Institut für Informatik der Ludwig–Maximilians–Universität München

Ausarbeitung

zum

Praktikum Rechnernetze

im Wintersemester 2016/17

Block: 3

Gruppe: 4

Felix Buchdrucker, Gerhard Gröschl

Datum: 17. Dezember 2016

Inhaltsverzeichnis

A 300 Fragmentierung und IP-Tunneling (Theorie) **1**
Teilaufgabe i) . . . 1
Teilaufgabe ii) . . . 1
Teilaufgabe iii) . . . 1

A 301 IPv6 (Theorie) **3**
Teilaufgabe i) . . . 3
Teilaufgabe ii) . . . 3
Teilaufgabe iii) . . . 4
Teilaufgabe iv) . . . 4
Teilaufgabe v) . . . 4

A 302 Dynamisches Routing (Theorie) **7**
Teilaufgabe i) . . . 7
Teilaufgabe ii) . . . 7

A 303 Fragmentierung und Tunneling **9**
Teilaufgabe i) . . . 9
Teilaufgabe ii) . . . 10
Teilaufgabe iii) . . . 10
Teilaufgabe iv) . . . 11
Teilaufgabe v) . . . 12
Teilaufgabe vi) . . . 12
Teilaufgabe vii) . . . 13
Teilaufgabe viii) . . . 13

A 304 IPv6 **15**
Teilaufgabe i) . . . 15
Teilaufgabe ii) . . . 15
Teilaufgabe iii) . . . 16
Teilaufgabe iv) . . . 17
Teilaufgabe v) . . . 17
Teilaufgabe vi) . . . 18
Teilaufgabe vii) . . . 19

A 305 Distanz-Vektor Routing mit RIP **21**
Teilaufgabe i) . . . 21
Teilaufgabe ii) . . . 21
Teilaufgabe iii) . . . 22
Teilaufgabe iv) . . . 23
Teilaufgabe v) . . . 24

A 306 Link-State Routing mit OSPF **27**
Teilaufgabe i) . . . 27
Teilaufgabe ii) . . . 27
Teilaufgabe iii) . . . 28

A 307 BGP **37**
Teilaufgabe i) 37

A 300 Fragmentierung und IP-Tunneling (Theorie)

Teilaufgabe i)

Es wird davon Ausgegangen, dass es sich um ein IPv4 Paket ohne weitere Optionen handelt. Der Header, der zwangsweise in jedem Frame ist, ist damit 20 B groß.

$\lceil \frac{9000-20}{1500-20} \rceil = 7$

Teilaufgabe ii)

- Die Fragmentierung auf den Netzkomponenten des Transitnetzes verbraucht dort Rechenzeit. Dies verringert die Datenrate (wenn die Hardware nicht entsprechend überdimensioniert ist).
- Durch die Framentierung steigt die zu übertragende Datenmenge, da pro Fragment ein zusätlicher IP-Header übertragen werden muss. Es verlassen also mehr Daten das Transitnetz, als eingespeist wurden. Dies könnte Abrechnungen verzerren, wenn es überhand nimmt.
- Durch ungünstige mehrfache Fragmentierung können sich oben geschildetten Effekte verschlimmern. Wäre die Pfad-MTU bekannt, könnte der Absender eine effizientere Fragementierung vorehmen.

Teilaufgabe iii)

Vor dem eigentlichen Verbindungsaufbau zu einem neuen Edpunkt wird eine PMTUD durchgeführt. Bei IP Datagrammen, die ein TCP Segment transportieren ist das Don't Fragment Flag gesetzt. Sollte ein Segment zu groß werden und ein Router müsste es Fragmentieren, so wird dies dem Absender per ICMP mitgeteilt. In der ICMP Nachricht ist auch die MTU des Routers enthalten. Der Absender speichert diese Informationen zwischen und beachtet sie beim Versenden weiterer TCP Segmente.

A 301 IPv6 (Theorie)

Teilaufgabe i)

Es soll Beispielhaft aus der Ethernet MAC-Adresse 00:16:3e:00:00:02 die IPv6 Link-local Adresse abgeleitet werden. Dazu werden die einzelnen Felder, aus denen eine Ethernet MAC-Adresse aufgebaut ist, in ein Gerüst der IPv6-Adresse eingebettet. Die Felder der Ethernet MAC-Adresse sind in Abbildung A 301.1 auf dieser Seite zu sehen. In der Abbildung steht „u“ für ein Bit, welches angibt ob es sich um eine global eindeutige oder eine lokal verwaltete Adresse handelt. Das „g“ git darüber auskunft, ob es eine unicast oder multicast Adresse ist.

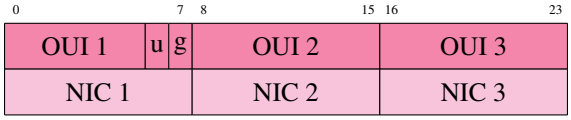

Abbildung A 301.1: Aufbau einer Eithernet MAC-Adresse.

Bei Link-local Adressen handelt es sich um Host-Adressen aus dem fe80::/10 Netz. Es ist jedoch spezifiziert, dass die restlichen Netz-Bits nach dem Prefix stets 0 sein müssen. Daran schließt sich die 64 bit Schnitstellen ID an. Da es sich Bei Ethernet MAC-Adresse jedoch nur um 48 bit IDs handelt, müssen diese „aufgeblasen“ werden. Dazu werden zwischen dem OUI und dem NIC die zwei Byte ff fe eingefügt. Das „u“-Bit wird noch invertiert und zack, fertig: Link-local Adresse. Der fertige Aufbau ist in Abbildung A 301.2 auf dieser Seite zu sehen.

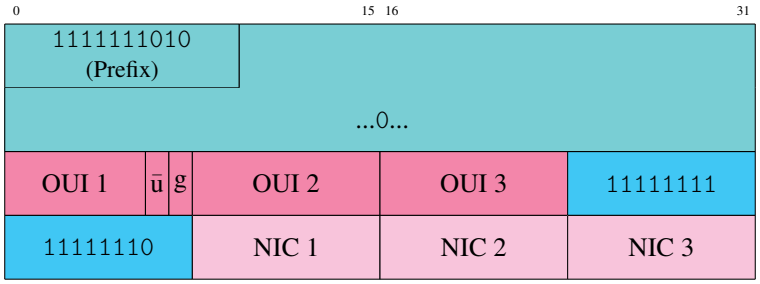

Abbildung A 301.2: Aufbau einer IPv6 Link-local Adresse für Ethernet.

Aus der Eingangs gezeigten Ethernet MAC-Adresse wird nach der Anwendung des eben beschriebenen Schemas die IPv6 Adreesse fe80:0000:0000:0000:0216:3eff:fe00:0002/64 beziehungsweise in gekürzter Schreibweise die fe80::216:3eff:fe00:2/64. Das Byte mit dem invertierten „u“-Bit ist in allen Adressen jeweils unterstrihen.

Teilaufgabe ii)

Gar nicht.
Wer sowas *ähnliches* will kann Multicast verwenden

Teilaufgabe iii)

Das Pendant sind Unique Local IPv6 Unicast Addresses (ULA). Sie besitzen das `fc00::/7` Präfix. Ihr Aufbau ist in Abbildung A 301.3 auf dieser Seite gezeigt. Für das „L“ Feld ist bislnag nur der Wert 1 spezifiziert. Man erkennt ULAs daher leicht an ihrem spezifischeren Prefix `fd00::/8`. Die *Global ID* muss Weltweit eindeutig sein und muss daher per Zufall zugewiesen werden.

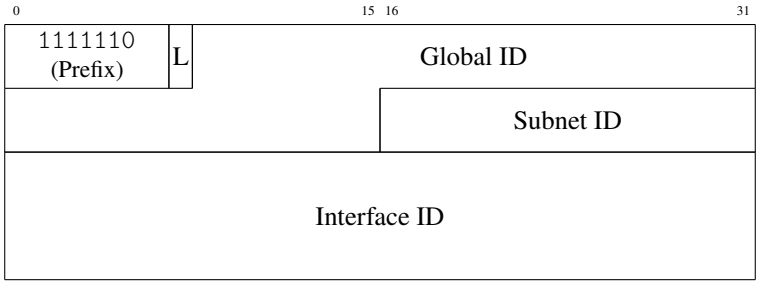

Abbildung A 301.3: Aufbau einer Unique Local IPv6 Unicast Addresses.

