INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Ausarbeitung

zum

Rechnernetzepraktikum

im WS 24/25

Aufgabenblock 2

Gruppe: 05
Zeyu Li
Hanzhang Ma

Contents

1 A300 Fragmentierung und IP-Tunneling (Theorie)

i) In wie viele Fragmente wird ein IP-Paket mit Größe 9000 Byte zerlegt, um über ein Ethernet mit MTU = 1500 Byte übertragen zu werden? Geben Sie eine vollständige Rechnung an!

Antwort:

We first define symbols for calculation.

- S_{total}: Total size of the IP packet (in bytes), S_total=9000.
- S_{header}: Size of the IP header (in bytes). Here we assume $S_{\text{header}} = 20$.
- S_{MTU}: Size of the MTU (in bytes), $S_{\text{MTU}} = 1500$.
- $S_{\text{data\_per\_fragment}}$: Maximum data size per fragment (in bytes).
- $N_{\text{fragments}}$: Total number of fragments.

Now we start calculation. The maximum data size per fragment is:

$$S_{\text{data\_per\_fragment}} = S_{\text{MTU}} - S_{\text{header}} = 1500 - 20 = 1480 \text{ bytes}$$

The total size of data section is:

$$S_{\text{data\_total}} = S_{\text{total}} - S_{\text{header}} = S_{\text{data\_total}} = 9000 - 20 = 8980 \text{ bytes}$$

Each fragment can carry $S_{\text{data\_per\_fragment}}$ bytes of data. The total number of fragments required is:

$$N_{\text{fragments}} = \lceil \frac{S_{\text{data\_total}}}{S_{\text{data\_per\_fragment}}} \rceil = N_{\text{fragments}} = \lceil \frac{8980}{1480} \rceil = \lceil 6.068 \rceil = 7$$

ii) Nennen Sie drei Gründe dafür, dass Netzbetreiber IP-Fragmentierung in ihren Netzen verbieten. Erläutern Sie diese Gründe!

1. One reason is that there is no timeout retransmission mechanism in the IP layer. If the IP layer slices a packet, as long as one slice is lost, it can only rely on the transport layer to retransmit it, and the result is that all the slices have to be retransmitted again, which is a bit costly.

2. IP fragmentation can be misused for a variety of network attacks, such as Fragmentation Attacks and Tiny Fragment Attacks, which use fragmentation to bypass firewalls or Intrusion Detection Systems (IDS).

3. One reason can be they want to reduce cross-network boundary issues. In

multi-operator networks or international links, MTU settings may be inconsistent, resulting in the need for intermediate nodes to slice and dice traffic. The performance of the transport may be seriously affected after fragmentation, and at the same time, it will increase the debugging difficulty at both ends of the link. Different carriers may have different rules for handling fragmented packets, which may cause some fragments to be prioritized and discarded, further damaging the overall transmission effect.

iii) Wie wird in modernen TCP/IP-Implementierungen dafür gesorgt, dass Fragmentierung in der Regel nicht erforderlich ist?

Antwort:

In modern TCP/IP implementations, path MTU detection (PMTUD) is a common method used to avoid the need for fragmentation. By setting the "Don't Fragment" (DF) bit in the IP packet, the sender can dynamically discover the minimum MTU supported by each node in the network path. When a packet is too large to be transmitted, the intermediate router returns an ICMP message prompting the sender to resize the packet to fit the path MTU, thus preventing fragmentation by the intermediate router.

Additionally, segmentation layer path MTU detection (PLPMTUD) provides a more robust alternative, especially when some networks block ICMP messages. PLPMTUD dynamically adjusts packet sizes without relying on ICMP messages by testing whether packets of different sizes successfully reach their destination. This method effectively avoids transmission failure problems caused by ICMP messages being lost or intercepted.

TCP's fragmentation mechanism further shifts the responsibility for data fragmentation from the network layer to the transport layer. By negotiating the MSS (Maximum Segment Size) when the connection is established, TCP ensures that the size of the data segments it generates does not exceed the MTU of the underlying network, essentially avoiding the need for IP layer fragmentation. This mechanism greatly improves transmission efficiency and reliability.

At the same time, many networks default to standard MTU values, such as Ethernet's 1500 bytes, which have become the design baseline for mainstream network equipment and protocols. By ensuring that the data packet size is within the standard MTU range, fragmentation problems caused by MTU mismatch can be greatly reduced and transmission compatibility improved.

Modern network equipment also limits fragmentation behavior through configuration of firewall and router policies. Many devices force the DF bit of IP packets to be set, requiring the sender to adapt to the path MTU, or simply drop fragmented packets. This strategy both reduces network complexity and prevents security threats such as sharding attacks.

2 A301 IPv6 (Theorie)

i) Erklären Sie anhand eines Beispiels auf einer Ihrer VMs wie link-local Adressen aus der MAC-Adresse abgeleitet werden!

Antwort: We use pc2 eth1's Link-Local IPv6 address as example:

```
root@pc2:~# ip a
...
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
↪  state UP group default qlen 1000
    link/ether 00:16:3e:00:00:04 brd ff:ff:ff:ff:ff:ff
    inet 10.5.1.2/24 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:4/64 scope link
       valid_lft forever preferred_lft forever
...
```

We notice the MAC address is 00:16:3e:00:00:04. We need to translate it to EUI-64. We first split the MAC address into two parts:

$$\text{First 3 bytes: } 00:16:3e$$
$$\text{Last 3 bytes: } 00:00:04$$

Then we insert a fixed ff:fe in the middle:

$$00:16:3e:ff:fe:00:00:04$$

After that, we adjust the seventh bit (U/L bit) of the first byte. The original first byte is 00 and its binary is 00000000. Invert the seventh bit (left to right): 00000010 (i.e., it becomes 02). Now the MAC address is converted into:

$$02:16:3e:FF:FE:00:00:04$$

The final, EUI-64 address portion is:

$$0216:3eff:fe00:0004$$

Finally, we combine the prefix fe80::/10 with the EUI-64 part to get the link-local IPv6 address:

$$fe80::0216:3eff:fe00:4$$

ii) Wie implementiert IPv6 „Broadcasts"?

Antwort: IPV6 implements communication through three types of addresses: unicast, multicast, and anycast.

Multicast is used to replace the broadcast function of IPv4.

Multicast is a communication method that sends packets to all devices belonging to a specific group.

The IPv6 multicast address starts with FF00::/8, which is divided into link-local multicast, organization-local multicast, and global multicast according to the scope and range.

IPv6 Implementation

1.Discover all devices on the network

In IPv4, this is done using broadcast. In IPv6, the multicast address FF02::1 can be used, which sends the message to all nodes on the same link.

2. Neighbor Discovery Protocol

IPv6 uses the Neighbor Discovery Protocol to replace the ARP protocol in IPv4. Neighbor Discovery Protocol uses multicast addresses for communication.

3. Send a specific service request

A specific multicast address (FF02::1:2) is used in IPv6 to communicate with the DHCPv6 server.

IPV4 broadcast messages are sent to all hosts in the same subnet, causing network congestion and performance degradation in large-scale networks. Broadcasts are effective for all hosts and may pose security issues.

Therefore, broadcast is removed in IPV6, and three address types are used to achieve communication

iii) Welches sind die privaten Adressbereiche in IPv6, analog zu 10.0.0.0/8, 172.16.0.0/12 und 192.168.0.0/16 in IPv4?

Antwort:

In IPv6, an address that is functionally similar to IPv4's private address ranges is the Unique Local Address (ULA).

The unique local address range for IPv6 is: FC00::/7.

This range is defined in RFC 4193 and is used as a replacement for IPv4 private addresses, and is divided into two sub-ranges:

$FC00$ :: /8: Reserved and currently unallocated.

$FD00$ :: /8: Common range, manually assigned by users.

Therefore, IPv6 private addresses are usually in the form of FDxx:xxxx:xxxx::/48.

iv) Für besondere Zwecke, außer für den privaten Gebrauch, sind noch weitere Bereiche

reserviert. Wie teilt sich der IPv6 Addressraum auf? Hinweis: IANA, ignorieren Sie die vom IETF reservierten Bereiche.

Antwort:

Address range	**Prefix**	**Usage**
::/128	Single address	Unspecified Address
::1/128	Single address	Loopback Address, used for device self-communication, similar to 127.0.0.1 in IPv4
::/96	Special Purpose	IPv4-Mapped IPv6 Address
100::/64	Special Purpose	Reserved for Discard-Only Address Block
2000::/3	Global unicast address	A globally routable unicast address allocation range that is the primary IPv6 address space allocated to the Internet.
3000::/4	Special Purpose	Reserved for future use
fc00::/7	Local unicast address	Unique Local Address (ULA)
fe80::/10	Link-local unicast address	Used for link-local communication (Link-Local Address)
ff00::/8	Multicast Address	Scope for multicast communication, including link-local and global multicast
2001:db8::/32	Document purpose address	Dedicated to documentation and sample code, similar to 192.0.2.0/24 for IPv4
2002::/16	6to4 Translation Address	Used for IPv6 over IPv4 networks
ffx0::/12	Anycast Address	Addresses assigned by a specific network and used in special communication scenarios

3 A302 Fragmentierung und Tunneling

i) Legen Sie den Pfad (pc1, router1, router4, router3, router2, pc2) an, so dass Nachrichten zwischen pc1 und pc2 vermittelt werden und zeigen Sie mittels traceroute, dass die Nachrichten entlang dieses Pfades vermittelt werden!

Antwort: We assign the IP and add route rules on each device and interface. The IP address and eth information we have assigned are shown in A302 ii).

The codes of how we assigned are shown as follows.

```
assign_ip(){
        local dev=$1
        local eth_num=$2
        local ip=$3
        ssh $dev "ip link set dev eth$eth_num up"
        ssh $dev "ip addr add $ip dev eth$eth_num"
        echo "dev $dev eth$eth_num assign $ip"
}
add_route(){
        local dev=$1
        local ip_to=$2
        local ip_from=$3
        local eth_num=$4
        ssh $dev "ip route add $ip_to via $ip_from dev eth$eth_num"
        echo "dev $dev add route to $ip_to"
}

ssh "router4" "sysctl -w net.ipv4.ip_forward=1"

assign_ip "pc1" "1" "10.5.1.1/24"
assign_ip "router1" "1" "10.5.1.2/24"

assign_ip "router1" "4" "10.5.2.1/24"
assign_ip "router4" "1" "10.5.2.2/24"

assign_ip "router4" "3" "10.5.3.1/24"
assign_ip "router3" "4" "10.5.3.2/24"

assign_ip "router3" "3" "10.5.4.1/24"
assign_ip "router2" "3" "10.5.4.2/24"

assign_ip "router2" "1" "10.5.5.1/24"
```

```
assign_ip "pc2" "1" "10.5.5.2/24"

ssh "pc1" "ip route add default via 10.5.1.2"
add_route "router1" "10.5.5.0/24" "10.5.2.2" 4

add_route "router4" "10.5.1.0/24" "10.5.2.1" 1
add_route "router4" "10.5.5.0/24" "10.5.3.2" 3

add_route "router3" "10.5.1.0/24" "10.5.3.1" 4
add_route "router3" "10.5.5.0/24" "10.5.4.2" 3

add_route "router2" "10.5.1.0/24" "10.5.4.1" 3
ssh "pc2" "ip route add default via 10.5.5.1"
```

Finally we run the traceroute command. The output displayed that the packet sequentially passes through router1, then router4, router3, router2, and finally arrive at pc2.

```
traceroute to 10.5.5.2 (10.5.5.2), 30 hops max, 60 byte packets
 1  10.5.1.2 (10.5.1.2)  3.391 ms  3.322 ms  3.272 ms
 2  10.5.2.2 (10.5.2.2)  3.223 ms  3.168 ms  3.107 ms
 3  10.5.3.2 (10.5.3.2)  3.061 ms  3.015 ms  3.004 ms
 4  10.5.4.2 (10.5.4.2)  2.959 ms  2.913 ms  2.866 ms
 5  10.5.5.2 (10.5.5.2)  2.822 ms  2.777 ms  2.695 ms
```

ii) Bestimmen Sie mittels ip die MTU der genutzten Schnittstellen! Erstellen Sie für die Ausarbeitung eine Tabelle mit den Spalten: Rechnername, Schnittstelle, IP-Adresse, MTU.

We use command `ip link show dev eth#` on each device to show the MTU infos.

```
ssh "router1" "ip link show dev eth1"
```

The table is generated from the information of A302 i), and the MTU information is from the "ip link show" command.

Computer Name	Interface	IP Address	MTU
pc1	eth1	10.5.1.1	1500
router1	eth1	10.5.1.2	1500
router1	eth4	10.5.2.1	1500
router4	eth1	10.5.2.2	1500
router4	eth3	10.5.3.1	1500
router3	eth4	10.5.3.2	1500
router3	eth3	10.5.4.1	1500
router2	eth3	10.5.4.2	1500
router2	eth1	10.5.5.1	1500
pc2	eth1	10.5.5.2	1500

iii) Beschränken Sie nun mittels ip die MTU von router3.eth3 auf 1000 Bytes und die MTU von router4.eth3 auf 1100 Bytes!

