Kacper Szaruch
Jan Wojciechowski

Politechnika Warszawska

Sprawozdanie z realizacji laboratorium KRI nr 6
L3VPN

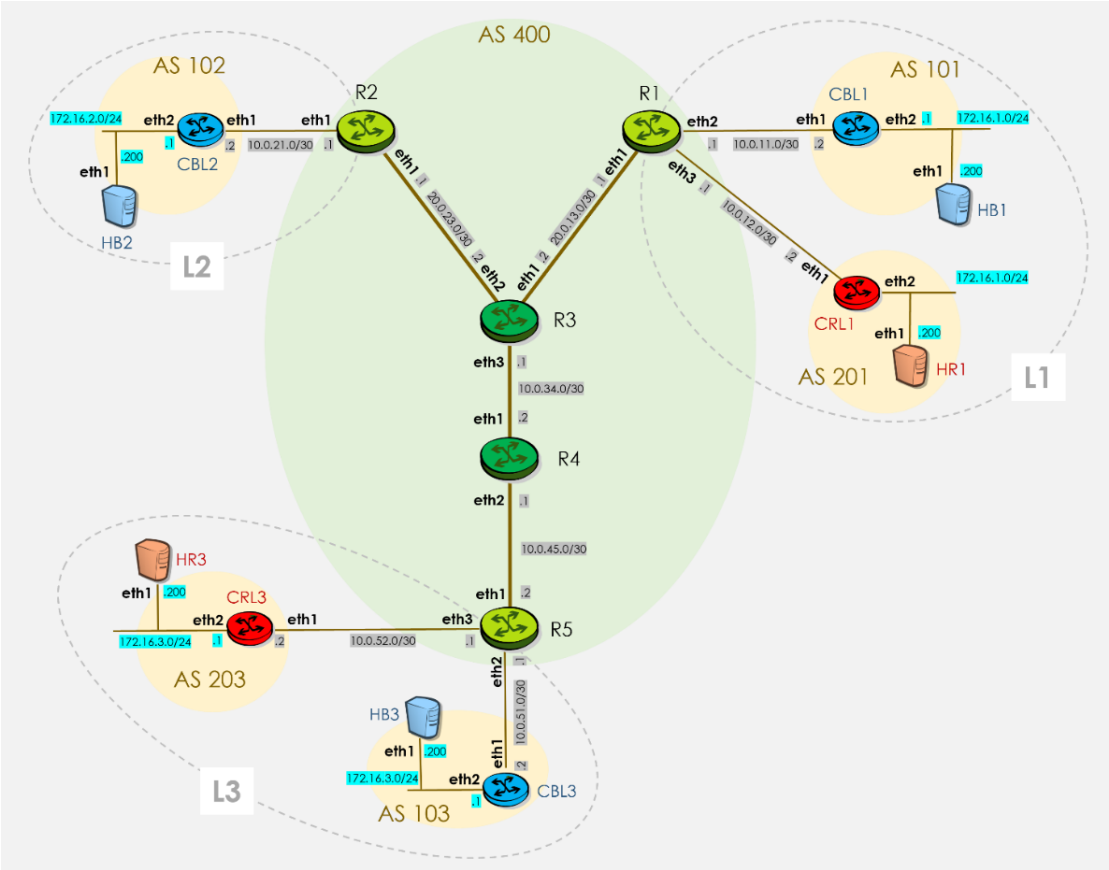
16 marca 2024

Spis treści

Wstęp

Niniejszy dokument to sprawozdanie z realizacji laboratorium w ramach przedmiotu KRI. Oświadczamy, że ta praca, stanowiąca podstawę do uznania osiągnięcia efektów uczenia się z przedmiotu KRI, została wykonana przez nas samodzielnie.

A: Przypisanie adresów IP



Rys. 1: Topologia sieci

Router	Address	Router	Address
R1	1.1.1.1/32	CBL1	11.11.11.11/32
R2	2.2.2.2/32	CRL1	21.21.21.21/32
R3	3.3.3.3/32	CBL2	12.12.12.12/32
R4	4.4.4.4/32	CRL3	23.23.23.23/32
R5	5.5.5.5/32	CBL3	13.13.13.13/32

Rys. 2: Adresacja interfejsów loopback

Powyżej przedstawiona została topologia sieci oraz adresacja interfejsów loopback wykorzystywanych w ramach tego ćwiczenia laboratoryjnego.