Teilaufgabe iv)

Prefix	RFC	Verwendung
`::1/128`	RFC4291	Loopback Address
`::/128`	RFC4291	Unspecified Address
`::ffff:0:0/96`	RFC4291	IPv4-mapped Address
`64:ff9b::/96`	RFC6052	IPv4-IPv6 Translat.
`100::/64`	RFC6666	Discard-Only Address Block
`2000::/3`	RFC4291	Global Unicast
`2001::/23`	RFC2928	IETF Protocol Assignments
`2001::/32`	RFC4380	TEREDO
`2001:1::1/128`	RFC7723	Port Control Protocol Anycast
`2001:2::/48`	RFC5180	Benchmarking
`2001:3::/32`	RFC7450	AMT
`2001:4:112::/48`	RFC7535	AS112-v6
`2001:5::/32`	RFC7954	EID Space for LISP (Managed by RIPE NCC)
`2001:20::/28`	RFC7343	ORCHIDv2
`2001:db8::/32`	RFC3849	Documentation
`2002::/16`	RFC3056	6to4
`2620:4f:8000::/48`	RFC7534	Direct Delegation AS112 Service
`fc00::/7`	RFC4193	Unique-Local
`fe80::/10`	RFC4291	Linked-Scoped Unicast
`ff00::/8`	RFC4291	Multicast

Teilaufgabe v)

Alle Regulären Ausdrücke werden schreibungsunabhängig angewendet und Leerzeichen werden ignoriert.

a)

```
1 ^((25[0-5]|2[0-4][0-9]|1[0-9]{0,2}|0)\.){3}(25[0-5]|2[0-4][0-9]|1[0-9]{0,2}|0)$
```

b)

```
^(([0-9a-f][1-9a-f]?)\.){3}([0-9a-f][1-9a-f]?)$
```

c)

```
^(
   (                    : (:|(:[0-9a-f]{1,4}){1,7}) )|
   ( ([0-9a-f]{1,4}:){1} (:|(:[0-9a-f]{1,4}){1,6}) )|
   ( ([0-9a-f]{1,4}:){2} (:|(:[0-9a-f]{1,4}){1,5}) )|
   ( ([0-9a-f]{1,4}:){3} (:|(:[0-9a-f]{1,4}){1,4}) )|
   ( ([0-9a-f]{1,4}:){4} (:|(:[0-9a-f]{1,4}){1,3}) )|
   ( ([0-9a-f]{1,4}:){5} (:|(:[0-9a-f]{1,4}){1,2}) )|
   ( ([0-9a-f]{1,4}:){6} (:|(:[0-9a-f]{1,4}){1,1}) )|
   ( ([0-9a-f]{1,4}:){7}  :                         )|
   ( ([0-9a-f]{1,4}:){7}    ( [0-9a-f]{1,4}){1,1}  )
)$
```

A 302 Dynamisches Routing (Theorie)

Teilaufgabe i)

a) Beim Distanz-Vektor Verfahren bekommen Router von ihren direkt verbundenen Nachbarroutern Informationen zugesendet. Sie enthalten die bislang kürzersten Pfad des Senders zu anderen Routern. Der Empfängerrouter kann jedoch nicht wissen ob die Pfade zu den empfangenen Distanzen über ihn selbst selbst gehen. Es wird somit keine Routingschleife erkannt. Wird die Distanz zum Ziel über die Routingschleife berechnet, nimmt die Distanz mit jeder neuen Berechnung zu. Dies gibt dem Problem seinen Namen. Es kann auftreten, wenn ein Teilgraph keine Verbindung mehr zu einem ausenstehenden Router hat.

b) Bei Split-Horizon werden die Information über die kürzerste Distanz zu einem Ziel nicht an den Nachbarrouter gesendet welcher der nächste Hop zu dem Ziel währe. Dadurch kann ein Empfänger ausschließen, dass er auf den letzen beiden Hops Teil des Pfades war. Eine direkte Routingschliefe wird somit verhindert und Count-to-Infinity kann dort nicht auftreten.

c) Ein Router, der von seinem Nachbarrouter die kürzersten Distanzen zu einem Ziel emfpangen hat kann jedoch nicht für den gesammten Pfad ausschließen, dass er ein Teil davon ist. Es ist somit noch möglich, dass Routingschleifen über mehrer als zwei Knoten hinweg entsehen.

d) Während Split-Horizon alleine das Zählen bis Unendlich verhindert, blokiert Poison Reverse den Grund dafür. Der Auslöser für Count-to-Infinity war, dass ein Teilgraph alle Verbindungen zu einem Ziel verloren hat. Dabei Blieb das Ziel jedoch weiterhin in den Routingtabellen des Abgeschnittenen Teilgraphen und eine Routingschleife stellt für das Distanz-Vektor Verfahren den einzigen Ausweg da.

   Bei Split-Horizon mit Poison Reverse Teilt ein Router, der bemerkt hat das ein Ziel nicht mehr erreichbar ist, dies seinen Nachbarroutern mit. Diesmal wird die Information auch dem Nachbarrouter mitgeteilt, von dem er die kürzerste Route gerlernt hat.

Teilaufgabe ii)

a) Damit triggered updates zu 100% Funktionieren, müsste das Ausgelöste update instantan an alle Router gesendet werden. Es dürften keine „normalen“ Updaten versendet werden, währen das ausgelöste Update Propagiert. Dies ist jedoch unmöglich sicher zu stellen. Jedoch ist das Verfahren gut genug um die meisten Count-to-Infinitys zu verhindern.

b) Da die Router die Empfangenen Updates selber erst bearbeiten müssen, dabei auch noch Puffer im Spiel sind und Signale sich nicht unendlich schnell ausbreiten können, ist es unmöglich die Bedingungen zu erfüllen.

A 303 Fragmentierung und Tunneling

Teilaufgabe i)

Alle beteiligten Rechner bekommen ihre IP-Adresen, wie bereits in A101 iv) beschrieben. Dies entspricht auf der in **??** auf Seite ?? gezeigten konfiguration. Es müssen daher nur noch die Routen hinzugefügt werden, wie im folgenden beschrieben ist.

Terminal auf `pc1`

```
# ip route add default via 10.4.1.1
```

Terminal auf `pc2`

```
# ip route add default via 10.4.2.1
```

Terminal auf `router1`

```
# ip route add 10.4.2.0/24 via 10.4.14.4
```

Terminal auf `router2`

```
# ip route add 10.4.1.0/24 via 10.4.23.3
```

Terminal auf `router3`

```
# ip route add 10.4.1.0/24 via 10.4.34.4
# ip route add 10.4.2.0/24 via 10.4.23.2
```

Terminal auf `router4`

```
# ip route add 10.4.1.0/24 via 10.4.14.1
# ip route add 10.4.2.0/24 via 10.4.34.3
```

Das Pakete auf dem Vorgegebenen Pfad vermittelt werden ist der folgenden Ausgabe zu entnehmen.

Terminal auf `pc1`

```
# traceroute 10.4.2.2
traceroute to 10.4.2.2 (10.4.2.2), 30 hops max, 60 byte packets
 1  10.4.1.1 (10.4.1.1)  1.539 ms  1.475 ms  1.414 ms
 2  10.4.14.4 (10.4.14.4)  2.167 ms  2.114 ms  2.057 ms
 3  10.4.34.3 (10.4.34.3)  2.882 ms  2.821 ms  2.763 ms
 4  10.4.23.2 (10.4.23.2)  4.323 ms  4.273 ms  4.214 ms
 5  10.4.2.2 (10.4.2.2)  5.881 ms  5.832 ms  5.775 ms
```

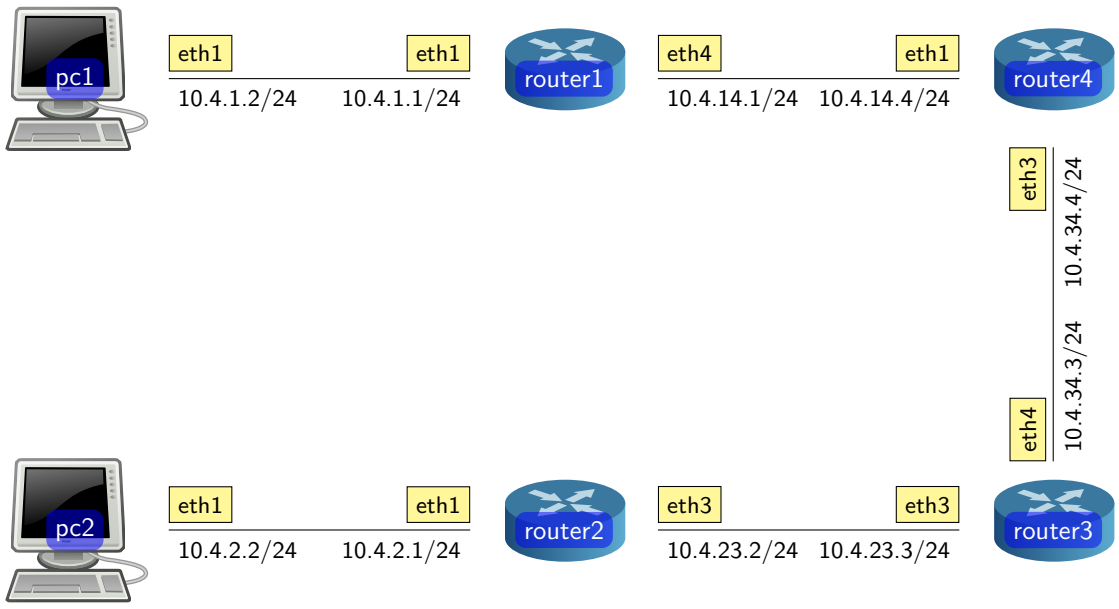

Abbildung A 303.1: ???

Teilaufgabe ii)

Rechnername	Schnittstelle	IP-Adresse	MTU
pc1	eth1	10.4.1.2	1500 B
pc2	eth1	10.4.2.2	1500 B
router1	eth1	10.4.1.1	1500 B
router1	eth4	10.4.14.1	1500 B
router2	eth3	10.4.23.2	1500 B
router2	eth1	10.4.2.1	1500 B
router3	eth4	10.4.34.3	1500 B
router3	eth3	10.4.23.3	1500 B
router4	eth1	10.4.14.4	1500 B
router4	eth3	10.4.34.4	1500 B

Teilaufgabe iii)

Zuerst wird die MTU auf den geforderten Schnitstellen beschränkt.