Antwort:

We use `ip link set dev eth3 mtu #` on router 3 and router 4 to change the MTU number.

```
# on node rnp
ssh "router3" "ip link set dev eth3 mtu 1000"
ssh "router4" "ip link set dev eth3 mtu 1100"

ssh "router3" "ip link show eth3"
ssh "router4" "ip link show eth3"
```

Here is the output.

```
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1000 qdisc fq_codel
↪  state UP mode DEFAULT group default qlen 1000
    link/ether 00:16:3e:00:00:20 brd ff:ff:ff:ff:ff:ff
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1100 qdisc pfifo_fast
↪  state UP mode DEFAULT group default qlen 1000
    link/ether 00:16:3e:00:00:25 brd ff:ff:ff:ff:ff:ff
```

iv) Beschreiben Sie das Verhalten der Rechner, wenn Sie IP-Nachrichten mit 1200 Bytes Nutzdaten von pc1 an pc2 schicken! Nutzen Sie ping um Nachrichten dieser Größe zu erzeugen. Achten Sie darauf, dass ping das Don't Fragment Flag setzt.

Antwort:

When a packet arrives at a router4 eth4 with an MTU of 1100 bytes, the device discards the packet because the packet size exceeds the MTU and the DF flag prohibits fragmentation. The device sends an ICMP "Fragmentation Needed" message back to PC1 informing it that the path MTU is 1100.

We use `-M do` parameter to ensure ping packet has the "Don't Fragment" flag.

```
ssh "pc1" "ping -s 1200 -M do 10.5.5.2"
```

Here is the output of ping.

```
ping: local error: message too long, mtu=1100
ping: local error: message too long, mtu=1100
ping: local error: message too long, mtu=1100
ping: local error: message too long, mtu=1100
ping: local error: message too long, mtu=1100
```

v) Wiederholen Sie den ping-Versuch ohne gesetztes Don't Fragment Flag. Wie verändert sich der Datenfluss im Vergleich zum vorherigen Versuch?

Antwort:

If DF flag not set, packets can be fragmented by the intermediate device according to the path MTU. If the path MTU is less than the total packet size (e.g., greater than 1100 or 1000 bytes), the packet will be divided into pieces, each of which does not exceed the MTU on the link. At the receiving end (target host), all fragments are reassembled into the original packet.

```
    ssh "pc1" "ping -s 1200 10.5.5.2"

PING 10.5.5.2 (10.5.5.2) 1200(1228) bytes of data.
From 10.5.2.2 icmp_seq=1 Frag needed and DF set (mtu = 1100)
1208 bytes from 10.5.5.2: icmp_seq=2 ttl=60 time=16.7 ms
1208 bytes from 10.5.5.2: icmp_seq=3 ttl=60 time=1.53 ms
1208 bytes from 10.5.5.2: icmp_seq=4 ttl=60 time=1.61 ms
1208 bytes from 10.5.5.2: icmp_seq=5 ttl=60 time=1.38 ms
1208 bytes from 10.5.5.2: icmp_seq=6 ttl=60 time=1.31 ms
1208 bytes from 10.5.5.2: icmp_seq=7 ttl=60 time=1.26 ms
```

We observe the data stream by tcpdump on pc1:

```
root@pc1:~# tcpdump -i eth1
...
11:11:30.695599 IP 10.5.1.1 > 10.5.5.2: ICMP echo request, id
↪  42017, seq 12, length 1080
11:11:30.695612 IP 10.5.1.1 > 10.5.5.2: icmp
11:11:30.696973 IP 10.5.5.2 > 10.5.1.1: ICMP echo reply, id 42017,
↪  seq 12, length 1208
```

We notice that the request packet is divided into two packets.

vi) Weisen Sie dem Interface router3.eth3 eine beliebige IPv6 Adresse zu. Ist der Vorgang erfolgreich? Wenn nein, was ist die Ursache? Ziehen sie hierzu auch die beschriebenen Rahmenbedingungen aus [RFC 2460] in Betracht.

Antwort:

According to RFC2460: "IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6."

Since router3.eth3's MTU is less than 1280, when we try to assign an IPv6 address on router3.eth3, we got the error message:

```
root@router3:~# ip a add 2001:db8:5::3/64 dev eth3
RTNETLINK answers: Invalid argument
```

vii) Konfigurieren Sie einen IP-Tunnel zwischen router1 und router2 und konfigurieren Sie die Systeme so, dass sämtliche IP-Nachrichten zwischen pc1 und pc2 durch den Tunnel übertragen werden!

Antwort: First we need to install `kmod-gre` on router 1 and router 2 to let these two devices support gre tunnel.

```
root@router1:~# ip route add default via 192.168.0.254 dev eth0
root@router1:~# opkg update
...
Signature check passed.
root@router1:~# opkg install kmod-gre
...
Configuring kmod-iptunnel.
Configuring kmod-gre.
root@router1:~# ip route del default via 192.168.0.254 dev eth0
```

Then we need to add some route rules to make router 1 can commnunicate with router 2. The bash function `add_route` can be found at A302 i).

```
source functions.sh

add_route "router1" "10.5.3.0/24" "10.5.2.2" 4
add_route "router1" "10.5.4.0/24" "10.5.2.2" 4

add_route "router4" "10.5.4.0/24" "10.5.3.2" 3

add_route "router3" "10.5.2.0/24" "10.5.3.1" 4

add_route "router2" "10.5.2.0/24" "10.5.4.1" 3
add_route "router2" "10.5.3.0/24" "10.5.4.1" 3
```

Then we add two tunnels from router 1 and router 2 and from router 2 to router 1. After that, we assign IP for two gre devices.

```
root@router1:~# ip tunnel add gre1 mode gre local 10.5.2.1 remote
↪  10.5.4.2 ttl 255
root@router1:~# ip link set gre1 up
root@router1:~# ip a add 10.5.10.1/24 dev gre1

root@router2:~# ip tunnel add gre1 mode gre local 10.5.4.2 remote
↪  10.5.2.1 ttl 255
root@router2:~# ip link set gre1 up
root@router2:~# ip addr add 10.5.10.2/24 dev gre1
```

viii) Zeigen Sie mit traceroute, dass alle Nachrichten durch den Tunnel übertragen werden!

Antwort: We need to add two route rules to let gre process the packet from pc1 to pc2.

```
root@router1:~# ip route del 10.5.5.0/24 via 10.5.2.2
root@router1:~# ip route add 10.5.5.0/24 via 10.5.10.2

root@router2:~# ip route del 10.5.1.0/24 via 10.5.4.1 dev eth3
root@router2:~# ip route add 10.5.1.0/24 via 10.5.10.1 dev gre1
```

Now we start traceroute.

```
root@pc1:~# traceroute -i eth1 10.5.5.2
traceroute to 10.5.5.2 (10.5.5.2), 30 hops max, 60 byte packets
 1  10.5.1.2 (10.5.1.2)  3.789 ms  3.729 ms  3.679 ms
 2  10.5.10.2 (10.5.10.2)  7.403 ms  7.343 ms  6.401 ms
 3  10.5.5.2 (10.5.5.2)  6.347 ms  6.304 ms  6.252 ms
```

ix) Vergleichen Sie die von traceroute angegebenen Zwischenstationen aus den vorangegangenen Versuchen. Nennen Sie alle Unterschiede!

Antwort:

The packet send to 10.5.10.2 after 10.5.1.2, which is the IP address of `router1.eth1`. Then the packet is sent to pc2 directly. The traceroute entry decreased by two.

x) (Freiwillig) Fangen Sie mit Hilfe von tcpdump ein Paket ab, das den Tunnel gerade „betritt" bzw. „verlässt" und analysieren Sie die gekapselte Headerstruktur. Welche Unterschiede zwischen den Header-Daten des inneren versus denen des äußeren IP-Paketes stellen Sie fest?

Antwort:

4 A303 IPv6

i) Entfernen Sie alle IPv4-Adressen von den Schnittstellen eth1,2,3,4 auf allen Routern!

Antwort:

```
ssh "router1" "ip a del 10.5.1.2/24 dev eth1"
ssh "router1" "ip a del 10.5.2.1/24 dev eth4"

ssh "router4" "ip a del 10.5.2.2/24 dev eth1"
ssh "router4" "ip a del 10.5.3.1/24 dev eth3"

ssh "router3" "ip a del 10.5.3.2/24 dev eth4"
ssh "router3" "ip a del 10.5.4.1/24 dev eth3"

ssh "router2" "ip a del 10.5.4.2/24 dev eth3"
ssh "router2" "ip a del 10.5.5.1/24 dev eth1"
```

ii) Bilden Sie das Netz aus Abbildung 3.7 nach! Verwenden Sie DHCPv6-PD (prefix delegation) um ihr Netz hierarchisch aufzuteilen mit router3 als Zentrum.

Antwort: First we append following content to `//router3//etc/config/dhcp`:

```
config dhcp 'lan_router1'
    option interface 'lan_router1'
    option dhcpv6 'server'
    option ra 'server'
    option ra_management '1'
    list dhcpv6_pd '1:64'

config dhcp 'lan_router2'
    option interface 'lan_router2'
    option dhcpv6 'server'
    option ra 'server'
    option ra_management '1'
    list dhcpv6_pd '1:64'

config dhcp 'lan_router4'
    option interface 'lan_router4'
    option dhcpv6 'server'
```

```
    option ra 'server'
    option ra_management '1'
    list dhcpv6_pd '1:64'
```

Then we append those content to `//router3//etc/config/network`:

```
config interface 'lan_router1'
    option ifname 'eth2'
    option proto 'static'
    option ip6addr '2001:db8:5:1::1/64'
    option ip6assign '64'

config interface 'lan_router2'
    option ifname 'eth3'
    option proto 'static'
    option ip6addr '2001:db8:5:2::1/64'
    option ip6assign '64'

config interface 'lan_router4'
    option ifname 'eth4'
    option proto 'static'
    option ip6addr '2001:db8:5:3::1/64'
    option ip6assign '64'
```

Append these to `//router1//etc/config/network`:

```
config interface 'wan'
    option ifname 'eth3'
    option proto 'dhcpv6'
    option reqprefix 'auto'
    option peerdns '1'

config interface 'lan'
    option proto 'static'
    option ifname 'eth1'
    option ip6assign '64'
```

Under `//router1//etc/config/network`, for dhcp 'lan' part, we update the configuration to the follows:

```
config dhcp 'lan'
    option interface 'lan'
    option dhcpv6 'server'
```

```
    option ra 'server'
    option ra_management '1'



root@router3:~# ip -6 a

4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:1::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:19/64 scope link
       valid_lft forever preferred_lft forever
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:2::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:20/64 scope link
       valid_lft forever preferred_lft forever
6: eth4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:3::1/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:21/64 scope link
       valid_lft forever preferred_lft forever
```

On router1, router2 and router3, these three devices get the IPv6 addresses as well.

```
root@router1:~# ip -6 a
...
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:1:216:3eff:fe00:10/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 2001:db8:5:1::72a/128 scope global dynamic noprefixroute
       valid_lft 32031sec preferred_lft 32031sec
    inet6 fe80::216:3eff:fe00:10/64 scope link
       valid_lft forever preferred_lft forever

root@router2:~# ip -6 a
...
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:2:216:3eff:fe00:15/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 2001:db8:5:2::647/128 scope global dynamic noprefixroute
```

```
       valid_lft 31858sec preferred_lft 31858sec
    inet6 fe80::216:3eff:fe00:15/64 scope link
       valid_lft forever preferred_lft forever

root@router3:~# ip -6 a
...
5: eth3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:db8:5:3:216:3eff:fe00:25/64 scope global dynamic mngtmpaddr
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe00:25/64 scope link
       valid_lft forever preferred_lft forever
```

Use ping command. We first ping router2 from router3.

```
root@router3:~# ping -6 2001:db8:5:2:216:3eff:fe00:15 -c 5 -W 2
PING 2001:db8:5:2:216:3eff:fe00:15 (2001:db8:5:2:216:3eff:fe00:15): 56 data bytes
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=0 ttl=64 time=1.770 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=1 ttl=64 time=0.410 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=2 ttl=64 time=0.315 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=3 ttl=64 time=0.335 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=4 ttl=64 time=4.045 ms

--- 2001:db8:5:2:216:3eff:fe00:15 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.315/1.375/4.045 ms
```

Ping router1 from router3.

```
root@router3:~# ping -6 2001:db8:5:1:216:3eff:fe00:10 -c 5 -W 2
PING 2001:db8:5:1:216:3eff:fe00:10 (2001:db8:5:1:216:3eff:fe00:10): 56 data bytes
64 bytes from 2001:db8:5:1:216:3eff:fe00:10: seq=0 ttl=64 time=1.536 ms
64 bytes from 2001:db8:5:1:216:3eff:fe00:10: seq=1 ttl=64 time=0.314 ms
64 bytes from 2001:db8:5:1:216:3eff:fe00:10: seq=2 ttl=64 time=0.325 ms
64 bytes from 2001:db8:5:1:216:3eff:fe00:10: seq=3 ttl=64 time=0.303 ms
64 bytes from 2001:db8:5:1:216:3eff:fe00:10: seq=4 ttl=64 time=0.311 ms

--- 2001:db8:5:1:216:3eff:fe00:10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.303/0.557/1.536 ms
```

We now try to ping devices through router3.