B: Konfiguracja OSPF w AS400

```
R1# show ip ospf database

       OSPF Router with ID (1.1.1.1)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1371 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1376 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1375 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1394 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1382 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1355 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1365 0x80000002 0xd638
20.0.34.2       4.4.4.4         1384 0x80000002 0x9364
20.0.45.2       5.5.5.5         1362 0x80000002 0x5090
```

Rys. 3: Wynik wykonania komendy *show ip ospf database* na **R1**

```
R1# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
 RXmtL RqstL DBsmL
3.3.3.3           1 Full/DR         50m24s            37.441s 20.0.13.2       eth1:20.0.13.1
     0     0     0
```

Rys. 4: Wynik wykonania komendy *show ip ospf neighbors* na **R1**

```
R1# show ip ospf database

       OSPF Router with ID (1.1.1.1)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1371 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1376 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1375 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1394 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1382 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1355 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1365 0x80000002 0xd638
20.0.34.2       4.4.4.4         1384 0x80000002 0x9364
20.0.45.2       5.5.5.5         1362 0x80000002 0x5090
```

Rys. 5: Wynik wykonania komendy *show ip ospf database* na **R1**

```
R1# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
 RXmtL RqstL DBsmL
3.3.3.3           1 Full/DR         50m24s            37.441s 20.0.13.2       eth1:20.0.13.1
     0     0     0
```

Rys. 6: Wynik wykonania komendy *show ip ospf neighbors* na **R1**

```
R2# show ip ospf database

       OSPF Router with ID (2.2.2.2)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1381 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1383 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1384 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1402 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1391 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1364 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1374 0x80000002 0xd638
20.0.34.2       4.4.4.4         1392 0x80000002 0x9364
20.0.45.2       5.5.5.5         1371 0x80000002 0x5090
```

Rys. 7: Wynik wykonania komendy *show ip ospf database* na **R2**

```
R2# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
 RXmtL RqstL DBsmL
3.3.3.3           1 Full/DR         50m49s            39.210s 20.0.23.2       eth1:20.0.23.1
     0     0     0
```

Rys. 8: Wynik wykonania komendy *show ip ospf neighbors* na **R2**

```
R3# show ip ospf database

       OSPF Router with ID (3.3.3.3)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1385 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1389 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1388 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1406 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1395 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1368 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1378 0x80000002 0xd638
20.0.34.2       4.4.4.4         1396 0x80000002 0x9364
20.0.45.2       5.5.5.5         1375 0x80000002 0x5090
```

Rys. 9: Wynik wykonania komendy *show ip ospf database* na **R3**

```
R3# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
 RXmtL RqstL DBsmL
4.4.4.4           1 Full/DR         50m58s            30.652s 20.0.34.2       eth3:20.0.34.1
     0     0     0
2.2.2.2           1 Full/Backup     50m58s            30.668s 20.0.23.1       eth2:20.0.23.2
     0     0     0
1.1.1.1           1 Full/Backup     50m50s            30.652s 20.0.13.1       eth1:20.0.13.2
     0     0     0
```

Rys. 10: Wynik wykonania komendy *show ip ospf neighbors* na **R3**

```
R4# show ip ospf database

       OSPF Router with ID (4.4.4.4)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1390 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1393 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1392 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1409 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1397 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1372 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1382 0x80000002 0xd638
20.0.34.2       4.4.4.4         1399 0x80000002 0x9364
20.0.45.2       5.5.5.5         1377 0x80000002 0x5090
```

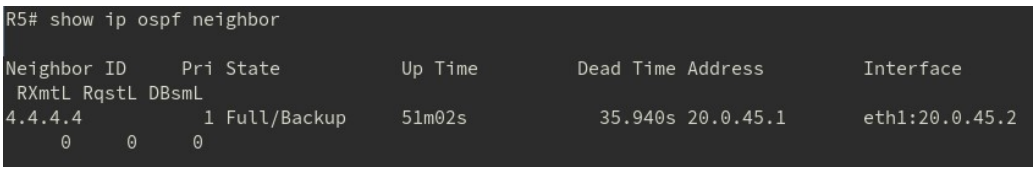
Rys. 11: Wynik wykonania komendy *show ip ospf database* na **R4**

```
R4# show ip ospf neighbor

Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
 RXmtL RqstL DBsmL
3.3.3.3           1 Full/Backup     50m54s            35.013s 20.0.34.1       eth1:20.0.34.2
     0     0     0
5.5.5.5           1 Full/DR         50m53s            35.012s 20.0.45.2       eth2:20.0.45.1
     0     0     0
```

