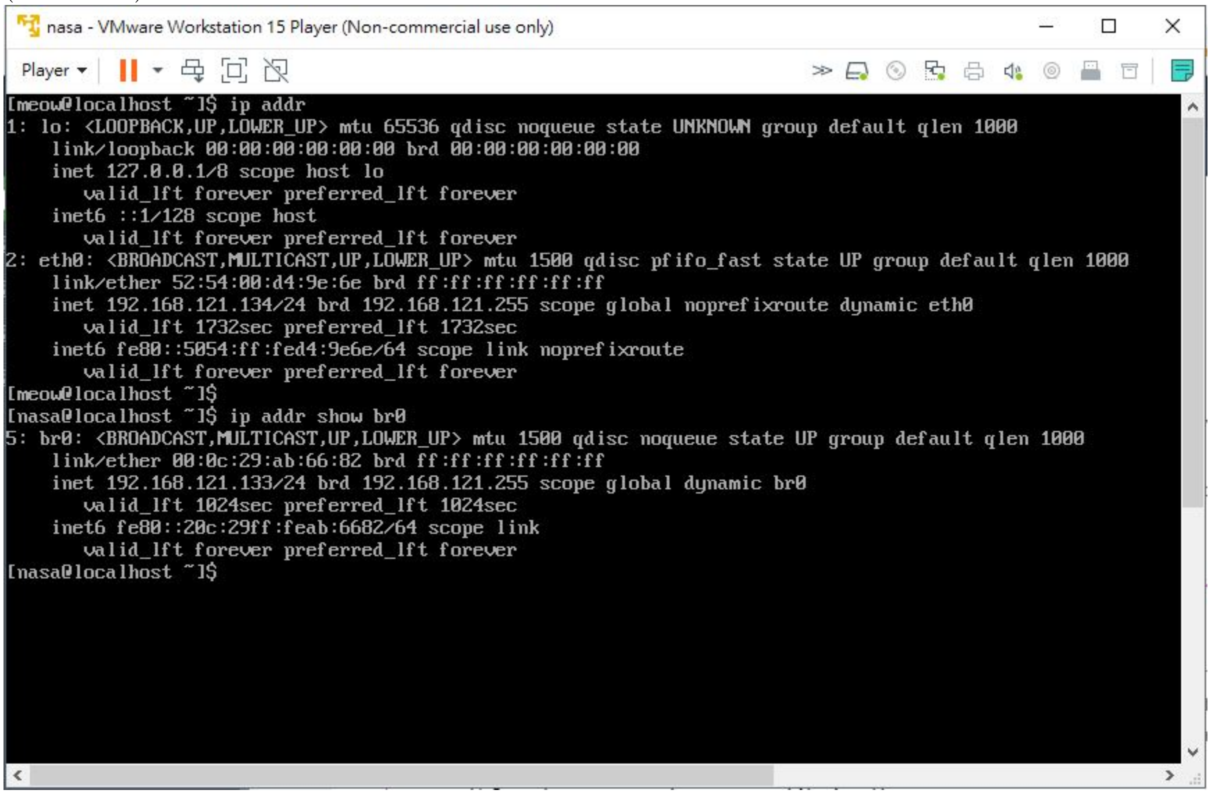
- Create a *QCOW2* image: `$ sudo qemu-img create -f qcow2 /data/img/disk1.qcow2 10G`
- Complete the kickstart script, first download the provided template: `$ curl ix.io/1ElA > ks.cfg`
  1. create user: (the encrypted password is generated by provided python script)
     `user --name meow --groups wheel --password [ENCRYPTED_PASSWORD] --iscrypted`
  2. install packages:
     ```
     %packages
     epel-release
     vim
     openssh-server
     sudo
     wget
     %end
     ```
  3. add a repo for `epel-release`, since the provided nctu repo doesn't have it:
     `repo --name=epel --baseurl=http://download.fedoraproject.org/pub/epel/6/x86_64/`
- set up bridge network interface:
  1. Use libvirt built-in utility: `$ sudo virsh iface-bridge ens33 br0`
  2. However, it is conflicted with *NetworkManager*, so disable it:
     ```
     $ sudo chkconfig NetworkManager off
     $ sudo chkconfig network on
     $ sudo service NetworkManager stop
     $ sudo service network start
     ```
- Use *virt-install* to create VM, note that anaconda kickstart installation requires memory ≥ 2GB, and the `--initrd-inject` argument is for inject the kickstart configuration file:

```
$ sudo virt-install \
        --connect qemu:///system \
        --name Spiorad \
        --memory 2048 \
        --vcpus 1 \
        --cpu host \
        --disk /data/img/disk1.qcow2 \
        --graphics none \
        --location http://centos.cs.nctu.edu.tw/7/os/x86_64/ \
        --network bridge=br0 \
        --initrd-inject=/home/nasa/ks.cfg \
        --extra-args "ks=file:/ks.cfg console=ttyS0"
```