To reach this goal, we first need to add route rules for the devices.

```
root@router1:~# ip -6 route add 2001:db8:5:2::/64 via 2001:db8:5:1::1
root@router1:~# ip -6 route add 2001:db8:5:3::/64 via 2001:db8:5:1::1
root@router1:~# ip -6 route show
2001:db8:5:1::/64 dev eth3 proto static metric 256 pref medium
unreachable 2001:db8:5:1::/64 dev lo proto static metric 2147483647 pref medium
2001:db8:5:2::/64 via 2001:db8:5:1::1 dev eth3 metric 1024 pref medium
2001:db8:5:3::/64 via 2001:db8:5:1::1 dev eth3 metric 1024 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev eth1 proto kernel metric 256 pref medium
fe80::/64 dev eth3 proto kernel metric 256 pref medium

root@router2:~# ip -6 route add 2001:db8:5:1::/64 via 2001:db8:5:2::1
root@router2:~# ip -6 route add 2001:db8:5:3::/64 via 2001:db8:5:2::1
root@router2:~# ip -6 route show
2001:db8:5:1::/64 via 2001:db8:5:2::1 dev eth3 metric 1024 pref medium
2001:db8:5:2::/64 dev eth3 proto static metric 256 pref medium
unreachable 2001:db8:5:2::/64 dev lo proto static metric 2147483647 pref medium
2001:db8:5:3::/64 via 2001:db8:5:2::1 dev eth3 metric 1024 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev eth1 proto kernel metric 256 pref medium
fe80::/64 dev eth3 proto kernel metric 256 pref medium


root@router4:~# ip -6 route add 2001:db8:5:1::/64 via 2001:db8:5:3::1
root@router4:~# ip -6 route add 2001:db8:5:2::/64 via 2001:db8:5:3::1
root@router4:~# ip -6 route show
::1 dev lo proto kernel metric 256 pref medium
2001:db8:5:1::/64 via 2001:db8:5:3::1 dev eth3 metric 1024 pref medium
2001:db8:5:2::/64 via 2001:db8:5:3::1 dev eth3 metric 1024 pref medium
2001:db8:5:3::/64 dev eth3 proto kernel metric 256 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev eth3 proto kernel metric 256 pref medium
fe80::/64 dev eth2 proto kernel metric 256 pref medium
fe80::/64 dev eth1 proto kernel metric 256 pref medium


root@router3:~# ip -6 route show
2001:db8:5:1::/64 dev eth2 proto kernel metric 256 pref medium
2001:db8:5:2::/64 dev eth3 proto kernel metric 256 pref medium
2001:db8:5:3::/64 dev eth4 proto kernel metric 256 pref medium
fe80::/64 dev eth0 proto kernel metric 256 pref medium
fe80::/64 dev eth2 proto kernel metric 256 pref medium
fe80::/64 dev eth3 proto kernel metric 256 pref medium
fe80::/64 dev eth4 proto kernel metric 256 pref medium
```

Now we start ping experiments.

```
# from router1 to router2
root@router1:~# ping -6 2001:db8:5:2:216:3eff:fe00:15 -c 5 -W 2
PING 2001:db8:5:2:216:3eff:fe00:15 (2001:db8:5:2:216:3eff:fe00:15): 56 data bytes
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=0 ttl=63 time=1.372 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=1 ttl=63 time=0.752 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=2 ttl=63 time=0.508 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=3 ttl=63 time=0.455 ms
64 bytes from 2001:db8:5:2:216:3eff:fe00:15: seq=4 ttl=63 time=0.655 ms

--- 2001:db8:5:2:216:3eff:fe00:15 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.455/0.748/1.372 ms

# from router1 to router3
root@router1:~# ping -6 2001:db8:5:3:216:3eff:fe00:25 -c 5 -W 2
PING 2001:db8:5:3:216:3eff:fe00:25 (2001:db8:5:3:216:3eff:fe00:25): 56 data bytes
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=0 ttl=63 time=2.013 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=1 ttl=63 time=0.687 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=2 ttl=63 time=0.645 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=3 ttl=63 time=0.592 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=4 ttl=63 time=0.664 ms

--- 2001:db8:5:3:216:3eff:fe00:25 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.592/0.920/2.013 ms


# from router2 to router3
root@router2:~# ping -6 2001:db8:5:3:216:3eff:fe00:25 -c 5 -W 2
PING 2001:db8:5:3:216:3eff:fe00:25 (2001:db8:5:3:216:3eff:fe00:25): 56 data bytes
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=0 ttl=63 time=1.776 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=1 ttl=63 time=0.678 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=2 ttl=63 time=0.706 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=3 ttl=63 time=0.636 ms
64 bytes from 2001:db8:5:3:216:3eff:fe00:25: seq=4 ttl=63 time=1.514 ms

--- 2001:db8:5:3:216:3eff:fe00:25 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.636/1.062/1.776 ms
```

iii) Konfigurieren Sie Ihre Router so, dass die PCs automatisch IPv6-Adressen mit Ihrem Präfix erhalten. Erläutern Sie anhand von geeigneten Aufzeichnungen den Vorgang wie

pc1 und pc2 eine globale IPv6-Adresse erhalten! Betrachten Sie auch die Kommunikation zwischen router3 und router{1,2}.

Antwort:

To observe the packets, we try to set pc1.eth1 first down, then up.

```
root@pc1:~# ip link set dev eth1 down
root@pc1:~# ip link set dev eth1 up
```

At the same time, we start the `tcpdump` command on router1.eth1 router3.eth2

```
root@router1:~# tcpdump -i eth1 -vv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture
↪  size 262144 bytes
13:47:49.372808 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
↪  ICMP6, multicast listener report v2, 1 group record(s) [gaddr
↪  ff02::1:ff00:2 to_ex { }]
13:47:49.508571 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
↪  ICMP6, multicast listener report v2, 1 group record(s) [gaddr
↪  ff02::1:ff00:2 to_ex { }]
13:47:49.980616 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) :: > ff02::1:ff00:2: [icmp6 sum ok] ICMP6, neighbor
↪  solicitation, length 32, who has fe80::216:3eff:fe00:2
          unknown option (14), length 8 (1):
            0x0000:  02ff 6e2c 516e
13:47:51.004674 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) fe80::216:3eff:fe00:2 > ff02::16: HBH (rtalert
: 0x0000) (padn) [icmp6 sum ok] ICMP6, multicast listener report v2,
↪  1 group record(s) [gaddr ff02::1:ff00:2 to_ex { }]
13:47:51.004678 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 16) fe80::216:3eff:fe00:2 > ip6-allrouters: [icm
p6 sum ok] ICMP6, router solicitation, length 16
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:02
            0x0000:  0016 3e00 0002
13:47:51.005009 IP6 (flowlabel 0xdc9fc, hlim 255, next-header ICMPv6
↪  (58) payload length: 96) fe80::216:3eff:fe80::216:3eff:fe00:2:
↪  [icmp6 sum ok] ICMP6, router advertisement, length 96
        hop limit 64, Flags [managed, other stateful], pref medium,
        ↪  router lifetime 0s, reachable time 0ms, retrans timer
        ↪  0ms
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:08
```

```
            0x0000:  0016 3e00 0008
          mtu option (5), length 8 (1):  1500
            0x0000:  0000 0000 05dc
          prefix info option (3), length 32 (4):
          ↪  2001:db8:5:101::/64, Flags [onlink, auto], valid time
          ↪  22760s, pref. time 22760s
            0x0000:  40c0 0000 58e8 0000 58e8 0000 0000 2001
            0x0010:  0db8 0005 0101 0000 0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 1800s, addr:
          ↪  2001:db8:5:101::1
            0x0000:  0000 0000 0708 2001 0db8 0005 0101 0000
            0x0010:  0000 0000 0001
          advertisement interval option (7), length 8 (1):  600000ms
            0x0000:  0000 0009 27c0
13:47:51.548561 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) fe80::216:3eff:fe00:2 > ff02::16: HBH (rtalert
: 0x0000) (padn) [icmp6 sum ok] ICMP6, multicast listener report v2,
↪  1 group record(s) [gaddr ff02::1:ff00:2 to_ex { }]
13:47:51.964602 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) :: > ff02::1:ff00:2: [icmp6 sum ok] ICMP6, n
eighbor solicitation, length 32, who has
↪  2001:db8:5:101:216:3eff:fe00:2
          unknown option (14), length 8 (1):
            0x0000:  23ec 6077 5c91
13:47:56.047874 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) fe80::216:3eff:fe00:8 > fe80::216:3eff:fe00:
2: [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has
↪  fe80::216:3eff:fe00:2
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:08
            0x0000:  0016 3e00 0008
13:47:56.048254 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 24) fe80::216:3eff:fe00:2 > fe80::216:3eff:fe00:
8: [icmp6 sum ok] ICMP6, neighbor advertisement, length 24, tgt is
↪  fe80::216:3eff:fe00:2, Flags [solicited]
13:48:01.213842 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) fe80::216:3eff:fe00:2 > fe80::216:3eff:fe00:
8: [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has
↪  fe80::216:3eff:fe00:8

root@router3:~# tcpdump -i eth2 -vv

tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture
↪  size 262144 bytes
```

```
13:52:42.837547 IP6 (flowlabel 0xd54fb, hlim 255, next-header ICMPv6
↪  (58) payload length: 120) fe80::216:3eff:fe00:19 >
 ip6-allnodes: [icmp6 sum ok] ICMP6, router advertisement, length
 ↪  120
       hop limit 64, Flags [managed, other stateful], pref medium,
        ↪  router lifetime 0s, reachable time 0ms, retrans tim
er 0ms
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:19
            0x0000:  0016 3e00 0019
          mtu option (5), length 8 (1):  1500
            0x0000:  0000 0000 05dc
          prefix info option (3), length 32 (4):
          ↪  2001:db8:5:100::/64, Flags [onlink, auto], valid time
          ↪  infinity, pref.
time infinity
            0x0000:  40c0 ffff ffff ffff ffff 0000 0000 2001
            0x0010:  0db8 0005 0100 0000 0000 0000 0000
          route info option (24), length 24 (3):  2001:db8:5::/48,
          ↪  pref=medium, lifetime=1800s
            0x0000:  3000 0000 0708 2001 0db8 0005 0000 0000
            0x0010:  0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 1800s, addr:
          ↪  2001:db8:5:100::1
            0x0000:  0000 0000 0708 2001 0db8 0005 0100 0000
            0x0010:  0000 0000 0001
          advertisement interval option (7), length 8 (1):  600000ms
            0x0000:  0000 0009 27c0


root@router2:~# tcpdump -i eth1 -vv
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture
↪  size 262144 bytes
14:26:05.632386 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
↪  ICMP6, multicast listener report v2, 1 group record(s) [gaddr
↪  ff02::1:ff00:4 to_ex { }]
14:26:05.924172 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) :: > ff02::1:ff00:4: [icmp6 sum ok] ICMP6, neighbor
↪  solicitation, length 32, who has fe80::216:3eff:fe00:4
          unknown option (14), length 8 (1):
            0x0000:  48e3 105c 3b8b
14:26:06.628206 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) :: > ff02::16: HBH (rtalert: 0x0000) (padn) [icmp6 sum ok]
↪  ICMP6, multicast listener report v2, 1 group record(s) [gaddr
↪  ff02::1:ff00:4 to_ex { }]
```

```
14:26:06.948232 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) fe80::216:3eff:fe00:4 > ff02::16: HBH (rtalert: 0x0000)
↪  (padn) [icmp6 sum ok] ICMP6, multicast listener report v2, 1
↪  group record(s) [gaddr ff02::1:ff00:4 to_ex { }]
14:26:06.948239 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 16) fe80::216:3eff:fe00:4 > ip6-allrouters: [icmp6 sum
↪  ok] ICMP6, router solicitation, length 16
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:04
            0x0000:  0016 3e00 0004
14:26:06.948873 IP6 (flowlabel 0xe4d43, hlim 255, next-header ICMPv6
↪  (58) payload length: 96) fe80::216:3eff:fe00:13 >
↪  fe80::216:3eff:fe00:4: [icmp6 sum ok] ICMP6, router
↪  advertisement, length 96
        hop limit 64, Flags [managed, other stateful], pref medium,
        ↪  router lifetime 0s, reachable time 0ms, retrans timer
        ↪  0ms
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:13
            0x0000:  0016 3e00 0013
          mtu option (5), length 8 (1):  1500
            0x0000:  0000 0000 05dc
          prefix info option (3), length 32 (4):
          ↪  2001:db8:5:201::/64, Flags [onlink, auto], valid time
          ↪  42065s, pref. time 42065s
            0x0000:  40c0 0000 a451 0000 a451 0000 0000 2001
            0x0010:  0db8 0005 0201 0000 0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 1800s, addr:
          ↪  2001:db8:5:201::1
            0x0000:  0000 0000 0708 2001 0db8 0005 0201 0000
            0x0010:  0000 0000 0001
          advertisement interval option (7), length 8 (1):  600000ms
            0x0000:  0000 0009 27c0
14:26:07.844146 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) :: > ff02::1:ff00:4: [icmp6 sum ok] ICMP6, neighbor
↪  solicitation, length 32, who has 2001:db8:5:201:216:3eff:fe00:4
↪  [1/1803]
          unknown option (14), length 8 (1):
            0x0000:  39e6 1806 11a7
14:26:07.876149 IP6 (hlim 1, next-header Options (0) payload length:
↪  36) fe80::216:3eff:fe00:4 > ff02::16: HBH (rtalert: 0x0000)
↪  (padn) [icmp6 sum ok] ICMP6, multicast listener report v2, 1
↪  group record(s) [gadd
r ff02::1:ff00:4 to_ex { }]
```

```
14:26:12.053925 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) fe80::216:3eff:fe00:13 > fe80::216:3eff:fe00:4:
↪  [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has
↪  fe80::216:3eff:fe00:4
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:13
            0x0000:  0016 3e00 0013
14:26:12.054431 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 24) fe80::216:3eff:fe00:4 > fe80::216:3eff:fe00:13:
↪  [icmp6 sum ok] ICMP6, neighbor advertisement, length 24, tgt is
↪  fe80::216:3eff:fe00:4, F
lags [solicited]
14:26:17.060155 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 32) fe80::216:3eff:fe00:4 > fe80::216:3eff:fe00:13:
↪  [icmp6 sum ok] ICMP6, neighbor solicitation, length 32, who has
↪  fe80::216:3eff:fe00:13
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:04
            0x0000:  0016 3e00 0004
14:26:17.060197 IP6 (hlim 255, next-header ICMPv6 (58) payload
↪  length: 24) fe80::216:3eff:fe00:13 > fe80::216:3eff:fe00:4:
↪  [icmp6 sum ok] ICMP6, neighbor advertisement, length 24, tgt is
↪  fe80::216:3eff:fe00:13,
Flags [router, solicited]
14:26:51.307957 IP6 (flowlabel 0xa1adc, hlim 255, next-header ICMPv6
↪  (58) payload length: 96) fe80::216:3eff:fe00:13 > ip6-allnodes:
↪  [icmp6 sum ok] ICMP6, router advertisement, length 96
        hop limit 64, Flags [managed, other stateful], pref medium,
        ↪  router lifetime 0s, reachable time 0ms, retrans timer
        ↪  0ms
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:13
            0x0000:  0016 3e00 0013
          mtu option (5), length 8 (1):  1500
            0x0000:  0000 0000 05dc
          prefix info option (3), length 32 (4):
          ↪  2001:db8:5:201::/64, Flags [onlink, auto], valid time
          ↪  42021s, pref. time 42021s
            0x0000:  40c0 0000 a425 0000 a425 0000 0000 2001
            0x0010:  0db8 0005 0201 0000 0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 1800s, addr:
          ↪  2001:db8:5:201::1
            0x0000:  0000 0000 0708 2001 0db8 0005 0201 0000
            0x0010:  0000 0000 0001
          advertisement interval option (7), length 8 (1):  600000ms
            0x0000:  0000 0009 27c0
```

```
root@router3:~# tcpdump -i eth3 -vv
tcpdump: listening on eth3, link-type EN10MB (Ethernet), capture
↪  size 262144 bytes
14:27:49.305197 IP6 (flowlabel 0x261b5, hlim 255, next-header ICMPv6
↪  (58) payload length: 120) fe80::216:3eff:fe00:20 > ip6-allnodes:
↪  [icmp6 sum ok] ICMP6, router advertisement, length 120
        hop limit 64, Flags [managed, other stateful], pref medium,
        ↪  router lifetime 0s, reachable time 0ms, retrans timer
        ↪  0ms
          source link-address option (1), length 8 (1):
          ↪  00:16:3e:00:00:20
            0x0000:  0016 3e00 0020
          mtu option (5), length 8 (1):  1500
            0x0000:  0000 0000 05dc
          prefix info option (3), length 32 (4):
          ↪  2001:db8:5:200::/64, Flags [onlink, auto], valid time
          ↪  infinity, pref. time infinity
            0x0000:  40c0 ffff ffff ffff ffff 0000 0000 2001
            0x0010:  0db8 0005 0200 0000 0000 0000 0000
          route info option (24), length 24 (3):  2001:db8:5::/48,
          ↪  pref=medium, lifetime=1800s
            0x0000:  3000 0000 0708 2001 0db8 0005 0000 0000
            0x0010:  0000 0000 0000
          rdnss option (25), length 24 (3):  lifetime 1800s, addr:
          ↪  2001:db8:5:200::1
            0x0000:  0000 0000 0708 2001 0db8 0005 0200 0000
            0x0010:  0000 0000 0001
          advertisement interval option (7), length 8 (1):  600000ms
            0x0000:  0000 0009 27c0
```