Rys. 12: Wynik wykonania komendy *show ip ospf neighbors* na **R4**

```
R5# show ip ospf database

       OSPF Router with ID (5.5.5.5)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
1.1.1.1         1.1.1.1         1397 0x80000005 0x04e0 2
2.2.2.2         2.2.2.2         1401 0x80000006 0xa51e 2
3.3.3.3         3.3.3.3         1400 0x8000000c 0xc8ff 4
4.4.4.4         4.4.4.4         1416 0x80000009 0x658d 3
5.5.5.5         5.5.5.5         1403 0x80000005 0x3b38 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
20.0.13.2       3.3.3.3         1380 0x80000002 0x23f9
20.0.23.2       3.3.3.3         1390 0x80000002 0xd638
20.0.34.2       4.4.4.4         1406 0x80000002 0x9364
20.0.45.2       5.5.5.5         1383 0x80000002 0x5090
```

Rys. 13: Wynik wykonania komendy *show ip ospf database* na **R5**



Rys. 14: Wynik wykonania komendy *show ip ospf neighbors* na **R5**

C: Zadania do wykonania

C1: MPLS w AS400

Zadanie polega na skonfigurowaniu MPLS w AS 400, a następnie sprawdzić za pomocą komendy *ping* czy jest połączenie między **R2** i **R5**.

```
R1# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 3.3.3.3         OPERATIONAL 3.3.3.3         00:02:10
```

Rys. 15: Wynik wykonania komendy *show impls ldp neighbors* na **R1**

```
R2# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 3.3.3.3         OPERATIONAL 3.3.3.3         00:02:32
```

Rys. 16: Wynik wykonania komendy *show mpls ldp neighbors* na **R2**

```
R3# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 1.1.1.1         OPERATIONAL 1.1.1.1         00:02:45
ipv4 2.2.2.2         OPERATIONAL 2.2.2.2         00:02:38
ipv4 4.4.4.4         OPERATIONAL 4.4.4.4         00:01:48
```

Rys. 17: Wynik wykonania komendy *show mpls ldp neighbors* na **R3**

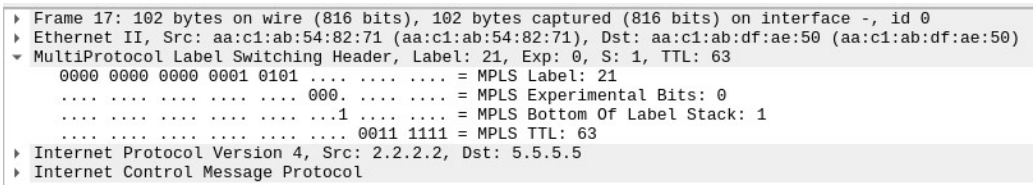
```
R4# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 3.3.3.3         OPERATIONAL 3.3.3.3         00:01:55
ipv4 5.5.5.5         OPERATIONAL 5.5.5.5         00:01:11
```

Rys. 18: Wynik wykonania komendy *show mpls ldp neighbors* na **R4**

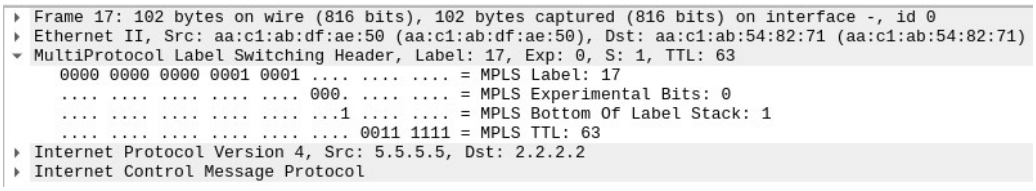
```
R5# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 4.4.4.4         OPERATIONAL 4.4.4.4         00:01:18
```

Rys. 19: Wynik wykonania komendy *show mpls ldp neighbors* na **R5**

Na powyższy zdjęciach można zauważyć, że wszystkie relacje sąsiedztwa w provider network zostały pomyślnie utworzone.



Rys. 20: Pakiet (request) po enkapsulacji MPLS



Rys. 21: Pakiet (reply) po enkapsulacji MPLS

Dzięki konfiguracji MPLS w AS400, można teraz przesyłać pakiety na podstawie MPLS labels.