Terminal auf router3

```
# ip link set eth3 mtu 1000
```

Terminal auf router4

```
# ip link set eth3 mtu 1100
```

Im anschluss wird ein ICMP Echo-Request von pc1 an pc2 geschickt. Dabei sind die Nutzdaten von ICMP 1200 B groß und das *Don't Fragment* Flag gesetzt. Wie der folgenden Ausgabe zu entnehmen ist, kann das IP-Datagramm nicht an das Ziel vermittelt werden. Dies wird vom ersten Hop, der es aufgrund seiner MTU

begrenzung nicht weitersenden kann, mit einer entsprechenden ICMP-Nachricht dem Absender signalisiert. In diesem Fall wurde die Weiterleitung von router4 verweigert. Dort ist die MTU, an der das Datagramm den Router wieder hätte verlassen sollen, mit 1100 B kleiner als das IP-Datagramm mit 1228 B.

Terminal auf pc1

```
# ping -c 1 -s 1200 -M do 10.4.2.2
PING 10.4.2.2 (10.4.2.2) 1200(1228) bytes of data.
ping: local error: Message too long, mtu=1100

--- 10.4.2.2 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Teilaufgabe iv)

Für diese Aufgabe wurde zunächst auf allen Routern tcpdump gestartet. Im Anschluss wurde ein ICMP Echo-Request von pc1 an pc2 gesendet.

Den Ausgaben der Programme ist zu entnehmen, dass das IP-Datagramm unfragmentiert von pc1 zu router1 gelangt. Da die Schnitstelle, über die das Datagramm den Router wieder verlässt, eine ausreichend große MTU hat, wird das Datagramm unfragmentiert weitervermittelt. Beim nächsten Hop, router4, hat die ausgehende Schnitstelle jedoch eine zu kleine MTU von 1100 B. In Folge muss das Datagramm in ein 1100 B und ein 148 B großes Datagramm zerlegt werden. Diese beiden Fragmente werden von router3, dem nächsten Hop, wieder zusammengeführt (defragmentiert), bevor das IP-Paket weiter vermitelt wird. Bei router3 ist die Ausgehende Schnitstelle auf eine MTU von 1000 B begrenzt. Das IP-Paket wird daher erneut in zwei Fragmente unterteilt. Die IP-Datagramme sind aufgrund der kleineren MTU jedoch 996 B und 252 B groß. router2, der letzte Router auf der Strecke, baut das IP-Paket abermals zusammen und schickt es als ein einzelnes IP-Datagramm an pc2.

Terminal auf pc1

```
# ping -c 1 -s 1200 10.4.2.2
PING 10.4.2.2 (10.4.2.2) 1200(1228) bytes of data.
1208 bytes from 10.4.2.2: icmp_seq=1 ttl=60 time=1.62 ms

--- 10.4.2.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.626/1.626/1.626/0.000 ms
```

In der Ausgabe von tcpdump werden normalerweise ausgehende IP-Datagramme *und* IP-Pakete angezeigt, wenn diese fragmentiert wurden. Der übersicht halber wurden die IP-Pakete entfernt. Dadurch sind nur noch die IP-Datagramme zu sehen, wie sie auch über die Leitung gehen. Eingehend werden leider keine fragmente angezeigt, da Linux diese immer zusammen baut damit netfilter seine volle funktionalität entfalten kann, auch wenn diese nicht genutzt wird.

Terminal auf router1

```
# tcpdump -evi any icmp
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
17:44:03.755372  In 00:16:3e:00:00:02 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 64, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
17:44:03.755396 Out 00:16:3e:00:00:11 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 63, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
```

Terminal auf router4

```
# tcpdump -evi any icmp
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
17:44:03.751816  In 00:16:3e:00:00:11 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 63, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
```

```
17:44:03.751842 Out 00:16:3e:00:00:25 (oui Unknown) ethertype IPv4 (0x0800), length 1116: (tos 0x0,
⇒ttl 62, id 24478, offset 0, flags [+], proto ICMP (1), length 1100)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1080
17:44:03.751859 Out 00:16:3e:00:00:25 (oui Unknown) ethertype IPv4 (0x0800), length 164: (tos 0x0,
⇒ttl 62, id 24478, offset 1080, flags [none], proto ICMP (1), length 148)
    10.4.1.3 > 10.4.2.3: icmp
```

Terminal auf router3

```
# tcpdump -evi any icmp
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
17:44:03.751796  In 00:16:3e:00:00:25 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 62, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
17:44:03.751830 Out 00:16:3e:00:00:20 (oui Unknown) ethertype IPv4 (0x0800), length 1012: (tos 0x0,
⇒ttl 61, id 24478, offset 0, flags [+], proto ICMP (1), length 996)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 976
17:44:03.751841 Out 00:16:3e:00:00:20 (oui Unknown) ethertype IPv4 (0x0800), length 268: (tos 0x0,
⇒ttl 61, id 24478, offset 976, flags [none], proto ICMP (1), length 252)
    10.4.1.3 > 10.4.2.3: icmp
```

Terminal auf router2

```
# tcpdump -evi any icmp
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
17:44:03.755453  In 00:16:3e:00:00:20 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 61, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
17:44:03.755485 Out 00:16:3e:00:00:13 (oui Unknown) ethertype IPv4 (0x0800), length 1244: (tos 0x0,
⇒ttl 60, id 24478, offset 0, flags [none], proto ICMP (1), length 1228)
    10.4.1.3 > 10.4.2.3: ICMP echo request, id 631, seq 1, length 1208
```

Teilaufgabe v)

Die TTL wurde explizit auf den Standardwert von 64 s gesetzt, damit bei Abschnitt A 303 vi) auf dieser Seite keine Pakete verlohren gehen. Als Tunnelendpunkte werden die Schnitstellen in den PC-Netzen verwendet, da für diese bereits die routen existieren. Damit diese Routen weiterhin funktioieren wurden nur Host Routen für pc1 und pc2 durch den Tunnel eingerichtet. Die Tunnelschnitstellen selbst bezeihen ihre Adressen aus dem Netz 10.4.100.0/24.

Terminal auf router1

```
# ip tunnel add tun0 mode gre local 10.4.1.1 remote 10.4.2.1 ttl 64
# ip link set tun0 up
# ip address add 10.4.100.1/24 dev tun0
# ip route add 10.4.2.2 via 10.4.100.2
```

Terminal auf router2

```
# ip tunnel add tun0 mode gre local 10.4.2.1 remote 10.4.1.1 ttl 64
# ip link set tun0 up
# ip address add 10.4.100.2/24 dev tun0
# ip route add 10.4.1.2 via 10.4.100.1
```

Teilaufgabe vi)

Den folgenden beiden Ausgaben ist kann man entnehmen, dass Pakete von pc1 nach pc2 und auch wieder zurück über den Tunnel geleitet werden.

Terminal auf pc1

```
# traceroute 10.4.2.2
traceroute to 10.4.2.2 (10.4.2.2), 30 hops max, 60 byte packets
 1  10.4.1.1 (10.4.1.1)  0.966 ms  0.926 ms  0.895 ms
 2  10.4.100.2 (10.4.100.2)  2.257 ms  2.226 ms  2.192 ms
 3  10.4.2.2 (10.4.2.2)  4.305 ms  4.280 ms  4.247 ms
```

Terminal auf pc2

```
# traceroute 10.4.1.2
traceroute to 10.4.1.2 (10.4.1.2), 30 hops max, 60 byte packets
 1  10.4.2.1 (10.4.2.1)  0.444 ms  0.365 ms  1.723 ms
 2  10.4.100.1 (10.4.100.1)  4.498 ms  4.437 ms  4.390 ms
 3  10.4.1.2 (10.4.1.2)  4.967 ms  4.908 ms  4.837 ms
```

Teilaufgabe vii)

Vergleich der Ausgaben von Traceroute von pc1 an pc2 aus Abschnitt A 303 i) auf Seite 9 und Abschnitt A 303 vi) auf der vorherigen Seite.

- Das getunnelte IP-Datagramm wird von weniger Hops beeinflusst. So sieht man dem Pfad durch den Tunnel nicht an das die Datagramme, wenn auch gekapselt, immer noch über sie selbe anzahl an Hops geleitet werden.

Teilaufgabe viii)

Terminal auf pc1

```
# ping -c 1 -t 20 10.4.2.2
```

Wie bei fragmentierung auch, werden Pakete mehrfach angezeigt (gekapselt und ungekapselt). Diese dopplungen wurde der Übersicht halber entfernt.

Terminal auf router1

```
# tcpdump -evi any 'not port 22'
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
06:47:25.435017  In 00:16:3e:00:00:02 (oui Unknown) ethertype IPv4 (0x0800), length 100: (tos 0x0,
⇒ttl 20, id 5457, offset 0, flags [DF], proto ICMP (1), length 84)
    10.4.1.2 > 10.4.2.2: ICMP echo request, id 591, seq 1, length 64
06:47:25.435051 Out 00:16:3e:00:00:11 (oui Unknown) ethertype IPv4 (0x0800), length 124: (tos 0x0,
⇒ttl 64, id 5361, offset 0, flags [DF], proto GRE (47), length 108)
    10.4.1.1 > 10.4.2.1: GREv0, Flags [none], proto IPv4 (0x0800), length 88
        (tos 0x0, ttl 19, id 5457, offset 0, flags [DF], proto ICMP (1), length 84)
    10.4.1.2 > 10.4.2.2: ICMP echo request, id 591, seq 1, length 64
```

Terminal auf router2

