0716306 project1 report

Question 1 : Configuration command

以下的 command 都是在 VM 的 terminal 下達

a. BRG1

start docker BRG1

・docker start BRG1

create a veth pair between interface BRG1-eth1 and br0-BRG1(BRG1-eth1 is the interface to br0, br0-BRG1 is the interface to br0)

・sudo ip link add BRG1-eth1 type veth peer name br0-BRG1

put the interface BRG1-eth1 into BRG1 container(br0 is created in VM)

・sudo ip link set BRG1-eth1 netns $(sudo dockeer inspect -f '{{.State.Pid}}') BRG1

bridge the interface br0 with br0-BRG1

・sudo brctl addif br0 br0-BRG1

set the interface up

・sudo ip link set br0-BRG1 up

enable ipv4 forwarding of BRG1

・sudo docker exec -it BRG1 sysctl -w net.ipv4.ip_forward=1

set up BRG1-eth1

・sudo docker exec -it BRG1 ip link set BRG1-eth1 up

b. BRGr

enable ipv4 forwarding

・sudo docker exec -it BRGr sysctl -w net.ipv4.ip_forward=1

set up interfaces

・sudo docker exec -it BRGr ip link set BRGr-eth0 up

・sudo docker exec -it BRGr ip link set BRGr-eth1 up

assign static ip to BRGr-eth0

・sudo docker exec -it BRGr ifconfig BRGr-eth0 140.113.0.2

copy GRE tunnel creation program to BRGr(executable file)

・docker cp gre_creation BRGr:/gre_creation

execute the GRE tunnel creation file

・./gre_creation

add route to ISP

・sudo docker exec -it BRGr route add -net 172.27.0.0/16 gw 140.114.0.1

c. Edge router

enable ipv4 forwarding

・.sudo docker exec -it edge_router sysctl -w net.ipv4.ip_forward=1

allocate ip for interface and set up

・sudo docker exec -it edge_router ifconfig edge-eth1 140.114.0.1

・sudo docker exec -it edge_router ifconfig edge-eth0 172.27.0.1

・sudo docker exec -it edge_router ip link set edge-eth0 up

・sudo docker exec -it edge_router ip link set edge-eth1 up

create a dhcpd.conf named dhcpd_edge.conf

・sudo docker exec -it edge_router echo -e “subnet172.27.0.0 netmask 255.255.255.0 {\n range 172.27.0.10 172.27.0.250;\n option routers 172.27.0.1;\n}” > dhcpd_edge.conf

run DHCP server and allocate IP to interface of BRG1

・sudo docker exec -it edge_router /usr/sbin/dhcpd 4 -cf ./dhcpd_edge.conf edge-eth0

change the iptables FORWARD chain setting to MASQUERADE

・sudo docker exec -it edge_router iptables -t nat -A POSTROUTING -s 172.27.0.0/24 -o edge-eth1 -j MASQUERADE

```
root@27f28b593be4:/# iptables -t nat -nvL
Chain PREROUTING (policy ACCEPT 4 packets, 824 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 4 packets, 824 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 4 packets, 245 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 4 packets, 245 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      edge-eth1  172.27.0.0/24        0.0.0.0/0
```

addd route to ISP

・sudo docker exec -it edge_router route add -net 140.113.0.0/16 gw 140.114.0.1

d. GWr

enable ipv4 forwarding

・sudo sysctl -w net.ipv4.ip_forward=1

create a dhcpd.conf named dhcpd_GWr.conf

・sudo echo -e “subnet 20.0.0.0 netmask 255.0.0.0 {\n range 20.0.0.10 20.0.0.250;\n option routers 10.0.2.15;\n option domain-name-servers 8.8.8.8;\n}” > dhcpd_GWr.conf

run DHCP server

・sudo /usr/sbin/dhcpd 4 -cf ./dhcpd_GWr.conf GWr-eth0

set NAT

・sudo iptables -t nat -A POSTROUTING -s 20.0.0.0/8 -o enp0s3 -j MASQUERADE

```
Chain PREROUTING (policy ACCEPT 25 packets, 12428 bytes)
 pkts bytes target     prot opt in     out     source               destination
   22 11688 DOCKER     all  --  *      *       0.0.0.0/0            0.0.0.0/0           ADDRTYPE match dst-type LOCA

Chain INPUT (policy ACCEPT 20 packets, 10536 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 946 packets, 78231 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER     all  --  *      *       0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCA

Chain POSTROUTING (policy ACCEPT 949 packets, 78971 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0
    0     0 MASQUERADE  all  --  *      enp0s3  20.0.0.0/8           0.0.0.0/0

Chain DOCKER (2 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 RETURN     all  --  docker0 *       0.0.0.0/0            0.0.0.0/0
```

Question 2 : Interface information

a. BRG1

```
BRG1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 8a:51:d9:99:eb:87  txqueuelen 1000  (Ethernet)
        RX packets 194  bytes 66348 (66.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

BRG1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.27.0.5  netmask 255.255.255.0  broadcast 172.27.0.255
        ether 7e:97:d1:95:76:a4  txqueuelen 1000  (Ethernet)
        RX packets 64  bytes 6959 (6.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10  bytes 1452 (1.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

b. BRG2

```
BRG2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 32:af:bb:81:cb:74  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

c. BRGr

```
BRGr-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 140.113.0.2  netmask 255.255.0.0  broadcast 140.113.255.255
        ether 3a:9d:4d:de:dc:08  txqueuelen 1000  (Ethernet)
        RX packets 6  bytes 476 (476.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 476 (476.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

BRGr-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 20.0.0.10  netmask 255.0.0.0  broadcast 20.255.255.255
        ether ea:fe:40:a5:19:1d  txqueuelen 1000  (Ethernet)
        RX packets 85  bytes 9574 (9.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 22  bytes 2516 (2.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether aa:05:69:59:9a:2f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 108 (108.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Question 3

Question 4

在 h1 發送封包後，BRGr 會在 MAC table 中記錄 h1 從哪個 gretap 進入，因此在 GWr 要將封包發送出去時，Linux kernel 可以透過 GRE packet 最外層 (outer ethernet header)得知要轉送到 h1 或 h2 的封包應該由哪個 gretap 出去