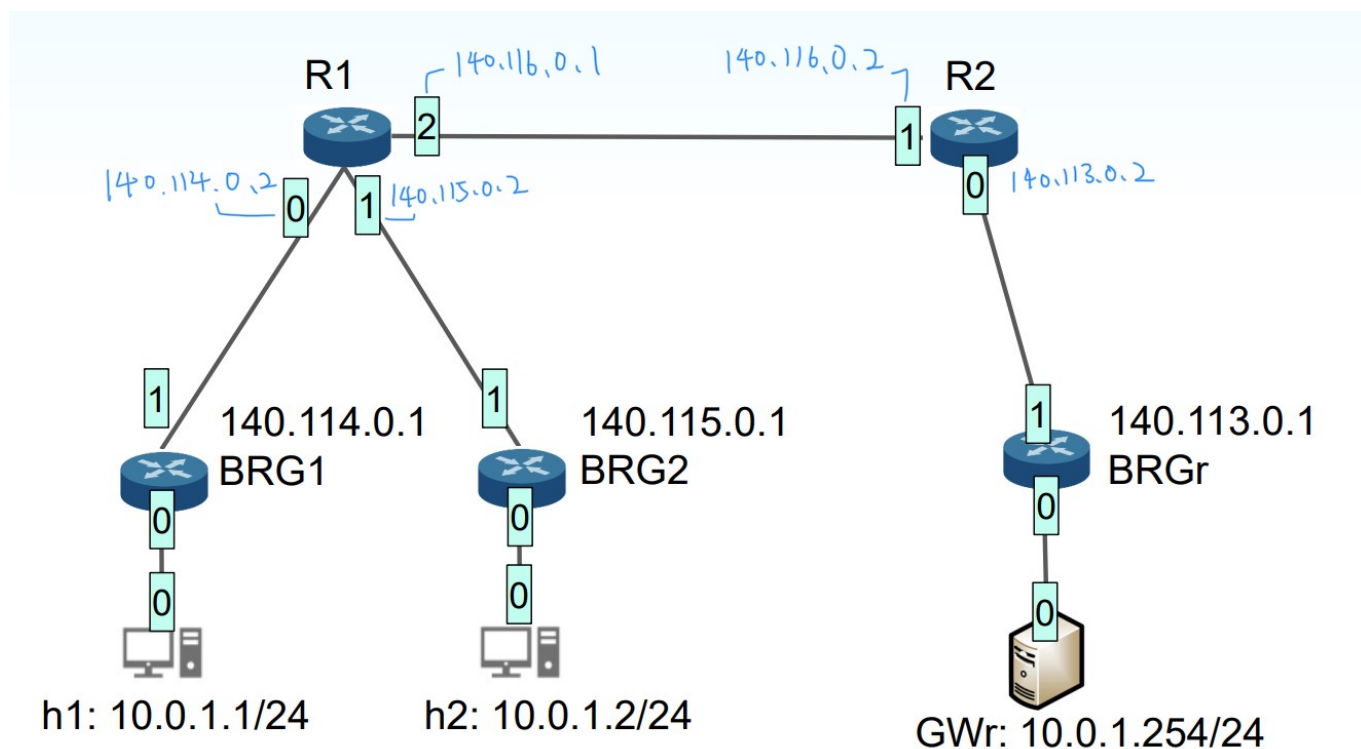


Lab 4: Dynamic Tunnel Creation

0816034 蔡家倫

Lab topology



Part 1

1. Show all interfaces of node BRGr and draw the interconnection diagram of interfaces and Linux bridge on BRGr. Explain your diagram with the interface list of BRGr. (10%)

- **BRGrGWrveth** connects **BRGr** and **GWr**.
- **br0** is the Linux bridge. It connects **BRGrGWrveth**, **GRETAP_1** and **GRETAP_2**.
- **GRETAP_1** is the GRE interface paired with **BRG1's GRETAP**.
- **GRETAP_2** is the GRE interface paired with **BRG2's GRETAP**.
- **BRGrR2veth** connects **BRGr** and **R2**.