```
# tcpdump -evi any 'not port 22'
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
06:47:25.434957  In 00:16:3e:00:00:20 (oui Unknown) ethertype IPv4 (0x0800), length 124: (tos 0x0,
⇒ttl 62, id 5361, offset 0, flags [DF], proto GRE (47), length 108)
    10.4.1.1 > 10.4.2.1: GREv0, Flags [none], proto IPv4 (0x0800), length 88
        (tos 0x0, ttl 19, id 5457, offset 0, flags [DF], proto ICMP (1), length 84)
    10.4.1.2 > 10.4.2.2: ICMP echo request, id 591, seq 1, length 64
06:47:25.435001 Out 00:16:3e:00:00:13 (oui Unknown) ethertype IPv4 (0x0800), length 100: (tos 0x0,
⇒ttl 18, id 5457, offset 0, flags [DF], proto ICMP (1), length 84)
    10.4.1.2 > 10.4.2.2: ICMP echo request, id 591, seq 1, length 64
```

Der markanterste unterschied ist wohl bei der TTL zu finden. Die TTL des inneren und äußeren IP-Datagramms sind voneinander unabhängig. So wird die äußere TTL beim erzeugen des GRE-Pakets auf den in Abschnitt A 303 v) auf Seite 12 gesetzten wert von 64 s gesetzt. Wenn dass Paket bei router2 ankommt ist die TTL entsprechend dem gegangenen Pfad reduziert. Die TTL des inneren IP-Datagramms ist jedoch konstant bei 19 s geblieben. Wenn das IP-Datagramm, welches die ICMP Nachricht transportiert, von den Routen ein- beziehungsweise ausgepackt wird, wird die TTL erwartungsgemäs von 20 s auf 19 s beziehungsweise von 19 s auf 18 s reduziert.

Desweitern ist zu sehen, dass die IP-Adressen des außeren IP-Datagramms die der Tunnelendpunkte sind. Die Adressen des inneren Datagramms sind die der eigentlichen Kommunikationspartner, den PCs.

A 304 IPv6

Teilaufgabe i)

Terminal auf `router1`

```
# ip address del 10.4.1.1 dev eth1
# ip address del 10.4.12.1 dev eth2
# ip address del 10.4.13.1 dev eth3
# ip address del 10.4.14.1 dev eth4
```

Terminal auf `router2`

```
# ip address del 10.4.2.1 dev eth1
# ip address del 10.4.12.2 dev eth2
# ip address del 10.4.23.2 dev eth3
# ip address del 10.4.24.2 dev eth4
```

Terminal auf `router3`

```
# ip address del 10.4.3.1 dev eth1
# ip address del 10.4.13.3 dev eth2
# ip address del 10.4.23.3 dev eth3
# ip address del 10.4.34.3 dev eth4
```

Terminal auf `router4`

```
# ip address del 10.4.14.4 dev eth1
# ip address del 10.4.24.4 dev eth2
# ip address del 10.4.34.4 dev eth3
```

Teilaufgabe ii)

Zunächst wurde auf `router1` der Datenverkehr an der Schnitstelle zu `router4` mitgeschnitten. Dann wurde mit einem leeren Nachbarschaftszwischenspeicher die Link-Local Adressse von `router4` „gepingt“. Die Ergebnisse sind im folgenden zu sehen.

Terminal auf `router1`

```
# ip -6 neighbour flush all
# ping6 -c 1 'fe80::216:3eff:fe00:23%eth4'
PING fe80::216:3eff:fe00:23%eth4(fe80::216:3eff:fe00:23) 56 data bytes
64 bytes from fe80::216:3eff:fe00:23: icmp_seq=1 ttl=64 time=0.516 ms

--- fe80::216:3eff:fe00:23%eth4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.516/0.516/0.516/0.000 ms
```

Terminal auf `router1`

```
# tcpdump -i eth4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on eth4, link-type EN10MB (Ethernet), capture size 262144 bytes
04:39:33.386462 IP6 fe80::216:3eff:fe00:11 > ff02::1:ff00:23: ICMP6, neighbor solicitation, who has
⇒fe80::216:3eff:fe00:23, length 32
04:39:33.386752 IP6 fe80::216:3eff:fe00:23 > fe80::216:3eff:fe00:11: ICMP6, neighbor advertisement,
⇒tgt is fe80::216:3eff:fe00:23, length 32
04:39:33.386766 IP6 fe80::216:3eff:fe00:11 > fe80::216:3eff:fe00:23: ICMP6, echo request, seq 1,
⇒length 64
04:39:33.386921 IP6 fe80::216:3eff:fe00:23 > fe80::216:3eff:fe00:11: ICMP6, echo reply, seq 1,
⇒length 64
04:39:38.389900 IP6 fe80::216:3eff:fe00:23 > fe80::216:3eff:fe00:11: ICMP6, neighbor solicitation,
⇒who has fe80::216:3eff:fe00:11, length 32
04:39:38.389952 IP6 fe80::216:3eff:fe00:11 > fe80::216:3eff:fe00:23: ICMP6, neighbor advertisement,
⇒tgt is fe80::216:3eff:fe00:11, length 24
```

Der letzten Ausgabe ist zu entnehmen, dass der Multicast zur Adressauflösung an die MAC-Adresse `33:33:ff:00:00:23` geschickt wird.

Teilaufgabe iii)

Die IPv6-Adressen werden nach dem selben schema vergeben wie die IPv4-Adressen.

Terminal auf `router1`

```
# ip address add fc00:dead:beef:4014::1/64 dev eth4
# ip route add default via fc00:dead:beef:4014::4
# ip -6 route
fc00:dead:beef:4014::/64 dev eth4  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth4  proto kernel  metric 256
default via fc00:dead:beef:4014::4 dev eth4  metric 1024
```

Terminal auf `router2`

```
# ip address add fc00:dead:beef:4024::2/64 dev eth4
# ip route add default via fc00:dead:beef:4024::4
# ip -6 route
fc00:dead:beef:4024::/64 dev eth4  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth4  proto kernel  metric 256
default via fc00:dead:beef:4024::4 dev eth4  metric 1024
```

Terminal auf `router3`

```
# ip address add fc00:dead:beef:4034::3/64 dev eth4
# ip route add default via fc00:dead:beef:4034::4
# ip -6 route
fc00:dead:beef:4034::/64 dev eth4  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth4  proto kernel  metric 256
default via fc00:dead:beef:4034::4 dev eth4  metric 1024
```

Terminal auf `router4`

```
# sysctl -w net.ipv6.conf.all.forwarding=1
# ip address add fc00:dead:beef:4014::4/64 dev eth1
# ip address add fc00:dead:beef:4024::4/64 dev eth2
# ip address add fc00:dead:beef:4034::4/64 dev eth3
# ip -6 route
fc00:dead:beef:4014::/64 dev eth1  proto kernel  metric 256
fc00:dead:beef:4024::/64 dev eth2  proto kernel  metric 256
fc00:dead:beef:4034::/64 dev eth3  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
fe80::/64 dev eth3  proto kernel  metric 256
```

Teilaufgabe iv)

Zunächst wurde auf `router1` der Datenverkehr an der Schnitstelle zu `router4` mitgeschnitten. Dann wurde mit einem leeren Nachbarschaftszwischenspeicher die Globale Adressse von `router4` „gepingt“. Die Ergebnisse sind im folgenden zu sehen.

Terminal auf `router1`

```
# ip -6 neighbour flush all
# ping6 -c 1 fc00:dead:beef:4014::4
PING fc00:dead:beef:4014::4(fc00:dead:beef:4014::4) 56 data bytes
64 bytes from fc00:dead:beef:4014::4: icmp_seq=1 ttl=64 time=0.629 ms

--- fc00:dead:beef:4014::4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.629/0.629/0.629/0.000 ms
```

Terminal auf `router1`