iv) Passen Sie die Routing-Konfigurationen von router1, router2 und router3 an, so dass IPv6-Pakete zwischen pc1 und pc2 vermittelt werden. Sind manuelle Änderungen daran notwendig? Zeigen Sie mittels traceroute6 (8) den Pfad von IPv6-Paketen zwischen pc1 und pc2!

Antwort:

We just need to add one route on each PC. The ping command can work.

```
root@pc1:~# ip -6 route add 2001:db8:5:201::/64 via
↪  2001:db8:5:101::1
root@pc2:~# ip -6 route add 2001:db8:5:101::/64 via
↪  2001:db8:5:201::1

root@pc1:~# ping -6 -I eth1 2001:db8:5:201:216:3eff:fe00:4
PING 2001:db8:5:201:216:3eff:fe00:4(2001:db8:5:201:216:3eff:fe00:4)
↪  from 2001:db8:5:101:216:3eff:fe00:2 eth1: 56 data bytes
```

```
64 bytes from 2001:db8:5:201:216:3eff:fe00:4: icmp_seq=1 ttl=61
↪  time=3.13 ms
64 bytes from 2001:db8:5:201:216:3eff:fe00:4: icmp_seq=2 ttl=61
↪  time=3.45 ms
64 bytes from 2001:db8:5:201:216:3eff:fe00:4: icmp_seq=3 ttl=61
↪  time=2.51 ms
64 bytes from 2001:db8:5:201:216:3eff:fe00:4: icmp_seq=4 ttl=61
↪  time=2.54 ms
64 bytes from 2001:db8:5:201:216:3eff:fe00:4: icmp_seq=5 ttl=61
↪  time=4.45 ms

--- 2001:db8:5:201:216:3eff:fe00:4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 2.513/3.217/4.451/0.712 ms
```

Now we test traceroute on pc1 and pc2.

```
root@pc1:~# traceroute6 2001:db8:5:201:216:3eff:fe00:4 -T
traceroute to 2001:db8:5:201:216:3eff:fe00:4
↪  (2001:db8:5:201:216:3eff:fe00:4), 30 hops max, 80 byte packets
 1  2001:db8:5:101::1 (2001:db8:5:101::1)  5.944 ms  5.890 ms  5.865
 ↪  ms
 2  2001:db8:5:100::1 (2001:db8:5:100::1)  5.838 ms  5.816 ms  5.792
 ↪  ms
 3  * * *
 4  2001:db8:5:201:216:3eff:fe00:4 (2001:db8:5:201:216:3eff:fe00:4)
 ↪  10.102 ms  10.077 ms  10.055 ms

root@pc2:~# traceroute6  2001:db8:5:101:216:3eff:fe00:2
traceroute to 2001:db8:5:101:216:3eff:fe00:2
↪  (2001:db8:5:101:216:3eff:fe00:2), 30 hops max, 80 byte packets
 1  2001:db8:5:201::1 (2001:db8:5:201::1)  4.072 ms  3.986 ms  3.935
 ↪  ms
 2  2001:db8:5:200::1 (2001:db8:5:200::1)  10.544 ms  10.498 ms
 ↪  10.448 ms
 3  * * *
 4  2001:db8:5:101:216:3eff:fe00:2 (2001:db8:5:101:216:3eff:fe00:2)
 ↪  17.762 ms  17.715 ms  17.666 ms
```

We found there was one record with no data (* * *). That is because router3 or router4 does not know how to reach another subnet. After we added the new route rule, we can get the full record.

```
root@router2:~# ip -6 route add  2001:db8:5:101::/64 via
↪  2001:db8:5:200::1

root@pc1:~# traceroute6 2001:db8:5:201:216:3eff:fe00:4 -T
traceroute to 2001:db8:5:201:216:3eff:fe00:4
↪  (2001:db8:5:201:216:3eff:fe00:4), 30 hops max, 80 byte packets
```

```
 1  2001:db8:5:101::1 (2001:db8:5:101::1)  5.183 ms  5.122 ms  5.097
 ↪  ms
 2  2001:db8:5:100::1 (2001:db8:5:100::1)  15.469 ms  15.446 ms
 ↪  15.423 ms
 3  2001:db8:5:200:216:3eff:fe00:15
 ↪  (2001:db8:5:200:216:3eff:fe00:15)  15.396 ms  15.373 ms  15.348
 ↪  ms
 4  2001:db8:5:201:216:3eff:fe00:4 (2001:db8:5:201:216:3eff:fe00:4)
 ↪  15.321 ms  15.298 ms  15.326 ms

root@router1:~# ip -6 route add  2001:db8:5:201::/64 via
 ↪  2001:db8:5:100::1

root@pc2:~# traceroute6  2001:db8:5:101:216:3eff:fe00:2
traceroute to 2001:db8:5:101:216:3eff:fe00:2
 ↪  (2001:db8:5:101:216:3eff:fe00:2), 30 hops max, 80 byte packets
 1  2001:db8:5:201::1 (2001:db8:5:201::1)  6.023 ms  5.945 ms  5.896
 ↪  ms
 2  2001:db8:5:200::1 (2001:db8:5:200::1)  5.848 ms  5.801 ms  5.755
 ↪  ms
 3  2001:db8:5:100:216:3eff:fe00:10
 ↪  (2001:db8:5:100:216:3eff:fe00:10)  16.467 ms  16.420 ms  16.374
 ↪  ms
 4  2001:db8:5:101:216:3eff:fe00:2 (2001:db8:5:101:216:3eff:fe00:2)
 ↪  16.327 ms  16.281 ms  16.233 ms
```

v) Konfigurieren Sie einen ip4ip6-Tunnel zwischen router1 und router2, so dass IPv4-Pakete von pc1 an pc2 (und zurück!) per IPv6 zwischen den Routern vermittelt werden! Zeigen Sie, dass Ihre Konfiguration funktioniert und IPv4-Pakete tatsächlich in IPv6 gekapselt werden.

Antwort: First we need to install `kmod-ip6-tunnel` to make router support ip4ip6 tunnel

```
root@router2:~# opkg update
root@router2:~# opkg install kmod-ip6-tunnel
root@router2:~# modprobe ip6_tunnel

root@router1:~# opkg update
root@router1:~# opkg install kmod-ip6-tunnel
root@router1:~# modprobe ip6_tunnel
```

Add the tunnel on router1 and router2. And assign the IPv4 address on tunnel and routerx.eth1.

```
root@router1:~# ip tunnel add tun0 mode ipip6 local
 ↪  2001:db8:5:100:216:3eff:fe00:10 remote
 ↪  2001:db8:5:200:216:3eff:fe00:15 ttl 255
```

```
root@router1:~# ip link set tun0 up
root@router1:~# ip a add 10.5.10.1/24 dev tun0
root@router1:~# ip a add 10.5.1.2/24 dev eth1

root@router2:~# ip tunnel add tun0 mode ipip6 local
↪  2001:db8:5:200:216:3eff:fe00:15 remote
↪  2001:db8:5:100:216:3eff:fe00:10 ttl 255
root@router2:~# ip link set tun0 up
root@router2:~# ip a add 10.5.10.2/24 dev tun0
root@router2:~# ip a add 10.5.2.2/24 dev eth1
```

Assign the IPv4 Address on pc1 and pc2.

```
root@pc1:~# ip a add 10.5.1.1/24 dev eth1
root@pc2:~# ip a add 10.5.2.1/24 dev eth1
```

Add route rules on each devices.

```
root@router1:~# ip route add 10.5.2.0/24 via 10.5.10.2
root@router2:~# ip route add 10.5.1.0/24 via 10.5.10.1
root@pc1:~# ip route add 10.5.2.0/24 via 10.5.1.2
root@pc2:~# ip route add 10.5.1.0/24 via 10.5.2.2
```

Now we test traceroute.

```
root@pc1:~# traceroute -i eth1 10.5.2.1
traceroute to 10.5.2.1 (10.5.2.1), 30 hops max, 60 byte packets
 1  10.5.1.2 (10.5.1.2)  6.862 ms  6.785 ms  6.734 ms
 2  10.5.10.2 (10.5.10.2)  6.684 ms  6.634 ms  6.582 ms
 3  10.5.2.1 (10.5.2.1)  6.529 ms  6.480 ms  6.485 ms
```

Only three hops are shown here. The packet is through the tunnel!