Interfaces

```
soulr@ubuntu:~/Desktop/lab4$ sudo docker exec -it BRGr ifconfig
BRGrGWrvteth Link encap:Ethernet HWaddr 16:8d:bd:2a:f3:5a
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:108 (108.0 B)

BRGrR2veth Link encap:Ethernet HWaddr da:d6:b6:1e:8e:94
inet addr:140.113.0.1 Bcast:0.0.0.0 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:6 errors:0 dropped:0 overruns:0 frame:0
TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:564 (564.0 B) TX bytes:452 (452.0 B)

GRETAP_1 Link encap:Ethernet HWaddr 8a:39:08:82:cb:ed
UP BROADCAST RUNNING MULTICAST MTU:1462 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:80 (80.0 B)

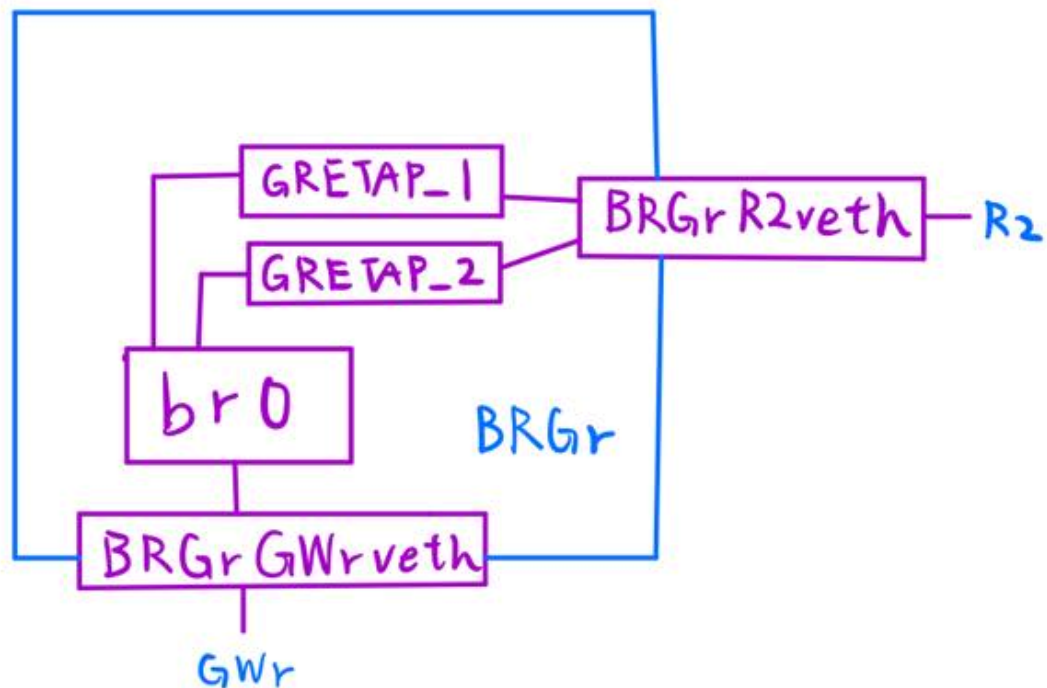
GRETAP_2 Link encap:Ethernet HWaddr f6:cc:55:05:6b:24
UP BROADCAST RUNNING MULTICAST MTU:1462 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:80 (80.0 B)
```

```
br0 Link encap:Ethernet HWaddr 16:8d:bd:2a:f3:5a
UP BROADCAST RUNNING MULTICAST MTU:1462 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:108 (108.0 B)

eth0 Link encap:Ethernet HWaddr 02:42:ac:11:00:07
inet addr:172.17.0.7 Bcast:172.17.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:392 errors:0 dropped:0 overruns:0 frame:0
TX packets:343 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:735076 (735.0 KB) TX bytes:19619 (19.6 KB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Interconnection diagram



2. Let h1 and h2 ping GWr and take screenshot of ping results. (5%)