```
# tcpdump -i eth4
04:45:29.443077 IP6 fc00:dead:beef:4014::1 > ff02::1:ff00:4: ICMP6, neighbor solicitation, who has
⇒fc00:dead:beef:4014::4, length 32
04:45:29.443431 IP6 fc00:dead:beef:4014::4 > fc00:dead:beef:4014::1: ICMP6, neighbor advertisement,
⇒tgt is fc00:dead:beef:4014::4, length 32
04:45:29.443445 IP6 fc00:dead:beef:4014::1 > fc00:dead:beef:4014::4: ICMP6, echo request, seq 1,
⇒length 64
04:45:29.443627 IP6 fc00:dead:beef:4014::4 > fc00:dead:beef:4014::1: ICMP6, echo reply, seq 1,
⇒length 64
04:45:34.453786 IP6 fe80::216:3eff:fe00:23 > fc00:dead:beef:4014::1: ICMP6, neighbor solicitation,
⇒who has fc00:dead:beef:4014::1, length 32
04:45:34.453848 IP6 fc00:dead:beef:4014::1 > fe80::216:3eff:fe00:23: ICMP6, neighbor advertisement,
⇒tgt is fc00:dead:beef:4014::1, length 24
04:45:39.463432 IP6 fe80::216:3eff:fe00:11 > fe80::216:3eff:fe00:23: ICMP6, neighbor solicitation,
⇒who has fe80::216:3eff:fe00:23, length 32
04:45:39.463794 IP6 fe80::216:3eff:fe00:23 > fe80::216:3eff:fe00:11: ICMP6, neighbor advertisement,
⇒tgt is fe80::216:3eff:fe00:23, length 24
04:45:44.469757 IP6 fe80::216:3eff:fe00:23 > fe80::216:3eff:fe00:11: ICMP6, neighbor solicitation,
⇒who has fe80::216:3eff:fe00:11, length 32
04:45:44.469807 IP6 fe80::216:3eff:fe00:11 > fe80::216:3eff:fe00:23: ICMP6, neighbor advertisement,
⇒tgt is fe80::216:3eff:fe00:11, length 24
```

Bei der eigentlichen Adressauflösung ist kein großer unterschied zu Aufgabe Abschnitt `A 304 iii)` auf der vorherigen Seite vorhanden. Es werden erwartungsgemäs die globalen anstad der link-lokalen Adressen verwendet.

Im Nachgang gibt es jedoch markante unterschiede. Bei Teilaufgabe iv) ist das selbe verhalten wie bei IPv4 in A204 iii) zu sehen. So wird 5 s nach der ersten eigentlich Adressauflösung eine zweite durchgeführt. Mit der zweiten überprfüft der ziel Rechner die Adresse des Quellrechners. Denn diese hatte er nur passiv durch dessen Namensauflöung gelernt.

Diese zweite Namensauflösung geschieht mit der Link-Lokalen Adresse. Dadurch entsteht ein eunterschied zwischen dieser Aufgabe und der Teilaufgabe iv). Während bei der Teilaufgabe iv) die Link-Lokale adresse dem Quellrechner bereits bekannt war, ist sie es ihm in dieser Aufgabe noch nicht. Dadurch wird hier eine dritte Adressauflösung nach der Link-Lokalen Adresse des Zielrechners durchgeführt. Hierbei wird abermals die Link-Lokale Adresse verwendet, welche der Zielrechner seinerseit in einer vierten Adressauflösung verifiziert.

Teilaufgabe v)

Zunächst wurde auf den beide Routern die Konfigurationsdatei des Router Advertisment Dienstes angelegt.

Datei `/etc/radvd.conf`

```
1 interface eth1 {
2    AdvSendAdvert on;
3    prefix ::/64 {};
```

```
4 };
```

Danach wurden den Schnistellen der PC-Netze, analog zu unserem Schema der IPv4 Adresszuweisung, eine IPv6 Adresse zugweisen. Bevor schließlich der Dienst gestartet wurde, musste noch das weiterleiten von IPv6-Paketen aktiviert werden.

Terminal auf router1

```
# ip address add fc00:dead:beef:4001::1/64 dev eth1
# sysctl -w net.ipv6.conf.all.forwarding=1
# systemctl start radvd
```

Terminal auf router2

```
# ip address add fc00:dead:beef:4002::1/64 dev eth1
# sysctl -w net.ipv6.conf.all.forwarding=1
# systemctl start radvd
```

Nachdem der Dienst lief wurden die Schnitstellen von pc1 und pc2 deaktivierunt und dann wieder aktiviert, damit eine erneute automatishe konfiguration der IPv6 Adressen ausgelöst wurde. Währenddesswen wurde der Netzverkehr dieser Schnitstellen mitgeschnitten, welcher im folgenden zu sehen ist.

Terminal auf pc1

```
# tcpdump -nvi eth1
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
04:36:44.033866 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::216:3eff:fe00:2 >
⇒ff02::2: [icmp6 sum ok] ICMP6, router solicitation, length 16
          source link-address option (1), length 8 (1): 00:16:3e:00:00:02
04:36:44.035068 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56) fe80::216:3eff:fe00:8 >
⇒ff02::1: [icmp6 sum ok] ICMP6, router advertisement, length 56
        hop limit 64, Flags [none], pref medium, router lifetime 1800s, reachable time 0s, retrans
        ⇒time 0s
          prefix info option (3), length 32 (4): fc00:dead:beef:4001::/64, Flags [onlink, auto,
          ⇒router], valid time 86400s, pref. time 14400s
          source link-address option (1), length 8 (1): 00:16:3e:00:00:08
```

Terminal auf pc2

```
# tcpdump -nvi eth1
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
04:34:49.574884 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 16) fe80::216:3eff:fe00:4 >
⇒ff02::2: [icmp6 sum ok] ICMP6, router solicitation, length 16
          source link-address option (1), length 8 (1): 00:16:3e:00:00:04
04:34:49.576025 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 56) fe80::216:3eff:fe00:13 >
⇒ff02::1: [icmp6 sum ok] ICMP6, router advertisement, length 56
        hop limit 64, Flags [none], pref medium, router lifetime 1800s, reachable time 0s, retrans
        ⇒time 0s
          prefix info option (3), length 32 (4): fc00:dead:beef:4002::/64, Flags [onlink, auto,
          ⇒router], valid time 86400s, pref. time 14400s
          source link-address option (1), length 8 (1): 00:16:3e:00:00:13
```

Den Mitschnitten ist zu entehmen, dass die automatische Konfiguration ICMPv6 Nachrichten verwendet. Dabei meldet sich der Rechner, der eine Adresse will, per Multicast bei allen Routern. Diese antworten dann an *alle* Rechner mit den Prefixen die verwendet werden können. Daraus kann sich dann der Rechner dann eine Adresse heraussuchen.

Teilaufgabe vi)

Da wir zufälligerweise in Abschnitt A 304 iii) auf Seite 16 auf router1 und router2 standard Routen verwendet haben, muss hier nur noch router4 konfiguriert werden.

Terminal auf router4

```
# ip route add fc00:dead:beef:4001::/64 via fc00:dead:beef:4014::1
# ip route add fc00:dead:beef:4002::/64 via fc00:dead:beef:4024::2
```

Dass die Konfiguration funktionier kann der folgenden Ausgabe entnommen werden.

Terminal auf pc1

```
# traceroute6 -n fc00:dead:beef:4002:216:3eff:fe00:4
traceroute to fc00:dead:beef:4002:216:3eff:fe00:4 (fc00:dead:beef:4002:216:3eff:fe00:4), 30 hops max
⇒, 80 byte packets
 1  fc00:dead:beef:4001::1  0.307 ms  1.468 ms  1.235 ms
 2  fc00:dead:beef:4014::4  2.553 ms  2.498 ms  2.438 ms
 3  fc00:dead:beef:4024::2  3.266 ms  3.209 ms  3.146 ms
 4  fc00:dead:beef:4002:216:3eff:fe00:4  3.304 ms  3.246 ms  3.185 ms
```

Teilaufgabe vii)

Zunächt bekommen die Router wieder ihre alten IPv4-Adressen aus dem PC-Netz zurück. Dann wird eine Tunnel erstelt. Im gegensatz zu Abschnitt A 303 v) auf Seite 12 können diesmal die gesammten PC-Netze weitergeleitet werden.

Terminal auf router1

```
# ip address add 10.4.1.1/24 dev eth1
# ip -6 tunnel add tun0 mode ip4ip6 local fc00:dead:beef:4014::1 remote fc00:dead:beef:4024::2
# ip address add 10.4.200.1/24 dev tun0
# ip link set tun0 up
# ip route add 10.4.2.0/24 via 10.4.200.2
```

Terminal auf router2

```
# ip address add 10.4.2.1/24 dev eth1
# ip -6 tunnel add tun0 mode ip4ip6 local fc00:dead:beef:4024::2 remote fc00:dead:beef:4014::1
# ip address add 10.4.200.2/24 dev tun0
# ip link set tun0 up
# ip route add 10.4.1.0/24 via 10.4.200.1
```

Da die PCs in Abschnitt A 303 i) auf Seite 9 ihre IPv4 Adressen behalten durfen, muss hier nur noch die standard Route hinzugefügt werden.

Terminal auf pc1

```
# ip route add default via 10.4.1.1
```

Terminal auf pc2

```
# ip route add default via 10.4.2.1
```

Das die Konfiguration funktioniert kann man der folgenden Ausgabe entnehmen.

Terminal auf pc1

```
# traceroute 10.4.2.2
traceroute to 10.4.2.2 (10.4.2.2), 30 hops max, 60 byte packets
 1  10.4.1.1 (10.4.1.1)  0.966 ms  0.920 ms  0.889 ms
 2  10.4.200.2 (10.4.200.2)  3.312 ms  3.283 ms  3.250 ms
 3  10.4.2.2 (10.4.2.2)  3.369 ms  3.340 ms  3.306 ms
```

Das die Daten auch gekapselt werden ist dem folgenden Mitschnitt des Datenverkehrs zu entnehmen. Wie bereits beim Mitschnitt des GRE Tunnels, werden Pakete mehrfach angezeigt (gekapselt und ungekapselt). Diese dopplungen wurde der Übersicht halber entfernt.

Terminal auf `router1`

