# NSCap Lab Report

## Lab4

1. Show the ping results.

   - ```
     mininet> h1 ping GWr -c 1
     PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
     64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.136 ms

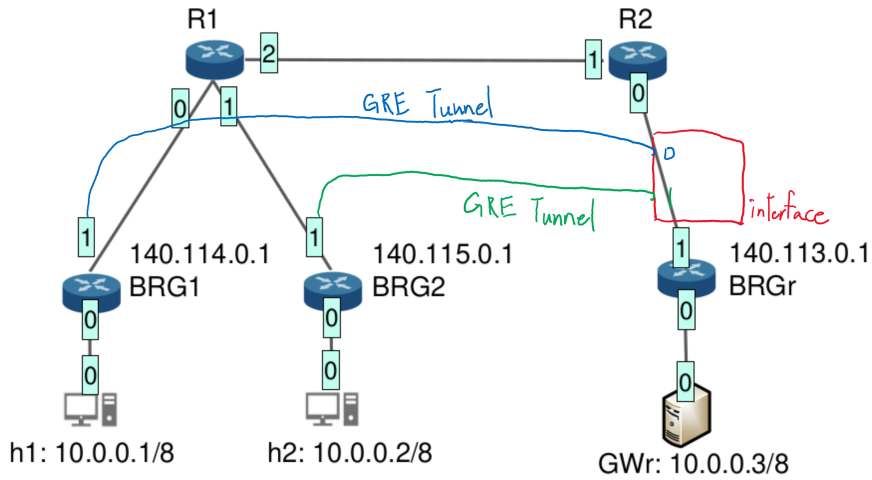
     --- 10.0.0.3 ping statistics ---
     1 packets transmitted, 1 received, 0% packet loss, time 0ms
     rtt min/avg/max/mdev = 0.136/0.136/0.136/0.000 ms
     ```
   - ```
     mininet> h2 ping GWr -c 1
     PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
     64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.131 ms

     --- 10.0.0.3 ping statistics ---
     1 packets transmitted, 1 received, 0% packet loss, time 0ms
     rtt min/avg/max/mdev = 0.131/0.131/0.131/0.000 ms
     ```
   - As the figure shown above, h1 and h2 can ping to GWr.

2. Show all interfaces of BRGr after h1 & h2 can ping GWr.

   - ```
     root@ubuntu:/home/tommytyc/NSCap/lab4# ./0616078
     0 Name: BRGr-eth0
     1 Name: BRGr-eth1
     2 Name: br0
     3 Name: 0
     4 Name: 1
     5 Name: any
     6 Name: lo
     7 Name: gre0
     8 Name: gretap0
     9 Name: erspan0
     10 Name: nflog
     11 Name: nfqueue
     Insert a number to select interface
     ```
   - As the figure shown above, there are two new interfaces, namely `interface 0` and `interface 1`.

3. Draw the interconnection diagramof interfaces and linux bridge on BRGr. Explain it with the screenshot of the interfaces.
     ```

- 

  - The red square is the interfaces of BRGr. We can see that `interface 0` (which is the option 3 in the interfaces list) has built a GRE Tunnel with BRG1 and `interface 1` (which is the option 4 in the interfaces list) has built an GRE Tunnel with BRG2.

4. Explain how Linux kernel of BRGr determines which gretap interface to forward packets from GWr to hosts (h1 or h2)?