```

soulr@ubuntu:~/Desktop/lab4$ sudo docker exec -it h1 ping 10.0.1.254 -c 3
[sudo] password for soulr:
PING 10.0.1.254 (10.0.1.254) 56(84) bytes of data.
64 bytes from 10.0.1.254: icmp_seq=1 ttl=64 time=0.616 ms
64 bytes from 10.0.1.254: icmp_seq=2 ttl=64 time=0.271 ms
64 bytes from 10.0.1.254: icmp_seq=3 ttl=64 time=0.768 ms

--- 10.0.1.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.271/0.551/0.768/0.209 ms
soulr@ubuntu:~/Desktop/lab4$ sudo docker exec -it h2 ping 10.0.1.254 -c 3
PING 10.0.1.254 (10.0.1.254) 56(84) bytes of data.
64 bytes from 10.0.1.254: icmp_seq=1 ttl=64 time=0.473 ms
64 bytes from 10.0.1.254: icmp_seq=2 ttl=64 time=0.695 ms
64 bytes from 10.0.1.254: icmp_seq=3 ttl=64 time=0.382 ms

--- 10.0.1.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
rtt min/avg/max/mdev = 0.382/0.516/0.695/0.134 ms
soulr@ubuntu:~/Desktop/lab4$
  
```

3. Can h1 ping h2? Take screenshots to explain why or why not (10%)

Yes. The packet will first send to BRG1, add GRE_TAP protocol (BRG1 to BRGr), send to BRGr, remove GRE_TAP protocol, add GRE_TAP protocol (BRGr to BRG2), send to BRG2, remove GRE_TAP protocol, then send to h2.

interfaces on BRGr

```

soulr@ubuntu:~/Desktop/Lab4$ <h proto GRE && proto ICMP <h proto GRE && p[5/5]
[ICMP] soulr@ubuntu:~/Desktop/Lab4$
soulr@ubuntu:~/Desktop/Lab4$ sudo docker exec -it BRGr tcpdump -i BRGrR2veth
[sudo] password for soulr:
Sorry, try again.
[sudo] password for soulr:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on BRGrR2veth, link-type EN10MB (Ethernet), capture size 262144 bytes
15:21:03.191423 IP 140.114.0.1 > 140.113.0.1: GREv0, length 46: ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:21:03.191522 IP 140.113.0.1 > 140.115.0.1: GREv0, length 46: ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:21:03.191686 IP 140.115.0.1 > 140.113.0.1: GREv0, length 46: ARP, Reply 10.0.1.2 is-at 8e:d8:bc:d3:3f:b5 (oui Unknown), length 28
15:21:03.191712 IP 140.113.0.1 > 140.114.0.1: GREv0, length 46: ARP, Reply 10.0.1.2 is-at 8e:d8:bc:d3:3f:b5 (oui Unknown), length 28
15:21:03.191803 IP 140.114.0.1 > 140.113.0.1: GREv0, length 102: IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 1, length 64
15:21:03.191829 IP 140.113.0.1 > 140.115.0.1: GREv0, length 102: IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 1, length 64
15:21:03.191963 IP 140.115.0.1 > 140.113.0.1: GREv0, length 102: IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 1, length 64
15:21:43.237370 IP 140.113.0.1.34717 > 192.168.170.2.domain: 21281+ PTR? 1.0.115.140.in-addr.arpa. (42)
soulr@ubuntu:~/Desktop/Lab4$ sudo docker exec -it BRGr tcpdump -i br0
[sudo] password for soulr:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on br0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:21:03.191423 ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:21:03.191686 ARP, Reply 10.0.1.2 is-at 8e:d8:bc:d3:3f:b5 (oui Unknown), length 28
15:21:03.191803 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 1, length 64
15:21:03.191963 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 1, length 64
15:21:04.193002 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 2, length 64
15:21:04.193163 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 2, length 64
15:21:05.218643 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 3, length 64
soulr@ubuntu:~/Desktop/Lab4$ sudo docker exec -it BRGr tcpdump -i GRETAP_1
[sudo] password for soulr:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on GRETAP_1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:21:03.191423 ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:21:03.191701 ARP, Reply 10.0.1.2 is-at 8e:d8:bc:d3:3f:b5 (oui Unknown), length 28
15:21:03.191803 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 1, length 64
15:21:03.191977 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 1, length 64
15:21:04.193002 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 2, length 64
15:21:04.193180 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 2, length 64
15:21:05.218643 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 3, length 64
soulr@ubuntu:~/Desktop/Lab4$ sudo docker exec -it BRGr tcpdump -i GRETAP_2
[sudo] password for soulr:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on GRETAP_2, link-type EN10MB (Ethernet), capture size 262144 bytes
15:21:03.191499 ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:21:03.191686 ARP, Reply 10.0.1.2 is-at 8e:d8:bc:d3:3f:b5 (oui Unknown), length 28
15:21:03.191818 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 1, length 64
15:21:03.191963 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 1, length 64
15:21:04.193030 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 2, length 64
15:21:04.193163 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 34, seq 2, length 64
15:21:05.218717 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 34, seq 3, length 64

```

h2BRG2veth on h2