```
# tcpdump -eni any 'not port 22'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
07:31:50.954636  In 00:16:3e:00:00:02 ethertype IPv4 (0x0800), length 100: 10.4.1.2 > 10.4.2.2: ICMP
⇒ echo request, id 536, seq 297, length 64
07:31:50.954674 Out 00:16:3e:00:00:11 ethertype IPv6 (0x86dd), length 140: fc00:dead:beef:4014::1 >
⇒fc00:dead:beef:4024::2: 10.4.1.2 > 10.4.2.2: ICMP echo request, id 536, seq 297, length 64
07:31:50.955573  In 00:16:3e:00:00:23 ethertype IPv6 (0x86dd), length 140: fc00:dead:beef:4024::2 >
⇒fc00:dead:beef:4014::1: 10.4.2.2 > 10.4.1.2: ICMP echo reply, id 536, seq 297, length 64
07:31:50.955610 Out 00:16:3e:00:00:08 ethertype IPv4 (0x0800), length 100: 10.4.2.2 > 10.4.1.2: ICMP
⇒ echo reply, id 536, seq 297, length 64
```

A 305 Distanz-Vektor Routing mit RIP

Teilaufgabe i)

Terminal auf `router1`

```
# ip link set eth3 down
# ip link set eth4 down
```

Terminal auf `router2`

```
# ip link set eth3 down
```

Terminal auf `router4`

```
# ip link set eth1 down
# ip link set eth4 down
```

Terminal auf `router3`

```
# ip link set eth2 down
# ip link set eth3 down
```

Teilaufgabe ii)

Zunächst wird sicherbestellt, dass auf allen Routen die folgenden Dateien mit dem angegebenen Inhalt existieren.

Datei `/etc/quagga/daemons`

```
1 zebra=yes
2 bgpd=no
3 ospfd=no
4 ospf6d=no
5 ripd=yes
6 ripngd=no
7 isisd=no
8 babeld=no
```

Datei `/etc/quagga/zebra.conf`

```
```

Datei `/etc/quagga/ripd.conf`

```
1 router rip
2   network 10.4.0.0/16
```

Danach wurden vom Gruppenrechner aus die Routing-Dienste gestartet. Sie wurden entlang des Pfades mit 5 s abstand gestartet. Dadurch sind ihre Auswirkungen im mitgeschnittenen Netzverkehr leichter zu erkennen.

Terminal auf `Gruppe04`

```
for i in 1 2 4 3 ; do
        ssh root@router$i systemctl start quagga &
        sleep 5
done
```

Terminal auf `router1`

```
# tcpdump -tttttvi eth2 udp
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
00:00:00.000000 IP (tos 0xc0, ttl 1, id 29434, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.1.route > 224.0.0.9.route:
  1     RIPv2, Request, length: 24, routes: 1 or less
          AFI 0,         0.0.0.0/0 , tag 0x0000, metric: 16, next-hop: self
00:00:00.992428 IP (tos 0xc0, ttl 1, id 29463, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.1.route > 224.0.0.9.route:
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,        10.4.1.0/24, tag 0x0000, metric: 1, next-hop: self
00:00:05.011852 IP (tos 0xc0, ttl 1, id 40191, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.2.route > 224.0.0.9.route:
  2     RIPv2, Request, length: 24, routes: 1 or less
          AFI 0,         0.0.0.0/0 , tag 0x0000, metric: 16, next-hop: self
00:00:05.012044 IP (tos 0xc0, ttl 64, id 50363, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.1.route > 10.4.12.2.route:
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,        10.4.1.0/24, tag 0x0000, metric: 1, next-hop: self
00:00:06.002782 IP (tos 0xc0, ttl 1, id 40306, offset 0, flags [DF], proto UDP (17), length 72)
    10.4.12.2.route > 224.0.0.9.route:
        RIPv2, Response, length: 44, routes: 2 or less
          AFI IPv4,        10.4.2.0/24, tag 0x0000, metric: 1, next-hop: self
          AFI IPv4,       10.4.24.0/24, tag 0x0000, metric: 1, next-hop: self
00:00:10.020472 IP (tos 0xc0, ttl 1, id 40814, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.2.route > 224.0.0.9.route:
  3     RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,       10.4.34.0/24, tag 0x0000, metric: 2, next-hop: self
00:00:16.020406 IP (tos 0xc0, ttl 1, id 40915, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.2.route > 224.0.0.9.route:
  4     RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,        10.4.3.0/24, tag 0x0000, metric: 3, next-hop: self
```

Bei Schritt ① Fragt `reouter1` zunächst vergeblich per Multicast, ob ein anderer Router irgendeine Route für ihn hat. Nach 1 s veröffentlich er dann selbst seine Routen per Multicast. Die Route die in das Netz der Schnitstellte führt, über die die Informationen verschickt werden, wird nicht mitgesendet.

In Schritt ② fragt diesmal `reouter2` per Multicast nach Routen. Er bekommt daraufhin von `reouter1` eine Antwort per Unicast. Sie enthält die Routen, welche `router1` in Schritt ① per Multicast vershcikt hat. Abermals 1 s später schickt dann `reouter2` seine Routen an die Multicastadresse.

In Schritt ③ und ④ teilt `router2` seine von `router3` und `router4` *neu* gelernten Routen allen interessieren per Multicast mit.

Teilaufgabe iii)

Terminal auf `router1`

```
# vtysh -c '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) 10.4.1.0/24     0.0.0.0               1 self              0
R(n) 10.4.2.0/24     10.4.12.2             2 10.4.12.2         0 02:42
R(n) 10.4.3.0/24     10.4.12.2             4 10.4.12.2         0 02:42
C(i) 10.4.12.0/24    0.0.0.0               1 self              0
R(n) 10.4.24.0/24    10.4.12.2             2 10.4.12.2         0 02:42
R(n) 10.4.34.0/24    10.4.12.2             3 10.4.12.2         0 02:42
```

Das *From* Feld zeigt die IP Adresse des Routers, von dem die Route gelernt wurde.

Teilaufgabe iv)

Diese Aufgabe wurde unter der Annahme verrichtet, dass die Verbindungen zwischen den Routern und den PC's nicht relevant seien. Zusätzlich schaffte bessere Übersicht.

Tabellen vor hochfahren der Links router2:

Terminal auf `router1`

```
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) 10.4.12.0/24       0.0.0.0               1 self              0
C(i) 10.4.24.0/24       0.0.0.0               1 self              0
R(n) 10.4.34.0/24       10.4.24.4             2 10.4.24.4         0 02:43
```

r4:

Terminal auf `router1`

```
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
R(n) 10.4.12.0/24       10.4.24.2             2 10.4.24.2         0 02:29
C(i) 10.4.24.0/24       0.0.0.0               1 self              0
C(i) 10.4.34.0/24       0.0.0.0               1 self              0
```

router2 erhält von router1 bescheid über neuen Netzzugang:

Terminal auf `router1`

```
15:17:39.416330 00:16:3e:00:00:09 (oui Unknown) > 01:00:5e:00:00:09 (oui Unknown), ethertype IPv4 (0
⇒x0800), length 66: (tos 0xc0, ttl 1, id 37810, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.12.1.route > 224.0.0.9.route: [bad udp cksum 0xf63f -> 0xed85!]
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,       10.4.13.0/24, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 0a04 0d00 ffff ff00
        0x0010:  0000 0000 0000 0001
```

Terminal auf `router1`

```
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) 10.4.12.0/24       0.0.0.0               1 self              0
R(n) 10.4.13.0/24       10.4.12.1             2 10.4.12.1         0 02:39
C(i) 10.4.24.0/24       0.0.0.0               1 self              0
R(n) 10.4.34.0/24       10.4.24.4             2 10.4.24.4         0 02:40
```

router4 erhält von ron router3 bescheid über neuen Netzzugang:

Terminal auf `router1`

```
15:17:29.045587 00:16:3e:00:00:21 (oui Unknown) > 01:00:5e:00:00:09 (oui Unknown), ethertype IPv4 (0
⇒x0800), length 66: (tos 0xc0, ttl 1, id 48892, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.34.3.route > 224.0.0.9.route: [bad udp cksum 0x0c42 -> 0xd783!]
```

```
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,       10.4.13.0/24, tag 0x0000, metric: 1, next-hop: self
        0x0000:  0202 0000 0002 0000 0a04 0d00 ffff ff00
        0x0010:  0000 0000 0000 0001
```

Terminal auf `router1`

```
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
R(n) 10.4.12.0/24       10.4.24.2             2 10.4.24.2        0 02:43
R(n) 10.4.13.0/24       10.4.34.3             2 10.4.34.3        0 03:00
C(i) 10.4.24.0/24       0.0.0.0               1 self             0
C(i) 10.4.34.0/24       0.0.0.0               1 self             0
```

Der Nachrichtenaustausch von router2 und router4 bezüglich der neuen Routen wurde hier vernachlässigt, weil die Router durch jeweiliges prüfen der Metriken erkennen, dass sie selbst bereits eine kürzere Route verfügen, und den somit entstehenden Traffic verwerfen.

Teilaufgabe v)

Der folgende Traffic wurde von router4 aus über beide relevante Schnittstellen (eth3 und eth2) beobachtet router2

sendet an router4: Achtung, Netz nicht erreichbar:

Terminal auf `router1`