C2: BGP FREE CORE

Polecenie polega na utworzeniu BGP-free core w AS400, co oznacza połączenie wszystkich **PE** ze sobą za pomocą **iBGP**. Przykładowa konfiguracja wykonana dla **R1** podczas tego ćwiczenia została zamieszczona poniżej. Konfiguracja dla pozostałych **PE** jest analogiczna, różnią się tylko adresy loopback.

```
R1# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 1.1.1.1, local AS number 400 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 2, using 1435 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
2.2.2.2         4        400         5         5        0    0    0 00:02:35            0        0 N/A
5.5.5.5         4        400         5         5        0    0    0 00:02:58            0        0 N/A

Total number of neighbors 2
```

Rys. 22: Przykładowa konfiguracja na **R1**

Po skonfigurowaniu wszystkich routerów sprawdzono czy nawiązały się wszystkie relacje sąsiedztwa.

```
R1# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 1.1.1.1, local AS number 400 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 2, using 1435 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
2.2.2.2         4        400         5         5        0    0    0 00:02:35            0        0 N/A
5.5.5.5         4        400         5         5        0    0    0 00:02:58            0        0 N/A

Total number of neighbors 2
```

Rys. 23: Wynik wykonania komendy *show ip bgp summary* na **R1**

```
R2# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 2.2.2.2, local AS number 400 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 2, using 1435 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
1.1.1.1         4        400         6         7        0    0    0 00:03:01            0        0 N/A
5.5.5.5         4        400         5         6        0    0    0 00:02:38            0        0 N/A

Total number of neighbors 2
```

Rys. 24: Wynik wykonania komendy *show ip bgp summary* na **R2**

```
R5# show ip bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 5.5.5.5, local AS number 400 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 2, using 1435 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
1.1.1.1         4        400         6         6        0    0    0 00:03:31            0        0 N/A
2.2.2.2         4        400         5         6        0    0    0 00:02:45            0        0 N/A

Total number of neighbors 2
```

Rys. 25: Wynik wykonania komendy *show ip bgp summary* na **R5**

C3: L3 VPN dla klienta blue

W ramach tego trzeba stworzyć VPN warstwy 3 dla klienta blue. W tym celu trzeba stworzyć sesje eBGP pomiędzy **PE** i obsługiwanymi przez nie **CE**, a następnie ustawić odpowiednie route distinguishery oraz route tragety. Dokładna konfiguracja na routerach **PE** i **CE** została przedstawiona poniżej.

```
router bgp 400
 neighbor 2.2.2.2 remote-as 400
 neighbor 2.2.2.2 update-source 1.1.1.1
 neighbor 5.5.5.5 remote-as 400
 neighbor 5.5.5.5 update-source 1.1.1.1
 !
 address-family ipv4 unicast
  neighbor 2.2.2.2 next-hop-self
  neighbor 5.5.5.5 next-hop-self
 exit-address-family
 !
 address-family ipv4 vpn
  neighbor 2.2.2.2 activate
  neighbor 5.5.5.5 activate
 exit-address-family
exit
!
router bgp 400 vrf blue
 no bgp ebgp-requires-policy
 neighbor 10.0.11.2 remote-as 101
 !
 address-family ipv4 unicast
  label vpn export auto
  rd vpn export 400:11
  rt vpn import 400:12 400:13
  rt vpn export 400:11
  export vpn
  import vpn
 exit-address-family
exit
```

Rys. 26: Konfiguracja na **R1**

```
router bgp 101
 no bgp ebgp-requires-policy
 neighbor 10.0.11.1 remote-as 400
 !
 address-family ipv4 unicast
  network 172.16.1.0/24
  neighbor 10.0.11.1 allowas-in
 exit-address-family
exit
```

Rys. 27: Konfiguracja na **CBL1**

```
router bgp 400
 neighbor 1.1.1.1 remote-as 400
 neighbor 1.1.1.1 update-source 2.2.2.2
 neighbor 5.5.5.5 remote-as 400
 neighbor 5.5.5.5 update-source 2.2.2.2
 !
 address-family ipv4 unicast
  neighbor 1.1.1.1 next-hop-self
  neighbor 5.5.5.5 next-hop-self
 exit-address-family
 !
 address-family ipv4 vpn
  neighbor 1.1.1.1 activate
  neighbor 5.5.5.5 activate
 exit-address-family
exit
!
router bgp 400 vrf blue
 no bgp ebgp-requires-policy
 neighbor 10.0.21.2 remote-as 102
 !
 address-family ipv4 unicast
  label vpn export auto
  rd vpn export 400:12
  rt vpn import 400:11 400:13
  rt vpn export 400:12
  export vpn
  import vpn
 exit-address-family
exit
!
```