```

soulr@ubuntu:~$ sudo docker exec -it h2 tcpdump -i h2BRG2veth
[sudo] password for soulr:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2BRG2veth, link-type EN10MB (Ethernet), capture size 262144 bytes
15:56:52.008343 ARP, Request who-has 10.0.1.2 tell 10.0.1.1, length 28
15:56:52.008372 ARP, Reply 10.0.1.2 is-at da:54:31:8a:d8:3c (oui Unknown), length 28
15:56:52.008601 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 33, seq 1, length 64
15:56:52.008653 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 33, seq 1, length 64
15:56:53.010005 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 33, seq 2, length 64
15:56:53.010130 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 33, seq 2, length 64
15:56:54.017950 IP 10.0.1.1 > 10.0.1.2: ICMP echo request, id 33, seq 3, length 64
15:56:54.018037 IP 10.0.1.2 > 10.0.1.1: ICMP echo reply, id 33, seq 3, length 64
15:56:57.057910 ARP, Request who-has 10.0.1.1 tell 10.0.1.2, length 28
15:56:57.058395 ARP, Reply 10.0.1.1 is-at 86:cb:57:81:8d:de (oui Unknown), length 28

```

4. Explain how Linux kernel of BRGr determines which gretap interface to forward packets from GWr to hosts (h1 or h2)? Describe your answer with appropriate screenshots. (5%)

br0 will decide send the packet to **GRETAP_1** or **GRETAP_2** according to its ARP table. (port 2 is connected to **GRETAP_1**, port 1 is connected to **GRETAP_2**)


```
soulr@ubuntu:~$ sudo docker exec -it BRGr brctl showmacs BRGrR2veth
read of forward table failed: Operation not supported
soulr@ubuntu:~$ sudo docker exec -it BRGr brctl showmacs br0
port no mac addr is local? ageing timer
 3 22:82:7b:69:40:fd yes 0.00
 3 22:82:7b:69:40:fd yes 0.00
 2 86:cb:57:81:8d:de no 10.18
 2 ae:94:1e:a2:e0:68 yes 0.00
 2 ae:94:1e:a2:e0:68 yes 0.00
 1 ca:48:e1:70:eb:e2 yes 0.00
 1 ca:48:e1:70:eb:e2 yes 0.00
 3 da:54:31:8a:d8:3c no 10.18
soulr@ubuntu:~$ sudo docker exec -it h1 ifconfig h1BRG1veth
h1BRG1veth Link encap:Ethernet HWaddr 86:cb:57:81:8d:de
inet addr:10.0.1.1 Bcast:0.0.0.0 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:20 errors:0 dropped:0 overruns:0 frame:0
TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1704 (1.7 KB) TX bytes:1596 (1.5 KB)
soulr@ubuntu:~$ sudo docker exec -it h2 ifconfig h2BRG2veth
h2BRG2veth Link encap:Ethernet HWaddr da:54:31:8a:d8:3c
inet addr:10.0.1.2 Bcast:0.0.0.0 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:20 errors:0 dropped:0 overruns:0 frame:0
TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1704 (1.7 KB) TX bytes:1596 (1.5 KB)
soulr@ubuntu:~$
```

5. Is h1 aware of GRE tunneling? Take screenshot to explain why or why not (5%)

No. The screenshot shows that h1 doesn't know the existence of GRE tunneling.