```
12:50:19.349012 00:16:3e:00:00:16 (oui Unknown) > 01:00:5e:00:00:09 (oui Unknown), ethertype IPv4 (0
⇒x0800), length 66: (tos 0xc0, ttl 1, id 43896, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.24.2.route > 224.0.0.9.route: [bad udp cksum 0x0241 -> 0xed75!]
       RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,        10.4.1.0/24, tag 0x0000, metric: 16, next-hop: self
        0x0000:  0202 0000 0002 0000 0a04 0100 ffff ff00
        0x0010:  0000 0000 0000 0010
```

router3 meldet ebenfalls, dass das Netz nicht mehr erreichbar ist :

Terminal auf `router1`

```
12:50:19.348999 00:16:3e:00:00:21 (oui Unknown) > 01:00:5e:00:00:09 (oui Unknown), ethertype IPv4 (0
⇒x0800), length 66: (tos 0xc0, ttl 1, id 58447, offset 0, flags [DF], proto UDP (17), length 52)
    10.4.34.3.route > 224.0.0.9.route: [bad udp cksum 0x0c42 -> 0xe374!]
        RIPv2, Response, length: 24, routes: 1 or less
          AFI IPv4,        10.4.1.0/24, tag 0x0000, metric: 16, next-hop: self
        0x0000:  0202 0000 0002 0000 0a04 0100 ffff ff00
        0x0010:  0000 0000 0000 0010
```

router2 & router3 senden in verkürztem Zeitabstand ihre Routen an router4 (inkl 10.4.1.0/24 mit metric 16), router4 wertet aus und aktualisiert die gültigen Routen. Die Metric für 10.4.1.0/24 wird auf 16 und der Countdown auf 120 Sekunden gesetzt. Nach Ablauf der 120 Sekunden wird die Route aus der Tabelle gelöscht. a) Auszug

RFC1058 As an example, in RIP every gateway that participates in routing sends an update message to all its neighbors once every 30 seconds. Suppose the current route for network N uses gateway G. If we don't hear from G for 180 seconds, we can assume that either the gateway has crashed or the network connecting us to it has become unusable. Thus, we mark the route as invalid. When we hear from another neighbor that has a valid route to N, the valid route will replace the invalid one. Note that we wait for 180 seconds before timing out a route even though we expect to hear from each neighbor every 30 seconds. b) Im Gegenzug zur RFC wird der

Timer nach erhalt der Metric 16 Message auf 120 Sekunden gesetzt. Das Routen-Update erfolgt nach Erkennen eines unerreichbaren Netzes kurzzeitig sekündlich. Auslöser-PDU's sieh oben.

A 306 Link-State Routing mit OSPF

Teilaufgabe i)

Terminal auf router1

```
# ip link set eth3 down
```

Terminal auf router3

```
# ip link set eth2 down
```

Terminal auf router1 / router2 / router3 / router4

```
# systemctl stop quagga
```

Teilaufgabe ii)

Datei /etc/quagga/daemons

```
1 zebra=yes
2 bgpd=no
3 ospfd=yes
4 ospf6d=no
5 ripd=no
6 ripngd=no
7 isisd=no
8 babeld=no
```

Datei TODO

```
1 hostname routerX
2 password kiwifi
3
4 access-list access deny 192.168.0.0/16
5
6 router ospf
7     network 10.4.0.0/16 area 0
8     ospf router-id 10.4.0.X
9 log file /home/root/ospfd.log
```

Terminal auf Gruppe04

```
for i in 1 2 4 3 ; do
        ssh root@router$i systemctl start quagga &
        sleep 300
done
```

Teilaufgabe iii)

Wiederholungen von Paketen haben wir aus der Ausgabe herausgenommen und die stellen kenntlich gemacht.

Terminal auf `router1`

```
00:00:00.000000 IP (tos 0xc0, ttl 1, id 32193, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 44
  1     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
00:00:10.000263 IP (tos 0xc0, ttl 1, id 32195, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 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
[...]
00:00:50.003020 IP (tos 0xc0, ttl 1, id 32203, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 44
  2     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
          Designated Router 10.4.12.1
[...]
```

In der ersten Phase, wo nur ein Router OSPF spricht, ist die unterhaltung auf der Leitung noch recht eintönig. So gibt `router1` per Multicast alle 10 s kund, dass er existiert ①. Nachdem er nach 40 s kein Lebenszeichen von einem andren Router gehört hat, kürt er sich selbst zum *Designated Router* ②, in seinem kleinen Königreich ohne Untertanen.

Terminal auf `router1`

```
00:05:00.025487 IP (tos 0xc0, ttl 1, id 32493, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.2 > 224.0.0.5: OSPFv2, Hello, length 44
  1     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
00:05:10.022656 IP (tos 0xc0, ttl 1, id 32255, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
  2     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
          Designated Router 10.4.12.1
          Neighbor List:
            192.168.0.12
00:05:10.023509 IP (tos 0xc0, ttl 1, id 32495, offset 0, flags [none], proto OSPF (89), length 52)
    10.4.12.2 > 10.4.12.1: OSPFv2, Database Description, length 32
  3     Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0)
        Options [External], DD Flags [Init, More, Master], MTU: 1500, Sequence: 0x5859039f
00:05:10.023817 IP (tos 0xc0, ttl 1, id 32256, offset 0, flags [none], proto OSPF (89), length 52)
    10.4.12.1 > 10.4.12.2: 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: 0x5859027c
00:05:10.023912 IP (tos 0xc0, ttl 1, id 32257, offset 0, flags [none], proto OSPF (89), length 72)
    10.4.12.1 > 10.4.12.2: OSPFv2, Database Description, length 52
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
        Options [External], DD Flags [none], MTU: 1500, Sequence: 0x5859039f
          Advertising Router 192.168.0.7, seq 0x80000005, age 270s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
00:05:10.024361 IP (tos 0xc0, ttl 1, id 32496, offset 0, flags [none], proto OSPF (89), length 72)
    10.4.12.2 > 10.4.12.1: OSPFv2, Database Description, length 52
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0)
        Options [External], DD Flags [Master], MTU: 1500, Sequence: 0x585903a0
          Advertising Router 192.168.0.12, seq 0x80000005, age 0s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
00:05:10.024531 IP (tos 0xc0, ttl 1, id 32258, offset 0, flags [none], proto OSPF (89), length 52)
    10.4.12.1 > 10.4.12.2: 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: 0x585903a0
00:05:10.024612 IP (tos 0xc0, ttl 1, id 32259, offset 0, flags [none], proto OSPF (89), length 56)
    10.4.12.1 > 10.4.12.2: OSPFv2, LS-Request, length 36
  4     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
00:05:10.025137 IP (tos 0xc0, ttl 1, id 32497, offset 0, flags [none], proto OSPF (89), length 56)
    10.4.12.2 > 10.4.12.1: OSPFv2, LS-Request, length 36
```

```
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0)
          Advertising Router: 192.168.0.7, Router LSA (1), LSA-ID: 192.168.0.7
00:05:10.025155 IP (tos 0xc0, ttl 1, id 32498, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000005, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Stub Network: 10.4.12.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:10.025502 IP (tos 0xc0, ttl 1, id 32260, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.1 > 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 271s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.1.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Stub Network: 10.4.12.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:10.025614 IP (tos 0xc0, ttl 1, id 32261, offset 0, flags [none], proto OSPF (89), length 128)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Update, length 108
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0), 2 LSAs
          LSA #1
          Advertising Router 192.168.0.7, seq 0x80000006, age 1s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.1.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.1
                topology default (0), metric 10
          LSA #2
          Advertising Router 192.168.0.7, seq 0x80000001, age 1s, length 12
            Network LSA (2), LSA-ID: 10.4.12.1
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.7
              192.168.0.12
00:05:10.025779 IP (tos 0xc0, ttl 1, id 32500, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:05:10.026325 IP (tos 0xc0, ttl 1, id 32502, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000006, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:10.262916 IP (tos 0xc0, ttl 1, id 32262, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.12, seq 0x80000005, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
00:05:11.024556 IP (tos 0xc0, ttl 1, id 32503, offset 0, flags [none], proto OSPF (89), length 84)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Ack, length 64
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.7, seq 0x80000005, age 271s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
          Advertising Router 192.168.0.7, seq 0x80000001, age 1s, length 12
```

```
            Network LSA (2), LSA-ID: 10.4.12.1
            Options: [External]
00:05:20.023594 IP (tos 0xc0, ttl 1, id 32264, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:05:20.024969 IP (tos 0xc0, ttl 1, id 32265, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.1 > 10.4.12.2: 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 0x80000006, age 10s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.1.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.1
                topology default (0), metric 10
00:05:20.025908 IP (tos 0xc0, ttl 1, id 32505, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:05:20.025929 IP (tos 0xc0, ttl 1, id 32506, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 10.4.12.1: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000006, age 10s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:20.033067 IP (tos 0xc0, ttl 1, id 32508, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.2 > 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 0x80000006, age 10s, length 28
            Router LSA (1), LSA-ID: 192.168.0.7
            Options: [External]
00:05:20.271167 IP (tos 0xc0, ttl 1, id 32266, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.12, seq 0x80000006, age 10s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
00:05:30.023770 IP (tos 0xc0, ttl 1, id 32268, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:05:30.025905 IP (tos 0xc0, ttl 1, id 32510, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:05:40.023997 IP (tos 0xc0, ttl 1, id 32512, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000007, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
```