Rys. 28: Konfiguracja na **R2**

```
router bgp 102
 no bgp ebgp-requires-policy
 neighbor 10.0.21.1 remote-as 400
 !
 address-family ipv4 unicast
  network 172.16.2.0/24
  neighbor 10.0.21.1 allowas-in
 exit-address-family
exit
```

Rys. 29: Konfiguracja na **CBL2**

```
router bgp 400
 neighbor 1.1.1.1 remote-as 400
 neighbor 1.1.1.1 update-source 5.5.5.5
 neighbor 2.2.2.2 remote-as 400
 neighbor 2.2.2.2 update-source 5.5.5.5
 !
 address-family ipv4 unicast
  neighbor 1.1.1.1 next-hop-self
  neighbor 2.2.2.2 next-hop-self
 exit-address-family
 !
 address-family ipv4 vpn
  neighbor 1.1.1.1 activate
  neighbor 2.2.2.2 activate
 exit-address-family
exit
!
router bgp 400 vrf blue
 no bgp ebgp-requires-policy
 neighbor 10.0.51.2 remote-as 103
 !
 address-family ipv4 unicast
  label vpn export auto
  rd vpn export 400:13
  rt vpn import 400:12 400:11
  rt vpn export 400:13
  export vpn
  import vpn
 exit-address-family
exit
```

Rys. 30: Konfiguracja na **R5**

```
router bgp 103
 no bgp ebgp-requires-policy
 neighbor 10.0.51.1 remote-as 400
 !
 address-family ipv4 unicast
  network 172.16.3.0/24
  neighbor 10.0.51.1 allowas-in
 exit-address-family
exit
```

Rys. 31: Konfiguracja na **CBL3**

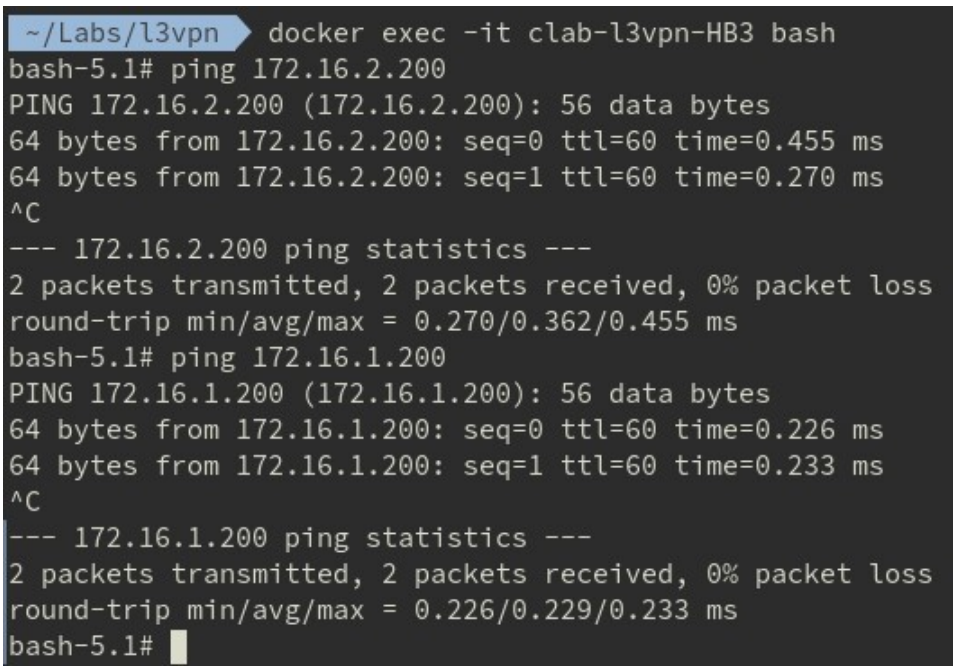
Po dokonaniu powyższej konfiguracji dokonano testu połączenia pomiędzy wszystkimi hostami klienta blue.

```
~/Labs/l3vpn docker exec -it clab-l3vpn-HB1 bash
bash-5.1# ping 172.16.2.200
PING 172.16.2.200 (172.16.2.200): 56 data bytes
64 bytes from 172.16.2.200: seq=0 ttl=60 time=0.630 ms
64 bytes from 172.16.2.200: seq=1 ttl=60 time=0.231 ms
^C
--- 172.16.2.200 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.231/0.430/0.630 ms
bash-5.1# ping 172.16.3.200
PING 172.16.3.200 (172.16.3.200): 56 data bytes
64 bytes from 172.16.3.200: seq=0 ttl=60 time=0.180 ms
64 bytes from 172.16.3.200: seq=1 ttl=60 time=0.229 ms
^C
--- 172.16.3.200 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.180/0.204/0.229 ms
bash-5.1#
```