```
soulr@ubuntu:~$ sudo docker exec -it h1 tcpdump -l h1BRG1veth -exX
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h1BRG1veth, link-type EN10MB (Ethernet), capture size 262144 bytes
02:49:48.187817 86:cb:57:81:8d:de (out Unknown) > Broadcast, ethertype ARP (0x0806), length 42: Request who-has 10.0.1.254 tell 10.0.1.1, length 28
0x0000: 0001 0000 0004 0001 86cb 5781 8dde 0a00 .....W.....
0x0010: 0101 0000 0000 0000 0a00 01fe .....6..2....
02:49:48.187955 36:c4:8e:32:92:e3 (out Unknown) > 86:cb:57:81:8d:de (out Unknown), ethertype ARP (0x0806), length 42: Reply 10.0.1.254 is-at 36:c4:8e:32:92:e3 (out Unknown), length 28
0x0000: 0001 0000 0004 0002 36c4 8e32 92e3 0a00 .....W.....
0x0010: 01fe 86cb 5781 8dde 0a00 0101 .....6..2....
02:49:48.187967 86:cb:57:81:8d:de (out Unknown) > 36:c4:8e:32:92:e3 (out Unknown), ethertype IPv4 (0x0800), length 98: 10.0.1.1 > 10.0.1.254: ICMP echo request, id 91, seq 1, length 64
0x0000: 4500 0054 ec4c 4000 4001 375e 0a00 0101 E..T.L@.7^....
0x0010: 0a00 01fe 0800 93ee 005b 0001 cca2 4f62 .....[....Ob
0x0020: 0000 0000 86dd 0200 0000 0000 1011 1213 .....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637 .....4567
02:49:48.188009 36:c4:8e:32:92:e3 (out Unknown) > 86:cb:57:81:8d:de (out Unknown), ethertype IPv4 (0x0800), length 98: 10.0.1.254 > 10.0.1.1: ICMP echo reply, id 91, seq 1, length 64
0x0000: 4500 0054 9eae 0000 4001 c504 0a00 01fe E..T...@.....
0x0010: 0a00 0101 0000 9bee 005b 0001 cca2 4f62 .....[....Ob
0x0020: 0000 0000 86dd 0200 0000 0000 1011 1213 .....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 .....!"#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,-./0123
0x0050: 3435 3637 .....4567
```

Part 2

Run Tunnel Auto Creation Program on BRGr and show the parsed result of one captured packet. (5%)

```
Device 0: br0
Device 1: eth0
Device 2: BRGrGWrVeth
Device 3: BRGrR2veth
Device 4: any
Device 5: lo
Device 6: nflog
Device 7: nfqueue
Device 8: usbmon1
Device 9: usbmon2
Insert a number to select interface:
3
Interface: 3 BRGrR2veth
Start listening at BRGrR2veth
Insert BPF filter expression:
ip proto gre
filter: ip proto gre

Waiting packets...
-----
Packet 1 captured
Packet capture length: 80
Packet total length 80
Source MAC: 12:1d:b6:e6:60:db
Destination MAC: 3e:ef:7f:f5:9e:11
Ethernet type: IPv4
Source IP: 140.114.0.1
Destination IP: 140.113.0.1
Next layer protocol: GRE
Found GRETAP
Inner source MAC: 26:f9:70:e2:25:81
Inner destination MAC: ff:ff:ff:ff:ff:ff
GRE tunnel 0 builded.
Byte code:
  0  3e ef 7f f5 9e 11 12 1d b6 e6 60 db 08 00 45 00
 16  00 42 2d 37 40 00 3e 2f f6 70 8c 72 00 01 8c 71
 32  00 01 00 00 65 58 ff ff ff ff ff ff 26 f9 70 e2
 48  25 81 08 06 00 01 08 00 06 04 00 01 26 f9 70 e2
 64  25 81 0a 00 01 01 00 00 00 00 00 00 0a 00 01 02
```