```
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:40.024019 IP (tos 0xc0, ttl 1, id 32513, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000008, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:40.024267 IP (tos 0xc0, ttl 1, id 32270, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:05:40.025823 IP (tos 0xc0, ttl 1, id 32515, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:05:40.290449 IP (tos 0xc0, ttl 1, id 32271, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.12, seq 0x80000007, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
00:05:45.027797 IP (tos 0xc0, ttl 1, id 32517, offset 0, flags [none], proto OSPF (89), length 108)
    10.4.12.2 > 10.4.12.1: OSPFv2, LS-Update, length 88
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 1 LSA
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000008, age 6s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
                topology default (0), metric 10
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:05:45.295588 IP (tos 0xc0, ttl 1, id 32272, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.12, seq 0x80000008, age 6s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
00:05:50.025050 IP (tos 0xc0, ttl 1, id 32274, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:05:50.026122 IP (tos 0xc0, ttl 1, id 32519, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
[...]
```

A Wild *Hello Package* Appears ①. Es stammt von `router2`. Das `router1` nun endlich einen Freund gefunden hat, verbreitet es ab sofort in seinen Hello Paketen ②.

Sobald `router2` das erste davon empfängt starten die beiden Nachbarn einen *Database Description* austausch

per Unicast ③. Die ersten beiden Pakete dienen zum bestimmen des Masters. Aufgrund der höheren Router-ID von `router2` wird dieser Master. Der unterlegene `router1` muss seine *Database Description* Anfrage noch einmal ohne das *Master Flag* stellen. Da der Master in vorauseilendem Gehorsam jedoch seine Antwort schon vorherg geschickt hat ist die Anfrage des Slaves das letzte Paket von Schritt ③.

Nachdem beide Router ihre Datenbanken ausgetauscht haben stellen sie fest, dass sie nicht viel über den anderen wissen, sich haben sich ja auch gerade erst kennen gelernt. Daher fordern sie in Schritt ④ gegenseitig detailiertere Informationen an.

Terminal auf `router1`

```
00:10:10.034781 IP (tos 0xc0, ttl 1, id 32604, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 192.168.0.22, seq 0x80000004, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.24.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Stub Network: 10.4.34.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:10:10.034800 IP (tos 0xc0, ttl 1, id 32605, offset 0, flags [none], proto OSPF (89), length 140)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 120
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 2 LSAs
          LSA #1
          Advertising Router 192.168.0.12, seq 0x80000009, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.2.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.12.1, Interface Address: 10.4.12.2
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.24.2, Interface Address: 10.4.24.2
                topology default (0), metric 10
          LSA #2
          Advertising Router 192.168.0.12, seq 0x80000001, age 1s, length 12
            Network LSA (2), LSA-ID: 10.4.24.2
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.12
              192.168.0.22
00:10:10.545715 IP (tos 0xc0, ttl 1, id 32327, offset 0, flags [none], proto OSPF (89), length 104)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 84
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.22, seq 0x80000004, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000009, age 1s, length 40
            Router LSA (1), LSA-ID: 192.168.0.12
            Options: [External]
          Advertising Router 192.168.0.12, seq 0x80000001, age 1s, length 12
            Network LSA (2), LSA-ID: 10.4.24.2
            Options: [External]
00:10:20.032680 IP (tos 0xc0, ttl 1, id 32329, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:10:20.032927 IP (tos 0xc0, ttl 1, id 32609, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:10:20.034677 IP (tos 0xc0, ttl 1, id 32612, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 192.168.0.22, seq 0x80000005, age 11s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
```

```
            Router LSA Options: [none]
              Neighbor Network-ID: 10.4.24.2, Interface Address: 10.4.24.4
                topology default (0), metric 10
              Stub Network: 10.4.34.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:10:20.555040 IP (tos 0xc0, ttl 1, id 32330, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.22, seq 0x80000005, age 11s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
00:10:30.032823 IP (tos 0xc0, ttl 1, id 32332, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:10:30.032998 IP (tos 0xc0, ttl 1, id 32615, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:10:40.033388 IP (tos 0xc0, ttl 1, id 32334, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:10:40.033787 IP (tos 0xc0, ttl 1, id 32618, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:10:40.033813 IP (tos 0xc0, ttl 1, id 32619, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 192.168.0.22, seq 0x80000006, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
            Router LSA Options: [none]
              Neighbor Network-ID: 10.4.24.2, Interface Address: 10.4.24.4
                topology default (0), metric 10
              Stub Network: 10.4.34.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:10:40.574626 IP (tos 0xc0, ttl 1, id 32335, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.22, seq 0x80000006, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
00:10:50.034078 IP (tos 0xc0, ttl 1, id 32337, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:10:50.034517 IP (tos 0xc0, ttl 1, id 32623, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
[...]
```

Und noch mehr Pakete mit Updates, Update-Bestätigungen und *Hello*s.

Terminal auf `router1`

```
00:15:00.052574 IP (tos 0xc0, ttl 1, id 32700, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 0x80000003, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.3.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Stub Network: 10.4.34.0, Mask: 255.255.255.0
                topology default (0), metric 10
00:15:00.052587 IP (tos 0xc0, ttl 1, id 32701, offset 0, flags [none], proto OSPF (89), length 128)
    10.4.12.2 > 224.0.0.5: OSPFv2, LS-Update, length 108
        Router-ID 192.168.0.12, Backbone Area, Authentication Type: none (0), 2 LSAs
          LSA #1
          Advertising Router 192.168.0.22, seq 0x80000007, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
            Router LSA Options: [none]
              Neighbor Network-ID: 10.4.24.2, Interface Address: 10.4.24.4
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.34.4, Interface Address: 10.4.34.4
                topology default (0), metric 10
          LSA #2
          Advertising Router 192.168.0.22, seq 0x80000001, age 2s, length 12
            Network LSA (2), LSA-ID: 10.4.34.4
            Options: [External]
            Mask 255.255.255.0
            Connected Routers:
              192.168.0.17
              192.168.0.22
00:15:00.825576 IP (tos 0xc0, ttl 1, id 32388, offset 0, flags [none], proto OSPF (89), length 104)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 84
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.17, seq 0x80000003, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
          Advertising Router 192.168.0.22, seq 0x80000007, age 2s, length 28
            Router LSA (1), LSA-ID: 192.168.0.22
            Options: [External]
          Advertising Router 192.168.0.22, seq 0x80000001, age 2s, length 12
            Network LSA (2), LSA-ID: 10.4.34.4
            Options: [External]
00:15:10.031008 IP (tos 0xc0, ttl 1, id 32703, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 0x80000005, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.3.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.34.4, Interface Address: 10.4.34.3
                topology default (0), metric 10
00:15:10.044115 IP (tos 0xc0, ttl 1, id 32390, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:15:10.044885 IP (tos 0xc0, ttl 1, id 32705, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:15:10.834680 IP (tos 0xc0, ttl 1, id 32391, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.17, seq 0x80000005, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
00:15:20.044423 IP (tos 0xc0, ttl 1, id 32393, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:15:20.045080 IP (tos 0xc0, ttl 1, id 32709, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:15:30.044604 IP (tos 0xc0, ttl 1, id 32395, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:15:30.045135 IP (tos 0xc0, ttl 1, id 32712, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:15:40.039054 IP (tos 0xc0, ttl 1, id 32714, offset 0, flags [none], proto OSPF (89), length 96)
    10.4.12.2 > 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 0x80000006, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
            Router LSA Options: [none]
              Stub Network: 10.4.3.0, Mask: 255.255.255.0
                topology default (0), metric 10
              Neighbor Network-ID: 10.4.34.4, Interface Address: 10.4.34.3
                topology default (0), metric 10
00:15:40.044710 IP (tos 0xc0, ttl 1, id 32397, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:15:40.045177 IP (tos 0xc0, ttl 1, id 32716, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
00:15:40.863083 IP (tos 0xc0, ttl 1, id 32398, offset 0, flags [none], proto OSPF (89), length 64)
    10.4.12.1 > 224.0.0.5: OSPFv2, LS-Ack, length 44
        Router-ID 192.168.0.7, Backbone Area, Authentication Type: none (0)
          Advertising Router 192.168.0.17, seq 0x80000006, age 3s, length 28
            Router LSA (1), LSA-ID: 192.168.0.17
            Options: [External]
00:15:50.045001 IP (tos 0xc0, ttl 1, id 32400, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.1 > 224.0.0.5: OSPFv2, Hello, length 48
        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
          Designated Router 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.12
00:15:50.045587 IP (tos 0xc0, ttl 1, id 32720, offset 0, flags [none], proto OSPF (89), length 68)
    10.4.12.2 > 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 10.4.12.1, Backup Designated Router 10.4.12.2
          Neighbor List:
            192.168.0.7
[...]
```

Und noch mehr Pakete mit Updates, Update-Bestätigungen und *Hello*s.

A 307 BGP

Teilaufgabe i)

Zuerst musst die Konfiguration der SChnittstelle herausgefunden werden. Anhand der folgenden Ausgabe kann man erkennen, dass ein Rechner mit der IP-Adresse 10.100.0.254 die IP-Adresse 10.100.0.4 sucht.

Terminal auf `router1`

```
# tcpdump -ttttti eth4
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth4, link-type EN10MB (Ethernet), capture size 262144 bytes
00:00:00.000000 ARP, Request who-has 10.100.0.4 tell 10.100.0.254, length 28
```