

1. Show the configuration commands you made on each node with explanation

(a) BRG1

▪ GRE over UDP (statically)

sudo docker exec -it BRG1 bash :

Command	explain
ip fou add port 1000 ipproto 47	Create port 1000 for FOU.
ip link add GRE_h1 type gretap remote 140.113.0.2 ikey 1 okey 1 encap fou encap-sport 1000 encap-dport 3000 ip link set GRE_h1 up	Create a gretap named GRE_h1 with encap type fou.
brctl addbr br0 brctl addif br0 GRE_h1 brctl addif br0 BRG1_h1 ip link set br0 up	Create a bridge named br0 and configure interfaces it attached.

(b) BRGr

▪ GRE over UDP (dynamically)

Command	explain
sudo g++ filter.cpp -lpcap sudo docker cp ./a.out f277bf0816b3:/	Compile the c++ file and copy it to the BRGr container
sudo docker exec -it BRGr bash ./a.out	Execute the a.out for auto tunnel creation


(c) Edge Router (r1)

▪ DHCP for BRG1, BRG2

sudo docker exec -it r1 bash

command	explain
ifconfig r1_br0 172.27.0.1 netmask 255.255.255.192 up	Assign an ip 172.27.0.1 for the interface r1_br0 on which DHCP is applied.
vim /etc/default/isc-dhcp-server	modify lterfacesV4="r1_br0"
vim /etc/dhcp/dhcpd.conf <pre> subnet 172.27.0.0 netmask 255.255.255.192 {     authoritative;     default-lease-time 600;     max-lease-time 7200;     range 172.27.0.2 172.27.0.61;     option routers 172.27.0.1;     option subnet-mask 255.255.255.192; } </pre>	Configure dhcpd.conf
/usr/sbin/dhcpd 4 -pf /run/dhcp-server-dhcpd.pid -cf /etc/dhcp/dhcpd.conf r1_br0	Run DHCP on r1.

▪ NAT rules for BRG1 (show NAT tables to justify your answer)

screenshot	explain
 <pre> lrts@SDH-NPV:~/mininet/containers/edges\$ sudo docker exec r1 iptables -t nat -L -n -v Chain PREROUTING (policy ACCEPT 1 packets, 328 bytes) pkts bytes target      prot opt in     out     source         destination  Chain INPUT (policy ACCEPT 1 packets, 328 bytes) pkts bytes target      prot opt in     out     source         destination  Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes) pkts bytes target      prot opt in     out     source         destination  Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes) pkts bytes target      prot opt in     out     source         destination 0      0 MASQUERADE  all  --  *      *       172.27.0.0/26  0.0.0.0/0 </pre>	<p>The NAT rule is applied by command “sudo docker exec r1 iptables -t nat -A POSTROUTING -s 172.27.0.0/26 -j MASQUERADE”.</p>

(d) GWr

▪ DHCP for hosts

command	explain
sudo ifconfig GWr_BRGr 20.0.0.1 netmask 255.0.0.0 up	Assign an ip 20.0.0.1 for the interface GWr_BRGr on which DHCP is applied.
sudo vi /etc/default/isc-dhcp-server	modify InterfacesV4="GWr_BRGr"
sudo vi /etc/dhcp/dhcpd.conf <pre> subnet 20.0.0.0 netmask 255.0.0.0 {     authoritative;     default-lease-time 1209600;     max-lease-time 1814400;     range 20.0.0.2 20.127.255.255;     option routers 10.0.2.15;     option broadcast-address 20.0.0.255;     option subnet-mask 255.0.0.0;     option domain-name-servers 8.8.8.8; }  subnet 20.0.0.0 netmask 255.0.0.0 {     authoritative;     default-lease-time 1209600;     max-lease-time 1814400;     range 20.128.0.0 20.255.255.254;     option routers 10.0.2.15;     option broadcast-address 20.0.0.255;     option subnet-mask 255.0.0.0;     option domain-name-servers 8.8.8.8; } </pre>	Configure dhcpd.conf
sudo /usr/sbin/dhcpd 4 -pf /run/dhcp-server-dhcpd.pid -cf /etc/dhcp/dhcpd.conf GWr_BRGr	Run the DHCP on interface GWr_BRGr

▪ NAT rules for hosts (show NAT tables to justify your answer)

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

Chain POSTROUTING (policy ACCEPT 64 packets, 11899 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
8	662	MASQUERADE	all	--	*	*	20.0.0.0/8	0.0.0.0/0