5 A304 Distanz-Vektor Routing mit RIP

i) Vernetzen Sie die Router wie folgt: router1–router2–router4–router3, deaktivieren Sie alle anderen Verbindungen zwischen den Routern! Vergeben Sie passende Subnetze und IP-Adressen aus Ihrem Gruppennetz 10.hGruppennummeri.0.0/16, legen Sie keine manuellen Routen an!

Antwort:

ii) Erstellen Sie die Konfigurationsdatei /etc/frr/ripd.conf! Die Syntax entnehmen Sie der Dokumentation. Hinweis: Auf Debian können Sie auch Beispielkonfigurationen in /usr/share/doc/frr/examples finden. Zeilen, die mit ! anfangen, sind Kommentare.

Antwort: First install frr, frr-ripd and frr-zebra.

```
root@router1:~# opkg install frr
root@router1:~# opkg install frr-ripd
root@router1:~# opkg install frr-zebra
```

We edit the frr.conf to make the router broadcast the routing information.

```
#/etc/frr/frr.conf
router rip
 network 10.5.0.0/16
```

iii) Passen Sie die Datei /etc/frr/daemons an und starten Sie die Dienste zebra und ripd nacheinander auf Ihren Routern! (/etc/init.d/frr start) Hinweis: starten Sie zuerst beide Dienste auf router1, dann auf router2, usw. usf.

Antwort:

iv) Beobachten Sie auf router1.eth2 die ein- und ausgehenden RIP-PDUs! Zeigen Sie die Zwischenschritte in denen router1 sein Wissen über die Infrastruktur vervollständigt!

Antwort:

```
root@router1:~# tcpdump -i eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol
↪  decode
```

```
listening on eth2, link-type EN10MB (Ethernet), capture size 262144
↪  bytes
09:46:05.622422 IP 10.5.1.1.520 > 224.0.0.9.520: RIPv2, Request,
↪  length: 24
09:46:05.623609 IP 10.5.1.2.520 > 10.5.1.1.520: RIPv2, Response,
↪  length: 64
09:46:05.631232 IP 10.5.1.1 > 224.0.0.22: igmp v3 report, 1 group
↪  record(s)
09:46:05.771243 IP 10.5.1.1 > 224.0.0.22: igmp v3 report, 1 group
↪  record(s)
09:46:06.475497 IP 10.5.1.1.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 24
09:46:09.648633 IP 10.5.1.2.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 64
09:46:10.805598 ARP, Request who-has 10.5.1.1 tell 10.5.1.2, length
↪  28
09:46:10.805622 ARP, Reply 10.5.1.1 is-at 00:16:3e:00:00:09 (oui
↪  Unknown), length 28
09:46:35.650180 IP 10.5.1.2.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 64
09:46:36.475573 IP 10.5.1.1.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 24
09:47:01.650258 IP 10.5.1.2.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 64
09:47:04.475686 IP 10.5.1.1.520 > 224.0.0.9.520: RIPv2, Response,
↪  length: 24
```

v) Machen Sie sich mit vtysh vertraut. Benutzen Sie vtysh um sich die Routing-Tabelle für router1 anzeigen zu lassen! Welche Bedeutung hat das Feld „From"?

Antwort: In the routing table output by vtysh, the `From` field indicates the IP address from which the route was learned. The meaning is as follows: Dynamic Routing Protocol Source: For dynamic routing protocols such as RIP, OSPF, etc., the From field indicates the IP address of the neighboring router, i.e., the device that sent the update for that route. Directly Connected Routes: If the route is directly connected, the From field is not displayed.

```
root@router1:~# vtysh

Hello, this is FRRouting (version 7.5).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

router1# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
```

```
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected,
        ↪  b - backup

C>* 10.5.1.0/24 is directly connected, eth2, 00:05:30
R>* 10.5.2.0/24 [120/2] via 10.5.1.2, eth2, weight 1, 00:05:29
R>* 10.5.3.0/24 [120/3] via 10.5.1.2, eth2, weight 1, 00:05:29
C>* 10.5.11.0/24 is directly connected, eth1, 00:05:30
R>* 10.5.12.0/24 [120/2] via 10.5.1.2, eth2, weight 1, 00:05:29
R>* 10.5.13.0/24 [120/4] via 10.5.1.2, eth2, weight 1, 00:05:27
C>* 192.168.0.0/24 is directly connected, eth0, 00:05:30
router1# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface

     Network            Next Hop         Metric From            Tag
      ↪  Time
C(i) 10.5.1.0/24        0.0.0.0               1 self              0
R(n) 10.5.2.0/24        10.5.1.2              2 10.5.1.2          0
↪  02:47
R(n) 10.5.3.0/24        10.5.1.2              3 10.5.1.2          0
↪  02:47
C(i) 10.5.11.0/24       0.0.0.0               1 self              0
R(n) 10.5.12.0/24       10.5.1.2              2 10.5.1.2          0
↪  02:47
R(n) 10.5.13.0/24       10.5.1.2              4 10.5.1.2          0
↪  02:47
C(r) 192.168.0.0/24     0.0.0.0               1 self              0
```

vi) Aktivieren Sie die Verbindung router1-router3 und zeigen Sie die Zwischenschritte, bis sich die Routingtabellen von router2 und router4 an den neuen Zustand angepasst haben!

Antwort: We first created the link between router1 and router3.

```
root@router1:~# ip link set eth3 up
root@router1:~# ip a add 10.5.4.1/24 eth3
root@router3:~# ip link set eth2 up
root@router3:~# ip a add 10.5.4.2/24 eth2
```

Then we used the `tcpdump` to observe the `RIPv2` packages on router2. eth2 and router4.eth3. Here are the results.

In figure 5.1, the output shows that `10.5.1.1` first sent a packet to a multicast address. The packet includes the new IP address and the hop count (metric). Router2 then updated the route table to add this entry.

In figure 5.2, the figure shows the router1.eth2 sent the packet, which included entries with a metric of 1 and 2. After that, router1.eth2 sent the full route table to router2.

In figure 5.3, we found router4 needed 3 hops to reach the 10.5.4.0/24 subnet, which means it was accessing the subnet via router2.

Next router3 sent a packet shown that router3 access 10.5.4.0/24 only need one hop in figure 5.4.

Router4 received the packet and updated the route table according to the RIP protocol. In figure 5.5, router4's route table shows that now it only needed two hops to reach the 10.5.4.0/24 subnet.

```
15:16:23.928108 IP (tos 0xc0, ttl 1, id 11485, offset 0, flags [DF], proto UDP (17), length 52)
    10.5.1.1.520 > 224.0.0.9.520: [bad udp cksum 0xeb40 -> 0x0184!]
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,         10.5.4.0/24, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 0a05 0400 ffff ff00
        0x0010:  0000 0000 0000 0001
15:16:23.928749 IP (tos 0xc0, ttl 1, id 6854, offset 0, flags [DF], proto UDP (17), length 92)
    10.5.1.2.520 > 224.0.0.9.520: [bad udp cksum 0xeb69 -> 0x2c7a!]
        RIPv2, Response, length: 64, routes: 3 or less
          AFI IPv4,         10.5.3.0/24, tag 0x0000, metric: 2, next-hop: self
          AFI IPv4,        10.5.13.0/24, tag 0x0000, metric: 3, next-hop: self
          AFI IPv4,     192.168.0.0/24, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 0a05 0300 ffff ff00
        0x0010:  0000 0000 0000 0002 0002 0000 0a05 0d00
        0x0020:  ffff ff00 0000 0000 0000 0003 0002 0000
        0x0030:  c0a8 0000 ffff ff00 0000 0000 0000 0001
15:16:44.017470 IP (tos 0xc0, ttl 1, id 9046, offset 0, flags [DF], proto UDP (17), length 132)
    10.5.1.2.520 > 224.0.0.9.520: [bad udp cksum 0xeb91 -> 0x0c18!]
        RIPv2, Response, length: 104, routes: 5 or less
          AFI IPv4,         10.5.2.0/24, tag 0x0000, metric: 1, next-hop: self
          AFI IPv4,         10.5.3.0/24, tag 0x0000, metric: 2, next-hop: self
          AFI IPv4,        10.5.12.0/24, tag 0x0000, metric: 1, next-hop: self
          AFI IPv4,        10.5.13.0/24, tag 0x0000, metric: 3, next-hop: self
          AFI IPv4,     192.168.0.0/24, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 0a05 0200 ffff ff00
        0x0010:  0000 0000 0000 0001 0002 0000 0a05 0300
        0x0020:  ffff ff00 0000 0000 0000 0002 0002 0000
```

Figure 5.1: A304 vi) `tcpdump` Output

vii) Deaktivieren Sie router1.eth1, um einen Verbindungsausfall zu simulieren! Beobachten Sie das Verhalten der Router!

a) Welches Verhalten der Router ist nach [RFC 1058] zu erwarten, um das über router1.eth1 erreichbare Subnetz aus den Routing-Tabellen zu entfernen?

Antwort:

According to [RFC 1058] (Routing Information Protocol, the original version of RIP), if router1.eth1 is disabled, simulating a connection loss, the RIP protocol performs the following steps to gradually remove subnets reachable through router1.eth1 from routing Removed from table:

When RIP detects that interface eth1 is disabled, the router will mark the subnet through router1.eth1 as invalid. RIP does not delete these routes immediately, but sets their metric to 16 (an infinite value in RIP, indicating unreachability). The router will continue to broadcast these route entries to other routers to notify devices on the network that the subnet is unreachable.

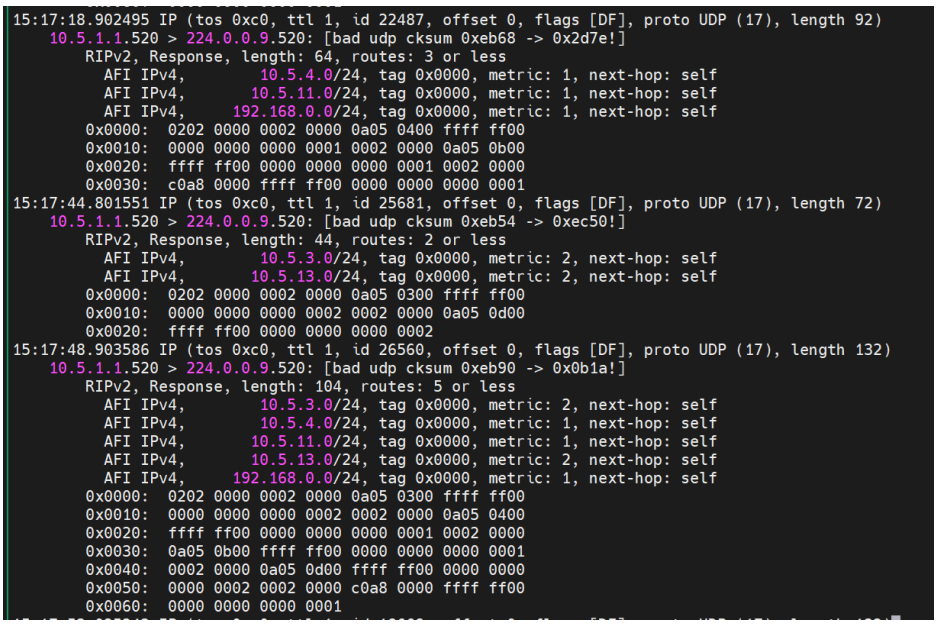

Figure 5.2: A304 vi) `tcpdump` Output

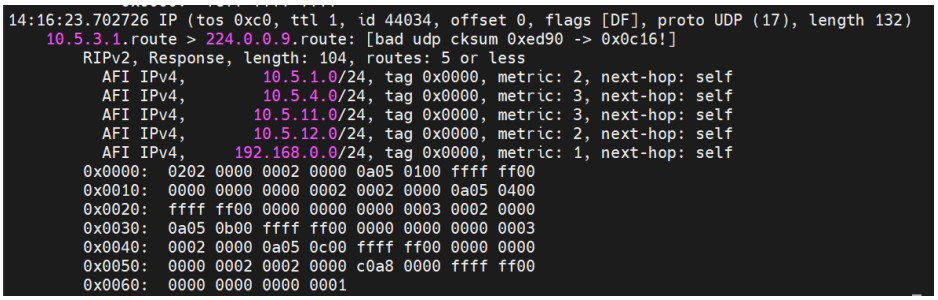

Figure 5.3: A304 vi) `tcpdump` Output

After marking a route as invalid, the RIP starts the Route Garbage Collection Timer with a default value of 120 seconds. Until the timer times out, the router continues to broadcast information with a metric value of 16 for that subnet into the network. If the route is still unreachable after the timer times out, the route is completely removed from the routing table.