Rys. 32: Wynik wykonania *ping* na **HB1** do pozostałych hostów w sieci

```
~/Labs/l3vpn docker exec -it clab-l3vpn-HB2 bash
bash-5.1# ping 172.16.1.200
PING 172.16.1.200 (172.16.1.200): 56 data bytes
64 bytes from 172.16.1.200: seq=0 ttl=60 time=0.365 ms
64 bytes from 172.16.1.200: seq=1 ttl=60 time=0.211 ms
^C
--- 172.16.1.200 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.211/0.288/0.365 ms
bash-5.1# ping 172.16.3.200
PING 172.16.3.200 (172.16.3.200): 56 data bytes
64 bytes from 172.16.3.200: seq=0 ttl=60 time=0.174 ms
64 bytes from 172.16.3.200: seq=1 ttl=60 time=0.142 ms
^C
--- 172.16.3.200 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.142/0.158/0.174 ms
bash-5.1#
```

Rys. 33: Wynik wykonania *ping* na **HB2** do pozostałych hostów w sieci



Rys. 34: Wynik wykonania *ping* na **HB3** do pozostałych hostów w sieci

C4: L3 VPN dla klienta red

Zadanie ponowienie polegało na utworzeniu VPN warstwy 3, ale tym razem dla klienta red. Sposób konfiguracji jest analogiczny jak dla klienta blue. Jedyną różnicą są adresy IP, route (ustawione na 400:NumerAsCE) oraz route targety (ustawione na 400:2X gdzie X to numer routera, z którym łączy się CE). Po dokonaniu konfiguracji zostało sprawdzone połączenie między **HR1** oraz **HR3** przy użyciu komend *ping* i *traceroute*.

```
~/Labs/l3vpn docker exec -it clab-l3vpn-HR1 bash
bash-5.1# ping 172.16.3.200
PING 172.16.3.200 (172.16.3.200): 56 data bytes
64 bytes from 172.16.3.200: seq=0 ttl=60 time=0.188 ms
64 bytes from 172.16.3.200: seq=1 ttl=60 time=0.203 ms
64 bytes from 172.16.3.200: seq=2 ttl=60 time=0.169 ms
64 bytes from 172.16.3.200: seq=3 ttl=60 time=0.181 ms
^C
--- 172.16.3.200 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.169/0.185/0.203 ms
bash-5.1# traceroute 172.16.3.200
traceroute to 172.16.3.200 (172.16.3.200), 30 hops max, 46 byte packets
 1  172.16.1.1 (172.16.1.1)  0.008 ms  0.017 ms  0.013 ms
 2  10.0.12.1 (10.0.12.1)  0.013 ms  0.008 ms  0.206 ms
 3  *  *  *
 4  *  *  *
 5  172.16.3.200 (172.16.3.200)  0.008 ms  0.008 ms  0.006 ms
bash-5.1#
```

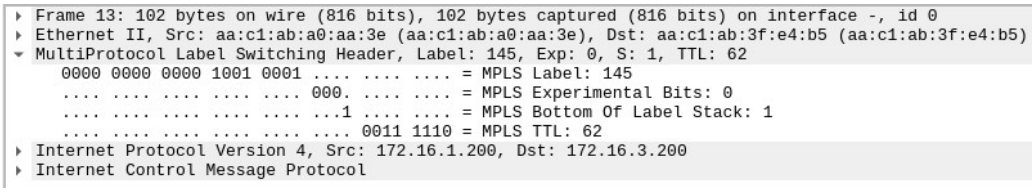
Rys. 35: Wynik wykonania *ping* i *traceroute* na **HB1** w kierunku **HB3**

```
bash-5.1# ping 172.16.1.200
PING 172.16.1.200 (172.16.1.200): 56 data bytes
64 bytes from 172.16.1.200: seq=0 ttl=60 time=0.151 ms
64 bytes from 172.16.1.200: seq=1 ttl=60 time=0.200 ms
^C
--- 172.16.1.200 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.151/0.175/0.200 ms
bash-5.1# traceroute 172.16.1.200
traceroute to 172.16.1.200 (172.16.1.200), 30 hops max, 46 byte packets
 1  172.16.3.1 (172.16.3.1)  0.007 ms  0.016 ms  0.013 ms
 2  10.0.52.1 (10.0.52.1)  0.013 ms  0.008 ms  0.006 ms
 3  *  *  *
 4  *  *  *
 5  172.16.1.200 (172.16.1.200)  0.010 ms  0.019 ms  0.013 ms
bash-5.1#
```