- ```
root@ubuntu:/home/tommytyc/NSCap/lab4# tcpdump -i GWr-eth0 -eXX
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on GWr-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
22:25:44.489950 6e:76:d7:7a:d9:55 (oui Unknown) > Broadcast, ethertype ARP (0x08
06), length 42: Request who-has 10.0.0.1 tell 10.0.0.3, length 28
        0x0000:  ffff ffff ffff 6e76 d77a d955 0806 0001  ......nv.z.U....
        0x0010:  0800 0604 0001 6e76 d77a d955 0a00 0003  ......nv.z.U....
        0x0020:  0000 0000 0000 0a00 0001                 ..........
22:25:44.490124 66:9b:d5:06:ec:41 (oui Unknown) > 6e:76:d7:7a:d9:55 (oui Unknown
), ethertype ARP (0x0806), length 42: Reply 10.0.0.1 is-at 66:9b:d5:06:ec:41 (ou
i Unknown), length 28
        0x0000:  6e76 d77a d955 669b d506 ec41 0806 0001  nv.z.Uf....A....
        0x0010:  0800 0604 0002 669b d506 ec41 0a00 0001  ......f....A....
        0x0020:  6e76 d77a d955 0a00 0003                 nv.z.U....
```
  - As the figure shown above, GWr will collect the arp packet information to “learn” the MAC address of the other host and fill in the MAC table. After the MAC learning, GWr can send the packet to the correct gretap interface.

5. Run tcpdump on h1 to capture packet and take screenshot to explain why or why not h1 is aware of GRE tunneling.
```

- ```
  root@ubuntu:/home/tommytyc/NSCap/lab4# tcpdump -i h1-eth0 -eXX
  tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
  listening on h1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
  22:01:31.893595 6a:74:63:71:fd:97 (oui Unknown) > Broadcast, ethertype ARP (0x08
  06), length 42: Request who-has 10.0.0.3 tell 10.0.0.1, length 28
          0x0000:  ffff ffff ffff 6a74 6371 fd97 0806 0001  ......jtcq......
          0x0010:  0800 0604 0001 6a74 6371 fd97 0a00 0001  ......jtcq......
          0x0020:  0000 0000 0000 0a00 0003                 ..........
  22:01:31.893708 ee:3f:da:c0:32:82 (oui Unknown) > 6a:74:63:71:fd:97 (oui Unknown
  ), ethertype ARP (0x0806), length 42: Reply 10.0.0.3 is-at ee:3f:da:c0:32:82 (ou
  i Unknown), length 28
          0x0000:  6a74 6371 fd97 ee3f dac0 3282 0806 0001  jtcq...?..2.....
          0x0010:  0800 0604 0002 ee3f dac0 3282 0a00 0003  .......?..2.....
          0x0020:  6a74 6371 fd97 0a00 0001                 jtcq......
  22:01:31.893717 6a:74:63:71:fd:97 (oui Unknown) > ee:3f:da:c0:32:82 (oui Unknown
  ), ethertype IPv4 (0x0800), length 98: 10.0.0.1 > 10.0.0.3: ICMP echo request, i
  d 26315, seq 1, length 64
          0x0000:  ee3f dac0 3282 6a74 6371 fd97 0800 4500  .?..2.jtcq....E.
          0x0010:  0054 5c6f 4000 4001 ca36 0a00 0001 0a00  .T\o@.@..6......
          0x0020:  0003 0800 140b 66cb 0001 bb52 7460 0000  ......f....Rt`..
          0x0030:  0000 81a2 0d00 0000 0000 1011 1213 1415  ................
          0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ..........!"#$%
          0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
          0x0060:  3637                                     67
  22:01:31.893749 ee:3f:da:c0:32:82 (oui Unknown) > 6a:74:63:71:fd:97 (oui Unknown
  ), ethertype IPv4 (0x0800), length 98: 10.0.0.3 > 10.0.0.1: ICMP echo reply, id
  26315, seq 1, length 64
          0x0000:  6a74 6371 fd97 ee3f dac0 3282 0800 4500  jtcq...?..2...E.
          0x0010:  0054 b066 0000 4001 b63f 0a00 0003 0a00  .T.f..@..?......
          0x0020:  0001 0000 1c0b 66cb 0001 bb52 7460 0000  ......f....Rt`..
          0x0030:  0000 81a2 0d00 0000 0000 1011 1213 1415  ................
          0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ..........!"#$%
          0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
          0x0060:  3637                                     67
  ```
- h1 is not aware of GRE tunneling, because the tunnel is built on BRG1 and BRGr. Once the tunnel has successfully built, h1 and GWr can talk to each other through the tunnel. So, as the figure shown above, h1 just send out the arp request to get where is GWr(h1 thinks GWr is in the same LAN with itself), and then send out the ICMP packet.
  ```