After other routers running RIP receive a triggered update from router1, other routers update the subnet metric to 16 and marks it as invalid. Also they start theirs own route aging timer. Broadcasts the updated routing table again to other neighboring routers to ensure routing consistency across the network.

b) Welches Verhalten der Router ist tatsächlich zu beobachten? Welche Technik wird eingesetzt um das nun nicht mehr erreichbare Subnetz aus den Routing-Tabellen zu entfernen? Zeigen Sie die entsprechenden PDUs der Router!

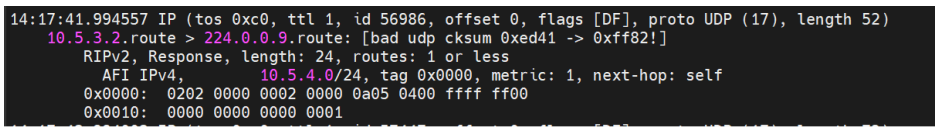

Figure 5.4: A304 vi) `tcpdump` Output

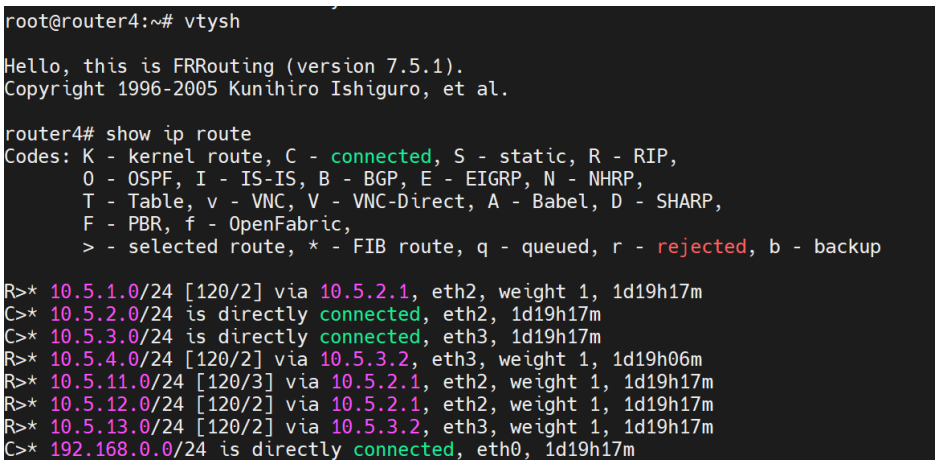

Figure 5.5: A304 vi) `vtysh` Output

Antwort:

First, we observed that router1 sent a packet, where the metric of 10.5.11.0/24 is 16. You can find the result in figure 5.6.

The next rip response packets sent by router1 all contain records for 10.5.11.0/24 with a metric of 16. You can find it in figure 5.7

After about two minutes, the rip response packet of router1 did not include 10.5.11.0/24 entry. The output is shown in figure 5.8.

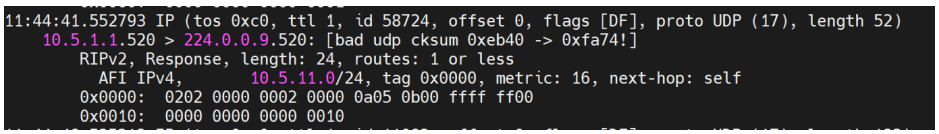

Figure 5.6: A304 vii) `tcpdump` Output

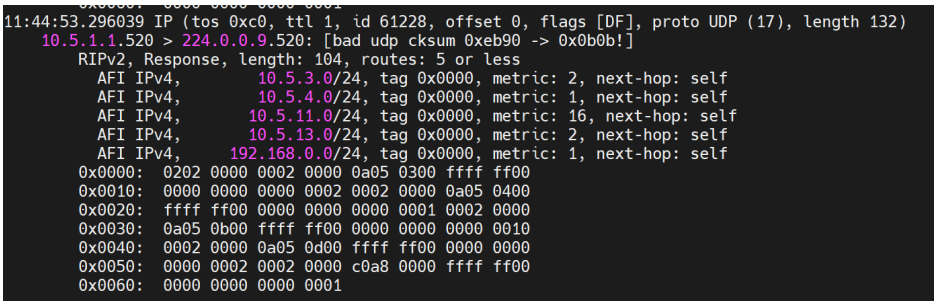

Figure 5.7: A304 vii) `tcpdump` Output

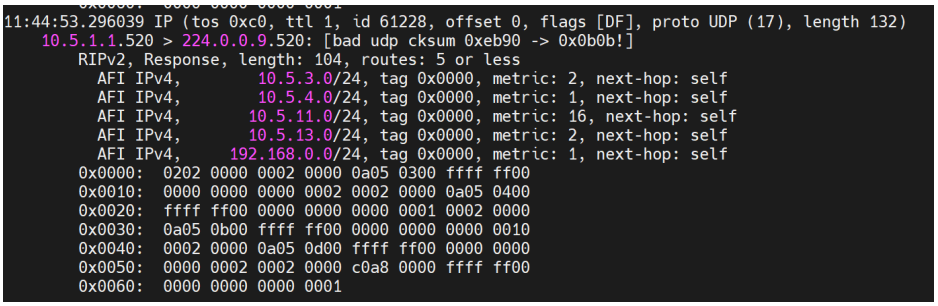

Figure 5.8: A304 vii) `tcpdump` Output

6 A305 Link-State Routing mit OSPF

i) Stellen Sie den Zustand aus A304, i) wieder her. Deaktivieren Sie FRR auf allen Routern!

Antwort:

ii) Erstellen Sie die Konfigurationsdatei /etc/frr/ospfd.conf! Die Syntax entnehmen Sie der Dokumentation

Antwort: router1

```
router ospf
 network 10.5.1.0/24 area 0
 network 10.5.11.0/24 area 1

interface eth1
 ip address 10.5.11.1/24
 ip ospf area 1

interface eth2
 ip address 10.5.1.1/24
 ip ospf area 0
```

router2

```
router ospf
 network 10.5.1.0/24 area 0
 network 10.5.2.0/24 area 0
 network 10.5.12.0/24 area 2

interface eth1
 ip address 10.5.12.1/24
 ip ospf area 2

interface eth2
 ip address 10.5.1.2/24
 ip ospf area 0
```

```
interface eth4
 ip address 10.5.2.1/24
 ip ospf area 0
```

router3

```
router ospf
 network 10.5.3.0/24 area 0
 network 10.5.13.0/24 area 3
interface eth1
 ip address 10.5.13.1/24
 ip ospf area 3

interface eth4
 ip address 10.5.3.2/24
 ip ospf area 0
```

router4

```
router ospf
 network 10.5.3.0/24 area 0
 network 10.5.2.0/24 area 0
interface eth2
 ip address 10.5.2.2/24
 ip ospf area 0
interface eth3
 ip address 10.5.3.1/24
 ip ospf area 0
```

iii) Passen Sie die Datei /etc/frr/daemons an und starten Sie zebra und ospfd nacheinander! Hinweis: zuerst beide Dienste auf router1, dann router2, usw. usf.

Antwort:

```
root@router1:~# /etc/init.d/frr restart
Stopped watchfrr
Stopped ospfd
Stopped staticd
Stopped zebra
Started watchfrr
root@router2:~# /etc/init.d/frr restart
Stopped watchfrr
Stopped ospfd
```

```
Stopped staticd
Stopped zebra
Started watchfrr
root@router3:~# /etc/init.d/frr restart
Stopped watchfrr
Stopped ospfd
Stopped staticd
Stopped zebra
Started watchfrr
root@router4:~# /etc/init.d/frr restart
Restarting frr (via systemctl): frr.service.
```

iv) Beobachten Sie die ein- und ausgehenden OSPF-PDUs auf router1.eth2! Beschreiben Sie die Zwischenschritte, in denen router1 sein Wissen über die Infrastruktur vervollständigt!

Antwort: We observed these packets: first is a hello packet from router2. Hello packet is used to discover and maintain neighbor relationships. You can find router-id, neighbor list in the packet.

```
15:53:19.631291 IP (tos 0xc0, ttl 1, id 42861, offset 0, flags
↪  [none], proto OSPF (89), length 68)
    10.5.1.2 > 224.0.0.5: OSPFv2, Hello, length 48
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.0,
          ↪  Priority 1
          Designated Router 10.5.1.1, Backup Designated Router
          ↪  10.5.1.2
          Neighbor List:
            192.168.0.7
```

Router1 sent a hello packet as well:

```
15:53:20.122306 IP (tos 0xc0, ttl 1, id 42849, offset 0, flags
↪  [none], proto OSPF (89), length 64)
    10.5.1.1 > 224.0.0.5: OSPFv2, Hello, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External]
          Hello Timer 10s, Dead Timer 40s, Mask 255.255.255.0,
          ↪  Priority 1
```

After that, router1 and router2 exchange the database description packet:

```
15:53:29.632078 IP (tos 0xc0, ttl 1, id 42851, offset 0, flags
↪  [none], proto OSPF (89), length 52)
    10.5.1.1 > 10.5.1.2: OSPFv2, Database Description, length 32
```

```
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External], DD Flags [Init, More, Master], MTU:
        ↪  1500, Sequence: 0x17489188
15:53:29.635309 IP (tos 0xc0, ttl 1, id 42867, offset 0, flags
↪  [none], proto OSPF (89), length 52)
    10.5.1.2 > 10.5.1.1: OSPFv2, Database Description, length 32
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External], DD Flags [Init, More, Master], MTU:
        ↪  1500, Sequence: 0x44b1afd8
15:53:29.635617 IP (tos 0xc0, ttl 1, id 42852, offset 0, flags
↪  [none], proto OSPF (89), length 92)
    10.5.1.1 > 10.5.1.2: OSPFv2, Database Description, length 72
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External], DD Flags [none], MTU: 1500, Sequence:
        ↪  0x44b1afd8
          Advertising Router 192.168.0.7, seq 0x80000003, age 0s,
          ↪  length 16
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
          Advertising Router 192.168.0.7, seq 0x80000001, age 9s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External]
15:53:29.640670 IP (tos 0xc0, ttl 1, id 42870, offset 0, flags
↪  [none], proto OSPF (89), length 132)
    10.5.1.2 > 10.5.1.1: OSPFv2, Database Description, length 112
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External], DD Flags [Master], MTU: 1500, Sequence:
        ↪  0x44b1afd9
          Advertising Router 192.168.0.12, seq 0x8000000d, age 9s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
          Advertising Router 192.168.0.22, seq 0x8000000a, age 15s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000002, age 19s,
          ↪  length 12
            Network LSA (2), LSA-ID: 10.5.2.1
            Options: [External]
```

```
          Advertising Router 192.168.0.12, seq 0x80000002, age 46s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.12.0
            Options: [External]
```

Router1 and its neighbour exchange summaries of the link-state database through DBD messages. These messages help Router1 understand the basic topology of the network.

With this packet, Router1 gets the following information:

Router1 knows that 192.168.0.12(router2) and 192.168.0.22(router4) are its neighbors;

Router1 knows what LSA information is in the LSDB maintained by neighbours 192.168.0.12(router2) and 192.168.0.22(router4), including 10.5.2.1 is a multicast network, announced by 192.168.0.12(router2). 10.5.12.0(the subnet of pc2 and router2) is a summarized route.

Router1 knows that these LSAs are from the backbone area (Area 0) and that the network contains at least two routers (192.168.0.12 and 192.168.0.22) and one network prefix 10.5.2.1.