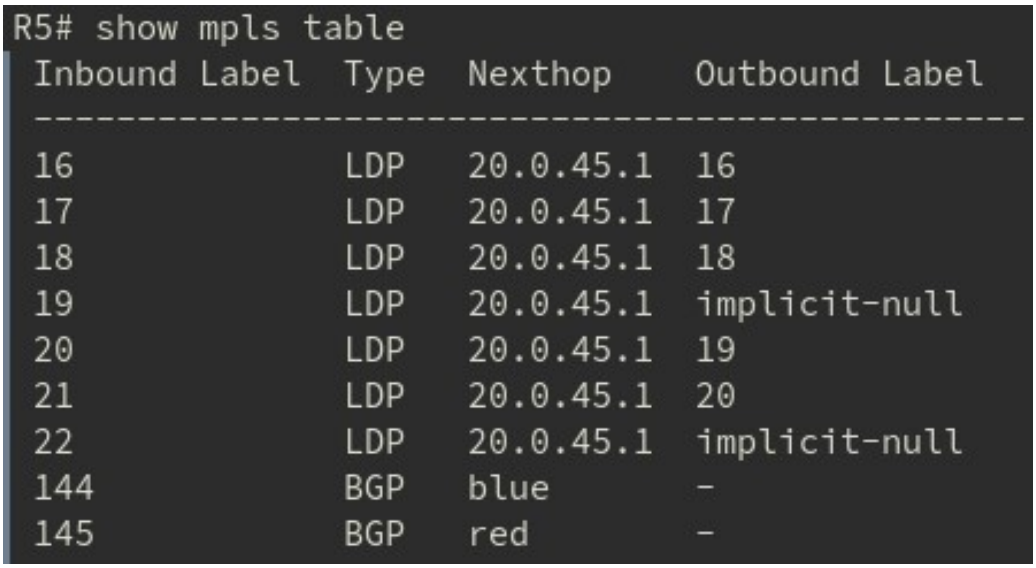
Rys. 36: Wynik wykonania *ping* i *traceroute* na **HB3** w kierunku **HB1**

Jak można zauważyć, połączenie jest możliwe. Ciekawić mogą "dziury" powstałe w trakcie wykorzystania komendy *traceroute*. Wynikają one z faktu, że w przypadku MPLS etykiety MPLS są dodawane przed nagłówkiem IP. Następnie routery wzdłuż ścieżki MPLS sprawdzają tylko etykiety MPLS, a nie nagłówek IP. Oznacza to, że wartość TTL w nagłówku IP nie jest zmniejszana przez routery MPLS. W rezultacie, gdy pakiet traceroute dociera do routera MPLS, TTL nie osiąga zera i nie jest generowany komunikat "Time Exceeded".

Polecenie wymagało również, aby na podstawie obserwacji ruchu na interfejsie **eth1** routera **R5** wyjaśnić w jaki sposób pakiety kierowane są do odpowiedniego hosta pomimo faktu, że w wykorzystywanej topologi istnieją routery dla klienta blue i red, które mają takie same adresy IP.



Rys. 37: Pakiet zaobserwowany w trakcie *ping* na **HR3** z **HR1**



Rys. 38: Wynik wykonania *show mpls table* na **R5**

Jak można zauważyć w momencie, kiedy komunikują się ze sobą hosty klienta red wykorzystywany jest label o wartości 145. Taki sam label posiada **R5** oraz pozostałe routery w swoich tablicach **MPLS** i to właśnie dzięki nim są one w rozróżnić routing dla klienta blue i red,a w konsekewncji pokierować pakiety w taki spsoób, żeby dotrały do zamierzonego odbiorcy.