- **Encountered Issue**: When I try to `virt-install` first time, I receive an error message: *Could not access KVM kernel module: Permission denied.* The solution is `$ sudo chown root:kvm /dev/kvm`, then add `user="root" group="root"` to */etc/libvirt/qemu.conf*. Finally `$ sudo systemctl restart libvirtd`

3 Enter Guest (5%)

1. `$ sudo virsh console Spiorad`
2. (screenshot)


nasa - VMware Workstation 15 Player (Non-commercial use only)
Player

```
[meow@localhost ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 52:54:00:d4:9e:6e brd ff:ff:ff:ff:ff:ff
    inet 192.168.121.134/24 brd 192.168.121.255 scope global noprefixroute dynamic eth0
       valid_lft 1732sec preferred_lft 1732sec
    inet6 fe80::5054:ff:fed4:9e6e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[meow@localhost ~]$
[nasa@localhost ~]$ ip addr show br0
5: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:0c:29:ab:66:82 brd ff:ff:ff:ff:ff:ff
    inet 192.168.121.133/24 brd 192.168.121.255 scope global dynamic br0
       valid_lft 1024sec preferred_lft 1024sec
    inet6 fe80::20c:29ff:feab:6682/64 scope link
       valid_lft forever preferred_lft forever
[nasa@localhost ~]$
```

4 Manage the VM from VM host (5%)

1. `# virsh list --all`
2. `# virsh undefine DOMAIN_NAME`
3. `# virsh edit DOMAIN_NAME`
4. `# virsh domiflist DOMAIN_NAME`
5. `# virsh iface-dumpxml br0`

5 Back up without stopping guest VM (12%)

- To get the utility of live backup, I need to update the *qemu-kvm* library.
  1. `# sudo yum install -y centos-release-qemu-ev`
  2. Edit /etc/yum.repos.d/CentOS-QEMU-EV.repo, change `enabled=0` to `enabled=1`
  3. `$ sudo yum install -y qemu-kvm-ev`
  4. Then restart the vm
- First check the state of the disk: `$ sudo virsh domblklist Spiorad`

-
  ```
  $ sudo virsh snapshot-create-as --domain Spiorad \
          --name snap1 --diskspec vda,snapshot=external --atomic --disk-only
  ```
- Now there are a base image `disk1.qcow2` and an overlay image `disk1.snap1`
- Copy the base image: `$ sudo cp /data/img/disk1.qcow2 /bc-img/nasa_backup.qcow2`

- Merge the overlay image back:
```
$ sudo virsh blockcommit Spiorad vda \
        --active --pivot --verbose
```

*** References**

- redhat online documantation 11.3. BRIDGED NETWORKING WITH LIBVIRT
- https://wiki.libvirt.org/page/Live-disk-backup-with-active-blockcommit
- https://pykickstart.readthedocs.io/en/latest/kickstart-docs.html
- Discuss with *b07902064*