Next, router1 sent an LS-Request packet to acquire additional information.

```
15:53:29.640899 IP (tos 0xc0, ttl 1, id 42853, offset 0, flags
↪  [none], proto OSPF (89), length 52)
    10.5.1.1 > 10.5.1.2: OSPFv2, Database Description, length 32
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0)
        Options [External], DD Flags [none], MTU: 1500, Sequence:
        ↪  0x44b1afd9
15:53:29.640965 IP (tos 0xc0, ttl 1, id 42854, offset 0, flags
↪  [none], proto OSPF (89), length 92)
    10.5.1.1 > 10.5.1.2: OSPFv2, LS-Request, length 72
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0)
          Advertising Router: 192.168.0.12, Router LSA (1), LSA-ID:
          ↪  192.168.0.12
          Advertising Router: 192.168.0.22, Router LSA (1), LSA-ID:
          ↪  192.168.0.22
          Advertising Router: 192.168.0.12, Network LSA (2), LSA-ID:
          ↪  10.5.2.1
          Advertising Router: 192.168.0.12, Summary LSA (3), LSA-ID:
          ↪  10.5.12.0
```

Here is the reply from router2:

```
15:53:29.642497 IP (tos 0xc0, ttl 1, id 42872, offset 0, flags
↪  [none], proto OSPF (89), length 284)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Update, length 264
```

```
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪ none (0), 6 LSAs
          LSA #1
          Advertising Router 192.168.0.12, seq 0x8000000d, age 10s,
          ↪ length 28
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [ABR]
              Stub Network: 10.5.1.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.2.1, Interface Address:
              ↪ 10.5.2.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0100 ffff ff00 0300 000a
            0x0010:  0a05 0201 0a05 0201 0200 000a
          LSA #2
          Advertising Router 192.168.0.22, seq 0x8000000a, age 17s,
          ↪ length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.5.3.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.2.1, Interface Address:
              ↪ 10.5.2.2
                topology default (0), metric 10
            0x0000:  0000 0002 0a05 0300 ffff ff00 0300 000a
            0x0010:  0a05 0201 0a05 0202 0200 000a
          LSA #3
          Advertising Router 192.168.0.12, seq 0x80000002, age 20s,
          ↪ length 12
            Network LSA (2), LSA-ID: 10.5.2.1
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.12
              192.168.0.22
            0x0000:  ffff ff00 c0a8 000c c0a8 0016
          LSA #4
          Advertising Router 192.168.0.12, seq 0x80000002, age 47s,
          ↪ length 8
            Summary LSA (3), LSA-ID: 10.5.12.0
            Options: [External]
            Mask 255.255.255.0
                topology default (0) metric 10
            0x0000:  ffff ff00 0000 000a
```

```
          LSA #5
          Advertising Router 192.168.0.12, seq 0x8000000e, age 1s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.2
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.2.1, Interface Address:
              ↪  10.5.2.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0102 0a05 0102 0200 000a
            0x0010:  0a05 0201 0a05 0201 0200 000a
          LSA #6
          Advertising Router 192.168.0.12, seq 0x80000002, age 1s,
          ↪  length 12
            Network LSA (2), LSA-ID: 10.5.1.2
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.7
              192.168.0.12
            0x0000:  ffff ff00 c0a8 0007 c0a8 000c
```

Router1 then sent a LS-Update packet to the broadcast address.

```
15:53:29.643483 IP (tos 0xc0, ttl 1, id 42855, offset 0, flags
↪  [none], proto OSPF (89), length 84)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Update, length 64
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000004, age 1s,
          ↪  length 16
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0001 0a05 0102 0a05 0101 0200 000a
```

Router2 received the LS-Update packet, and sent it to the broadcast address as well.

```
15:53:29.643955 IP (tos 0xc0, ttl 1, id 42874, offset 0, flags
↪  [none], proto OSPF (89), length 84)
```

```
    10.5.1.2 > 10.5.1.1: OSPFv2, LS-Update, length 64
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪ none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000007, age 3600s,
          ↪ length 16
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.1.1, Interface Address:
              ↪ 10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0001 0a05 0101 0a05 0101 0200 000a
```

When Router1 receives an LSU packet from a neighbor, it parses the LSAs in the LSU packet line by line. if it confirms that these LSAs have been received correctly, it constructs an LSAck packet for response to the neighbor.

So after router2 sent these LSU packets, router1 sent a LS-Ack packet.

```
15:53:30.631916 IP (tos 0xc0, ttl 1, id 42858, offset 0, flags
↪ [none], proto OSPF (89), length 144)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Ack, length 124
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪ none (0)
          Advertising Router 192.168.0.12, seq 0x8000000d, age 10s,
          ↪ length 28
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
          Advertising Router 192.168.0.22, seq 0x8000000a, age 17s,
          ↪ length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000002, age 20s,
          ↪ length 12
            Network LSA (2), LSA-ID: 10.5.2.1
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000002, age 47s,
          ↪ length 8
            Summary LSA (3), LSA-ID: 10.5.12.0
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000002, age 1s,
          ↪ length 12
            Network LSA (2), LSA-ID: 10.5.1.2
            Options: [External]
```

After that, router2 sent more information to router1 in a LS-Update packet. This

packet includes the 10.5.3.0/24, 10.5.13.0/24. The route information are completed.

```
15:53:34.638512 IP (tos 0xc0, ttl 1, id 42875, offset 0, flags
↪  [none], proto OSPF (89), length 272)
    10.5.1.2 > 10.5.1.1: OSPFv2, LS-Update, length 252
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0), 7 LSAs
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000007, age 3600s,
          ↪  length 16
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.1.1, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0001 0a05 0101 0a05 0101 0200 000a
          LSA #2
          Advertising Router 192.168.0.17, seq 0x80000004, age
          ↪  3600s, length 16
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.3.1, Interface Address:
              ↪  10.5.3.2
                topology default (0), metric 10
            0x0000:  0100 0001 0a05 0301 0a05 0302 0200 000a
          LSA #3
          Advertising Router 192.168.0.7, seq 0x80000002, age 3600s,
          ↪  length 12
            Network LSA (2), LSA-ID: 10.5.1.1
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.7
              192.168.0.12
            0x0000:  ffff ff00 c0a8 0007 c0a8 000c
          LSA #4
          Advertising Router 192.168.0.22, seq 0x80000002, age
          ↪  3600s, length 12
            Network LSA (2), LSA-ID: 10.5.2.2
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.12
```

```
    192.168.0.22
  0x0000:  ffff ff00 c0a8 000c c0a8 0016
LSA #5
Advertising Router 192.168.0.22, seq 0x80000001, age
↪  3600s, length 12
  Network LSA (2), LSA-ID: 10.5.3.1
  Options: [External]
  Mask 255.255.255.0
  Connected Routers:
    192.168.0.22
    192.168.0.17
  0x0000:  ffff ff00 c0a8 0016 c0a8 0011
LSA #6
Advertising Router 192.168.0.7, seq 0x80000001, age 3600s,
↪  length 8
  Summary LSA (3), LSA-ID: 10.5.11.0
  Options: [External]
  Mask 255.255.255.0
      topology default (0) metric 10
  0x0000:  ffff ff00 0000 000a
LSA #7
Advertising Router 192.168.0.17, seq 0x80000001, age
↪  3600s, length 8
  Summary LSA (3), LSA-ID: 10.5.13.0
  Options: [External]
  Mask 255.255.255.0
      topology default (0) metric 10
  0x0000:  ffff ff00 0000 000a
```

v) Aktivieren Sie die Verbindung router1-router3. Beschreiben Sie die Zwischenschritte, bis sich die Routingtabellen von router2 und router4 an den neuen Zustand angepasst haben! Belegen Sie Ihre Erklärung mit entsprechenden Programmausgaben!

Antwort:

First, we turn the two eth on and assign the ip addresses:

```
root@router1:~# ip link set eth3 up
root@router1:~# ip a add 10.5.4.1/24 dev eth3
root@router3:~# ip link set eth2 up
root@router3:~# ip a add 10.5.4.2/24 dev eth2
```

We used tcpdump on router2.eth2 and router4.eth3.

We observed this packet on router2.eth2. This LS-Update packet is shown that there is one stub network 10.5.4.0, the metric is 10.

```
23:17:11.860663 IP (tos 0xc0, ttl 1, id 24812, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000005, age 1s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Stub Network: 10.5.4.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0400 ffff ff00 0300 000a
            0x0010:  0a05 0102 0a05 0101 0200 000a
```

After about one miniute, router1 sent a LS-Update packet. This packet gave the 10.5.4.2 neighbor Network-ID.

```
23:18:04.214278 IP (tos 0xc0, ttl 1, id 24847, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x8000004d, age 1s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.4.2, Interface Address:
              ↪  10.5.4.1
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0402 0a05 0401 0200 000a
            0x0010:  0a05 0102 0a05 0101 0200 000a
```

Router2 then sent an LS-Update packet as well. This packet shows that the generator of the LSA for this network is router3(192.168.0.17). LSA ID 10.5.4.2 indicates the network ID of this multi-access network. 192.168.0.17 is connected to 192.168.0.7 and 192.168.0.17.

```
23:18:04.217724 IP (tos 0xc0, ttl 1, id 12565, offset 0, flags
↪  [none], proto OSPF (89), length 80)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Update, length 60
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.17, seq 0x80000001, age 3s,
          ↪  length 12
            Network LSA (2), LSA-ID: 10.5.4.2
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.7
              192.168.0.17
            0x0000:  ffff ff00 c0a8 0007 c0a8 0011
```

Then router1 sent a similar LS-Update packet, and router2 sent an LS Ack packet to reply.

```
23:18:04.218473 IP (tos 0xc0, ttl 1, id 24848, offset 0, flags
↪  [none], proto OSPF (89), length 80)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Update, length 60
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.17, seq 0x80000001, age 2s,
          ↪  length 12
            Network LSA (2), LSA-ID: 10.5.4.2
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.7
              192.168.0.17
            0x0000:  ffff ff00 c0a8 0007 c0a8 0011
23:18:05.119894 IP (tos 0xc0, ttl 1, id 12566, offset 0, flags
↪  [none], proto OSPF (89), length 64)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
          Advertising Router 192.168.0.7, seq 0x8000004c, age 1s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
```

Router1 replied to router2's LS update packet as well. We do not show there since they are basically same.

Router1 sent LS Update again to router2 rather than the broadcast address.

```
23:18:11.857370 IP (tos 0xc0, ttl 1, id 24854, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.1.1 > 10.5.1.2: OSPFv2, LS-Update, length 76
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x8000004d, age 8s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.4.2, Interface Address:
              ↪  10.5.4.1
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0402 0a05 0401 0200 000a
            0x0010:  0a05 0102 0a05 0101 0200 000a
```

Router2 replied router1's packet and sent the LS-Update as well.

```
23:18:12.121667 IP (tos 0xc0, ttl 1, id 12572, offset 0, flags
↪  [none], proto OSPF (89), length 64)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
          Advertising Router 192.168.0.7, seq 0x8000004d, age 8s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
23:18:12.916478 IP (tos 0xc0, ttl 1, id 12573, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.17, seq 0x80000049, age 11s,
          ↪  length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.3.1, Interface Address:
              ↪  10.5.3.2
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.4.2, Interface Address:
              ↪  10.5.4.2
```

```
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0301 0a05 0302 0200 000a
            0x0010:  0a05 0402 0a05 0402 0200 000a
```

Now router2 gets the idea of how to access 10.5.4.0/24 subnets.

On router4, it first sent an LS-Update packet, representing that it received the change from router1 through router2.

```
22:17:16.810216 IP (tos 0xc0, ttl 1, id 20355, offset 0, flags
↪  [none], proto OSPF (89), length 124)
    10.5.3.1 > 224.0.0.5: OSPFv2, LS-Update, length 104
        Router-ID router4, Backbone Area, Authentication Type: none
        ↪  (0), 2 LSAs
          LSA #1
          Advertising Router router1, seq 0x8000004a, age 3s, length
          ↪  28
            Router LSA (1), LSA-ID: router1
            Options: [External]
            Router LSA Options: [ABR]
              Stub Network: 10.5.4.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪  10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0400 ffff ff00 0300 000a
            0x0010:  0a05 0102 0a05 0101 0200 000a
          LSA #2
          Advertising Router router1, seq 0x80000045, age 3s, length
          ↪  8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External]
            Mask 255.255.255.0
                topology default (0) metric 10
            0x0000:  ffff ff00 0000 000a
```

After about ten seconds, router3 sent the LS-Update packet includes the 10.5.4.0 Stub Network.

```
22:17:27.869555 IP (tos 0xc0, ttl 1, id 24828, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.3.2 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID router3, Backbone Area, Authentication Type: none
        ↪  (0), 1 LSA
          LSA #1
          Advertising Router router3, seq 0x80000005, age 1s, length
          ↪  28
            Router LSA (1), LSA-ID: router3
```

```
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.3.1, Interface Address:
              ↪  10.5.3.2
                topology default (0), metric 10
              Stub Network: 10.5.4.0, Mask: 255.255.255.0
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0301 0a05 0302 0200 000a
            0x0010:  0a05 0400 ffff ff00 0300 000a
```

But router4 did not reply this packet. Later router3 sent the same content in a packet again, this time router4 replied.

```
22:17:32.866828 IP (tos 0xc0, ttl 1, id 24830, offset 0, flags
↪  [none], proto OSPF (89), length 96)
    10.5.3.2 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID router3, Backbone Area, Authentication Type: none
        ↪  (0), 1 LSA
          LSA #1
          Advertising Router router3, seq 0x80000047, age 1s, length
          ↪  28
            Router LSA (1), LSA-ID: router3
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.3.1, Interface Address:
              ↪  10.5.3.2
                topology default (0), metric 10
              Stub Network: 10.5.4.0, Mask: 255.255.255.0
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0301 0a05 0302 0200 000a
            0x0010:  0a05 0400 ffff ff00 0300 000a
22:17:32.866830 IP (tos 0xc0, ttl 1, id 24831, offset 0, flags
↪  [none], proto OSPF (89), length 76)
    10.5.3.2 > 224.0.0.5: OSPFv2, LS-Update, length 56
        Router-ID router3, Backbone Area, Authentication Type: none
        ↪  (0), 1 LSA
          LSA #1
          Advertising Router router3, seq 0x80000044, age 1s, length
          ↪  8
            Summary LSA (3), LSA-ID: 10.5.13.0
            Options: [External]
            Mask 255.255.255.0
                topology default (0) metric 10
            0x0000:  ffff ff00 0000 000a
22:17:33.079749 IP (tos 0xc0, ttl 1, id 20372, offset 0, flags
↪  [none], proto OSPF (89), length 84)
```

```
    10.5.3.1 > 224.0.0.5: OSPFv2, LS-Ack, length 64
        Router-ID router4, Backbone Area, Authentication Type: none
        ↪ (0)
          Advertising Router router3, seq 0x80000047, age 1s, length
          ↪ 28
            Router LSA (1), LSA-ID: router3
            Options: [External]
          Advertising Router router3, seq 0x80000044, age 1s, length
          ↪ 8
            Summary LSA (3), LSA-ID: 10.5.13.0
            Options: [External] [|ospf2]
```