## 2. Show interfaces list on node BRGr and BRG1, 2 (10%)

```
lris@SDN-NFV:~/mininet/containers/edge$ sudo docker exec BRGr ifconfig
BRGr_Gw: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 02:3a:f3:03:ff:2a txqueuelen 1000 (Ethernet)
    RX packets 140 bytes 30080 (30.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 67 bytes 19734 (19.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

BRGr_r2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 140.113.0.2 netmask 255.255.255.0 broadcast 140.113.0.255
    ether 56:2c:0f:c6:4d:55 txqueuelen 1000 (Ethernet)
    RX packets 200 bytes 56100 (56.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 155 bytes 35093 (35.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

GRE_h1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 5a:c9:77:17:72:6d txqueuelen 1000 (Ethernet)
    RX packets 9 bytes 1354 (1.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1712 (1.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

GRE_h2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether b2:7f:20:4c:c6:c0 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10 bytes 1824 (1.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 5a:c9:77:17:72:6d txqueuelen 1000 (Ethernet)
    RX packets 86 bytes 22721 (22.7 KB)
    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
```

```
lris@SDN-NFV:~/mininet/containers/edge$ sudo docker exec BRG1 ifconfig
BRG1_br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.27.0.11 netmask 255.255.255.192 broadcast 172.27.0.63
    ether c6:07:69:c3:fe:65 txqueuelen 1000 (Ethernet)
    RX packets 283 bytes 38855 (38.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 215 bytes 42812 (42.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

BRG1_h1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether f2:8e:56:e1:3d:d5 txqueuelen 1000 (Ethernet)
    RX packets 69 bytes 21874 (21.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 1653 (1.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

GRE_h1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 7e:82:46:67:6d:a1 txqueuelen 1000 (Ethernet)
    RX packets 12 bytes 1545 (1.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 60 bytes 20008 (20.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 7e:82:46:67:6d:a1 txqueuelen 1000 (Ethernet)
    RX packets 68 bytes 21169 (21.1 KB)
    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
```

```
lris@SDN-NFV:~/mininet/containers/edge$ sudo docker exec BRG2 ifconfig
BRG2_br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.27.0.12 netmask 255.255.255.192 broadcast 172.27.0.63
    ether 2e:63:fa:f6:94:ed txqueuelen 1000 (Ethernet)
    RX packets 237 bytes 32018 (32.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 121 bytes 17302 (17.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

BRG2_h2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 56:af:13:68:8c:78 txqueuelen 1000 (Ethernet)
    RX packets 6 bytes 1172 (1.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10 bytes 1364 (1.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

GRE_h2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 1a:15:e9:a4:14:3e txqueuelen 1000 (Ethernet)
    RX packets 8 bytes 1256 (1.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 1168 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1464
    ether 1a:15:e9:a4:14:3e txqueuelen 1000 (Ethernet)
    RX packets 7 bytes 1396 (1.3 KB)
    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
```

### 3. Capture packets and take screenshots on node (10%)

h1 ping 8.8.8.8 :

BRG1	input	06:29:40.546486 IP 20.255.213.40 > 8.8.8.8: ICMP echo request, id 178, seq 1, length 64
	output	06:29:40.546522 IP 172.27.0.11.1000 > 140.113.0.2.3000: UDP, length 106
		BRG1 add <b>gre</b> and <b>udp headers</b> to the packet. (→length increase) The outer ip header has BRG1's ip as src and BRGr's ip as dst. The udp port is from BRG1's port 1000 to BRGr's port 3000.
R1	input	06:29:40.546546 IP 172.27.0.11.1000 > 140.113.0.2.3000: UDP, length 106
	output	06:29:40.546567 IP 140.114.0.1.1000 > 140.113.0.2.3000: UDP, length 106
		R1 do <b>POSTROUTING MASQUERADE NAT</b> to turn ip into 140.114.0.1 and forward it to R2.
R2	input	06:29:40.546595 IP 140.114.0.1.1000 > 140.113.0.2.3000: UDP, length 106
	output	06:29:40.546617 IP 140.114.0.1.1000 > 140.113.0.2.3000: UDP, length 106
		R2 just forward the packet to BRGr.
BRGr	input	06:29:40.546621 IP 140.114.0.1.1000 > 140.113.0.2.3000: UDP, length 106
	output	06:29:40.547550 IP 20.255.213.40 > 8.8.8.8: ICMP echo request, id 178, seq 1, length 64
		BRGr <b>take off</b> the outer eth, ip, <b>gre</b> , <b>udp headers</b> . (→length decrease) The packet's src/dst ip is now same as the one just came from h1.
GWr	Input / output	14:29:40.547557 IP 20.255.213.40 > dns.google: ICMP echo request, id 178, seq 1, length 64
		The NAT configuration on GWr is POSTROUTING, so I only catch this packet on GWr on the way to 8.8.8.8.



### Reply from 8.8.8.8 to h1

GWr	Input / output	14:29:40.563793 IP dns.google > 20.255.213.40: ICMP echo reply, id 178, seq 1, length 64
		I don't know how to check if POSTROUTING NAT works QQ. I expected to catch packet with src ip 20.0.0.1, but I can only catch this packet on veth GWr on the way back to h1.
BRGr	input	06:29:40.563797 IP 8.8.8.8 > 20.255.213.40: ICMP echo reply, id 178, seq 1, length 64
	output	06:29:40.563834 IP 140.113.0.2.3000 > 140.114.0.1.1000: UDP, length 106
		BRGr <b>encapsulate</b> the packet <b>with gretap and udp headers</b> . (→length increase)
R2	input	06:29:40.563843 IP 140.113.0.2.3000 > 140.114.0.1.1000: UDP, length 106
	output	06:29:40.563877 IP 140.113.0.2.3000 > 140.114.0.1.1000: UDP, length 106
		R2 just forward the packet to r1.
R1	input	06:29:40.563884 IP 140.113.0.2.3000 > 140.114.0.1.1000: UDP, length 106
	output	06:29:40.563916 IP 140.113.0.2.3000 > 172.27.0.11.1000: UDP, length 106
		There's <b>NAT</b> on R1, so the dst ip is <b>translated into 172.27.0.11</b> .
BRG1	input	06:29:40.563932 IP 140.113.0.2.3000 > 172.27.0.11.1000: UDP, length 106
	output	06:29:40.564075 IP 8.8.8.8 > 20.255.213.40: ICMP echo reply, id 178, seq 1, length 64
		<b>BRG1 decapsulate the gretap headers</b> , so the packet is forward to h1 according to inner ip 20.255.213.40. (→length decrease)

4. Briefly describe how BRGr determines the GRE interface to tunnel the response packets back to BRG1. (10%)

**A :** BRGr will look up the MAC address recorded in bridge table and forward the packets to corresponding entries(GRE interfaces). #