Later, router3 sent the neighbour LS-Update packet, including 10.5.4.2. Router4 replied it.

```
22:18:11.809462 IP (tos 0xc0, ttl 1, id 20389, offset 0, flags
↪ [none], proto OSPF (89), length 96)
    10.5.3.1 > 224.0.0.5: OSPFv2, LS-Update, length 76
        Router-ID router4, Backbone Area, Authentication Type: none
        ↪ (0), 1 LSA
          LSA #1
          Advertising Router router1, seq 0x8000004d, age 10s,
          ↪ length 28
            Router LSA (1), LSA-ID: router1
            Options: [External]
            Router LSA Options: [ABR]
              Neighbor Network-ID: 10.5.4.2, Interface Address:
              ↪ 10.5.4.1
                topology default (0), metric 10
              Neighbor Network-ID: 10.5.1.2, Interface Address:
              ↪ 10.5.1.1
                topology default (0), metric 10
            0x0000:  0100 0002 0a05 0402 0a05 0401 0200 000a
            0x0010:  0a05 0102 0a05 0101 0200 000a
22:18:11.868167 IP (tos 0xc0, ttl 1, id 24857, offset 0, flags
↪ [none], proto OSPF (89), length 64)
    10.5.3.2 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID router3, Backbone Area, Authentication Type: none
        ↪ (0)
          Advertising Router router1, seq 0x8000004d, age 10s,
          ↪ length 28
            Router LSA (1), LSA-ID: router1
            Options: [External] [|ospf2]
```

Now, router4 has the idea of accessing the 10.5.4.0/24 subnet through router3 rather than router2.

vi) Deaktivieren Sie router1.eth1, um einen Verbindungsausfall zu simulieren! Erläutern Sie den Unterschied zwischen RIP und OSPF in der Art und Weise wie der Verbindungsaus-

fall an andere Router propagiert wird!

Antwort:

For RIP protocol, when router1.eth1 is disabled and RIP detects that the directly connected subnet is no longer available, it immediately triggers an update (Triggered Update).

The update sets the subnet's hop count (Metric) to 16, indicating that the destination is unreachable.

The RIP sends the update to neighboring routers via multicast.

For the OSPF process, router1 detects that eth1 is disabled and immediately generates a new LSA (link state update).

The new LSA is sent via multicast to the neighbor router, which immediately updates the LSDB and runs SPF to recalculate the route.

The neighboring router further propagates the LSA to other routers and synchronises the state across the network. The entire network can quickly learn about the state change of router1.eth1 and update the routes.

Here after we turn router1.eth1 down, we used tcpdump to observe router2.eth2 and router4.et3.

On router2.eth2, we noticed that router1 sent the packet that 10.5.11.0's age changed to 3600s, which indicated that the LSA is unreachable.

```
22:24:11.120988 IP (tos 0xc0, ttl 1, id 62530, offset 0, flags
↪  [none], proto OSPF (89), length 76)
    10.5.1.1 > 224.0.0.5: OSPFv2, LS-Update, length 56
        Router-ID 192.168.0.7, Backbone Area, Authentication Type:
        ↪  none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000001, age 3600s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External]
            Mask 255.255.255.0
                topology default (0) metric 10
            0x0000:  ffff ff00 0000 000a
22:24:11.233966 IP (tos 0xc0, ttl 1, id 29853, offset 0, flags
↪  [none], proto OSPF (89), length 64)
    10.5.1.2 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.12, Backbone Area, Authentication Type:
        ↪  none (0)
          Advertising Router 192.168.0.7, seq 0x80000001, age 3600s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External]
```

At the same time, on the router.eth3, we observed that router3 sent the packet, which included the same content as the above packet.

This demonstrates that when a network link state changes, the OSPF protocol immediately forwards the LS-Update packet without performing any computations. This behavior contrasts with the RIP protocol, where the router must first compute the routing table and then broadcast the updated table to its neighbors.

```
21:24:11.093604 IP (tos 0xc0, ttl 1, id 63209, offset 0, flags
↪  [none], proto OSPF (89), length 76)
    10.5.3.2 > 224.0.0.5: OSPFv2, LS-Update, length 56
        Router-ID router3, Backbone Area, Authentication Type: none
        ↪  (0), 1 LSA
          LSA #1
          Advertising Router router1, seq 0x80000001, age 3600s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External]
            Mask 255.255.255.0
                topology default (0) metric 10
            0x0000:  ffff ff00 0000 000a
21:24:11.844794 IP (tos 0xc0, ttl 1, id 9514, offset 0, flags
↪  [none], proto OSPF (89), length 64)
    10.5.3.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID router4, Backbone Area, Authentication Type: none
        ↪  (0)
          Advertising Router router1, seq 0x80000001, age 3600s,
          ↪  length 8
            Summary LSA (3), LSA-ID: 10.5.11.0
            Options: [External] [|ospf2]
```

7 A306 Autonome Systeme und BGP

i) Stellen Sie sicher, dass Sie Adressen aus Ihrem Subnet (10.hGruppei.0.0/16) verwenden. Ihre Infrastruktur bildet das Autonome System mit der Nummer 65000 + hGruppei.

Antwort:

Since we are using the configuration from A305, all of ip address are under 10.5.0.0/16.

ii) Für diese Aufgabe werden vom Lehrstuhl zwei weitere Maschinen bereitgestellt, Gruppe11 (Odd, AS 65011) und Gruppe12 (Even, AS 65012). Auf diesen Maschinen ist BGP bereits konfiguriert. Abhängig von ihrer Gruppennummer können Sie allerdings nur mit einer der Maschinen direkt kommunizieren. Ihr Ziel ist es mittels BGP eine Route in das Subnet der anderen Maschine zu bekommen.

Stellen Sie mit Hilfe eines GRE-Tunnels eine Verbindung zu der Maschine her. Verwenden Sie als Endpunkte die Adresse von router1 an eth0 und die globale IP-Adresse der Zielmaschine. Achten Sie darauf, dass eine TTL > 3 verwendet wird. Hinweis: Sie können auf die selbe Weise eine Verbindung zu anderen Gruppen herstellen.

Antwort: We get the ip address from management node:

```
rnp@Gruppe05:~$ ping Gruppe11
PING Gruppe11.rnp.lab.nm.ifi.lmu.de (10.153.211.198) 56(84) bytes of
↪  data.
64 bytes from Gruppe11.rnp.lab.nm.ifi.lmu.de (10.153.211.198):
↪  icmp_seq=1 ttl=64 time=0.215 ms
64 bytes from Gruppe11.rnp.lab.nm.ifi.lmu.de (10.153.211.198):
↪  icmp_seq=2 ttl=64 time=0.323 ms
^C
--- Gruppe11.rnp.lab.nm.ifi.lmu.de ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.215/0.269/0.323/0.054 ms
```

Then we add the route rules on router1 to access 10.153.211.198 since we cannot ping this ip address directly.

```
root@router1:~# ping Gruppe11
ping: bad address 'Gruppe11'
root@router1:~# ip route add 10.153.211.196/32 dev eth0
root@router1:~# ip route add 10.153.211.0/24 via 10.153.211.196 dev
↪  eth0
root@router1:~# ping 10.153.211.198 -I eth0
PING 10.153.211.198 (10.153.211.198): 56 data bytes
```

```
64 bytes from 10.153.211.198: seq=0 ttl=63 time=0.611 ms
64 bytes from 10.153.211.198: seq=1 ttl=63 time=0.583 ms
...
--- 10.153.211.198 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max = 0.458/0.560/0.611 ms
```

Next we added the tunnel and assigned the IP address.

```
root@router1:~# ip tunnel add gre1 mode gre local 192.168.0.7 remote
↪  10.153.211.198 ttl 64
root@router1:~# ip link set gre1 up
root@router1:~# ip a add 10.5.0.11/32 dev gre1
```

Add the route rule to 10.11.0.5/32

```
root@router1:~# ip route add 10.11.0.5/32 dev gre1
root@router1:~# ping 10.11.0.5
PING 10.11.0.5 (10.11.0.5): 56 data bytes
64 bytes from 10.11.0.5: seq=0 ttl=64 time=0.849 ms
64 bytes from 10.11.0.5: seq=1 ttl=64 time=0.904 ms
64 bytes from 10.11.0.5: seq=2 ttl=64 time=0.704 ms
64 bytes from 10.11.0.5: seq=3 ttl=64 time=0.718 ms
64 bytes from 10.11.0.5: seq=4 ttl=64 time=0.706 ms
^C
--- 10.11.0.5 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.704/0.776/0.904 ms
```

iii) Konfigurieren Sie Router1 als Border Router Ihres Autonomen Systems. Vermitteln Sie mit dem Border Router am anderen Ende des Tunnels. Dieser hat die IP-Adresse 10.11.0.hIhre Gruppeii bzw. 10.12.0.hIhre Gruppeii. Hinweis: Definieren Sie Host-Routen.

Antwort:

After we added the tunnel, we added the host route.

```
root@router1:~# ip route add 10.11.0.5/32 dev gre1
```

Then we restarted the frr service.

```
root@router1:~# /etc/init.d/frr restart
Stopped watchfrr
Stopped ospfd
Stopped staticd
Stopped zebra
Stopped bgpd
Started watchfrr
```

Now let's check the bgp summary. The output of `vtysh -c "show bgp summary"` are shown in figure 7.1.

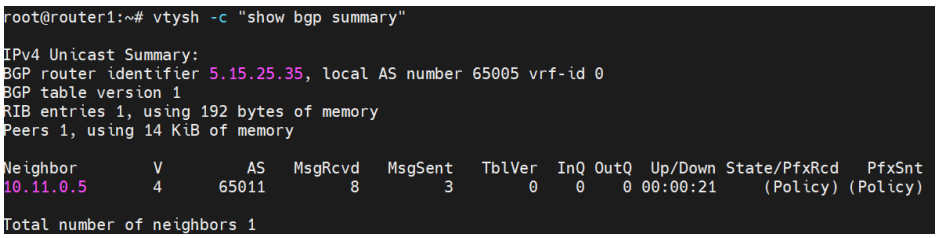

Figure 7.1: A306 iii) `bgp summary` Output

iv) assen Sie die OSPF-Konfiguration von Ihren Routern so an, dass die über BGP gelernten Routen auch den anderen Routern im jeweiligen Netz mitgeteilt werden. Hinweis: Sie können dazu entweder mit vtysh arbeiten, oder die Datei /etc/frr/ospfd.conf anpassen. Die Befehle hierzu finden sich in der Dokumentation zur FRR-Suite.

Antwort:

We just need to add `redistribute bgp` to the configuration.

Enter the FRR Shell of router1

sudo vtysh

router> enable

```
    router# configure terminal

    # Add a route to 10.11.0.hIhre Gruppei
router(config)# ip route 10.11.0.hIhre Gruppei/32 10.hIhre Gruppei.0.1

    # Add a route to 10.12.0.hIhre Gruppei
router(config)# ip route 10.12.0.hIhre Gruppei/32 10.hIhre Gruppei.0.1

    # save Configuration
    router# write
```

v) Zeigen Sie mit traceroute den Weg einer Nachricht von Ihrem PC3 zu 10.11.2.100 und 10.12.2.100. Über welche AS und welche Router wird die Nachricht vermittelt?

Antwort:

vi) Angenommen, der Lehrstuhl verlangt hohe Gebühren für den Transit Ihrer Nachrichten. Konfigurieren Sie Ihren Router1 so, dass über dieses AS nur vermittelt wird, wenn keine andere Verbindung zur Verfügung steht. Optional: Weisen Sie nach, das Ihre Konfiguration funktioniert, indem Sie mit den andern Gruppen kommunizieren und den Transit über diese abwickeln.

Antwort: We can solve this problem by lowering the local preference value.

We can set route-map Local permit to 10 and set local-preference to 50.

We also need to add nobgp ebgp-requires-policy and no bgp network import-check option.pin

vii) Konfigurieren Sie nun Router4 anstatt Router1 als BGP-Router Ihres AS, wobei die externen AS weiterhin an Router1 angeschlossen bleiben. Erläutern Sie anhand Ihrer Beobachtungen, inwiefern sich die beiden Szenarien unterscheiden. Theorie: Nennen Sie mindestens 2 Gründe, die für ein solches Szenario sprechen würden.

Antwort:

8 Appendix

8.0.1 A203

List of Figures