

IT-HTL Ybbs Schulring 6

Höhere Lehranstalt für Informationstechnologien

Laboratorium für Netzwerktechnik

Übung 12 – Protokoll 4/ Routing Information Protocol

Autor: Martin Dallinger Gruppe: 2

Jhg. /Klasse: 3BHIT

abgegeben am: 28.2.2019

Inhalt

[Aufgabenstellung 3](#_bookmark0)

[Übungsaufbau 3](#_bookmark1)

[Erläuterung zu RIP 4](#_bookmark2)

[Einrichten des Netzwerks 4](#_bookmark3)

[Konfiguration der Interfaces 4](#_bookmark4)

[Test durch ping 4](#_bookmark5)

[Konfiguration des Routings 5](#_bookmark6)

[Netzbereich 5](#_bookmark7)

[RIPv1 Grundkonfiguration 5](#_bookmark8)

[Debugging 6](#_bookmark9)

[RIPv1 Analyse mit Wireshark 8](#_bookmark10)

[RIPv2 9](#_bookmark11)

[Debugging 9](#_bookmark12)

[RIPv2 Analyse mit Wireshark 11](#_bookmark13)

[Unterschied RIPv1/RIPv2 12](#_bookmark14)

[Routing Tabelle 12](#_bookmark15)

[Zusätzliche Fragen 13](#_bookmark16)

[Verwendete Geräte, Software, Unterlagen 13](#_bookmark17)

[Weitere Anmerkungen zum Übungsverlauf + Erkenntnisse 14](#_bookmark18)

[Quellen 14](#_bookmark19)

# Aufgabenstellung

Es soll ein LAN-Netz mit 4 Routern aufgebaut werden (mit loopback interfaces), in dem keine statischen oder default-Routen gesetzt werden, sondern dynamisches Routing via RIP aktiv ist. Die Unterschiede zwischen RIPv1 und RIPv2 sollen mithilfe des Router Debugging-Modus und der Verwendung des Netzwerksniffers Wireshark analysiert werden.

Folgende Fragen müssen ebenfalls beantwortet werden:

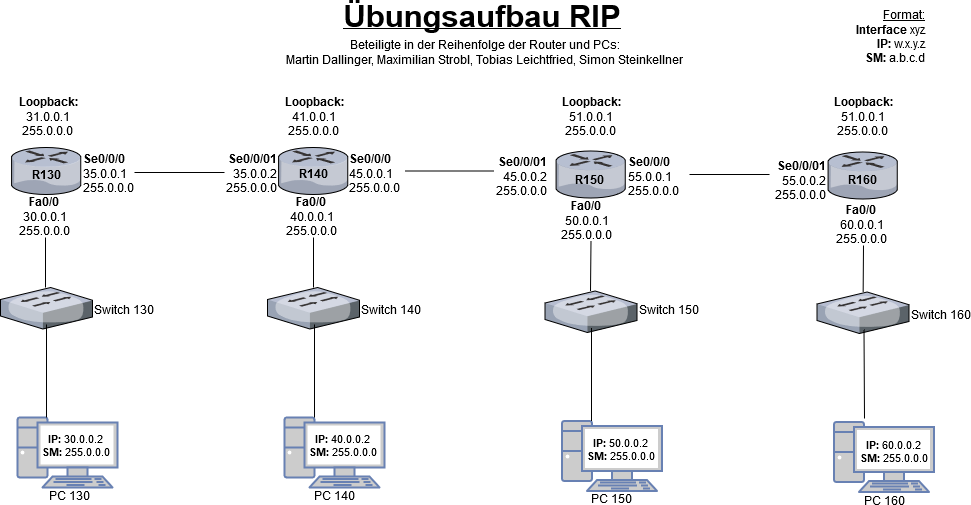
* Zu welcher Kategorie von Routingprotokollen zählt RIP?
* Welche Werte bilden bei RIP die Metrik?
* Welche „Administrative Distanz“ hat RIP per Default auf Cisco Routern?

# Übungsaufbau

In folgender Abbildung können alle verwendeten Geräte in Kombination mit deren Subnetzmasken und IP Adresse an den jeweiligen Interfaces eingesehen werden. Da sich dieses Protokoll lediglich auf das Routing selbst beschränkt, wird der Netzaufbau selbst nur vage erläutert.

Die Captures und Konfiguration wurden dabei auf dem PC 130 und dem Router 130 von mir (Martin Dallinger) durchgeführt.

Das Netz wurde folgendermaßen aufgebaut: Die IPs der Netze beziehen sich auf die Platznummer minus 100, da Klasse A Netze verwendet werden. Lokal (Netz zwischen Router und PC) bekommen die Router im 4. IPv4 Oktett immer die Ziffer .1 und die PCs .2 . Die Netze zwischen zwei Routern werden aus dem Durchschnitt der beiden Platznummern – 100 ermittelt. Weiters bekommt bei Netzen zwischen Routern der Router mit der niedrigeren Platznummer im 4. Oktett der IPv4 Adresse .1 und der mit der höheren Platznummer .2 . Das Loopback-Netz jedes Routers ist die Platznummer – 100 + 1 und der Router selbst nimmt jeweils die 1. belegbare IP ein.



*Abbildung 1 Netzaufbau*

# Erläuterung zu RIP

Das Routing Information Protocol dient dazu, um Routen automatisch im Netzwerk zu verteilen, wobei vorausgesetzt wird, dass jedem Router mitgeteilt wird, welche Netze er weitervermitteln soll. RIP dient auch dazu die wenigsten Hops zwischen Source und Destination zu errechnen und legt somit für jeden Eintrag eine Metrik fest – die jeweiligen Einträge für die Routen werden in den Routingtabellen gespeichert. Grundsätzlich schickt ein Router periodisch Routeninformationspakete aus. Verwendet man für Routing RIP, so unterscheidet man zwischen zwei Versionen von RIP: RIPv1 (auch RIP) und RIPv2 – der Unterschied zwischen den beiden Protokollsversionen ist, dass RIPv1 keine Subnetzinformationen unterstütz und somit nur Klassenhaft subnetten kann. Im Protokoll werden beide Versionen behandelt.

*Anmerkung: In* [*RFC 2080*](https://tools.ietf.org/html/rfc2080) *wurde eine dritte Version von RIP namens RIPng, diese wird jedoch in diesem Protokoll nicht behandelt, da der wesentliche unterschied zu RIPv2 die IPv6-Unterstützung ist.*

# Einrichten des Netzwerks

## Konfiguration der Interfaces

Zunächst wurde im Konfigurationsmodus mit interface loopback 0 und no shutdown ein neues Loopback Interface angelegt. Diesem wurde die IP Adresse 31.0.0.1 vergeben, da diese mithilfe des Netzplans vereinbart wurde. Anschließend wurden alle anderen Schnittstellen des Routers ebenfalls nach dem Netzplan konfiguriert (das serielle Interface und die beiden Fast-Ethernet- Interfaces). Bei der seriellen Schnittstelle wurde darauf geachtet, dass die Clock Rate gesetzt wurde, da ansonsten die Kommunikation nicht möglich wäre.

Generell wurden die Interfaces folgender maßen konfiguriert:

interface <Beispiel>

ip adress xx.0.0.yy 255.0.0.0 no shutdown

## Test durch ping

Es sollte nun möglich sein vom konfigurierten Router einen benachbarten Router zu pingen, da beide Schnittstellen konfiguriert wurden und für Netze die ein Router ohnehin kennt kein Routingprotokoll notwendig ist. In folgendem Konsolenausschnitt ist sichtbar, dass der Ping erfolgreich ist:

R130#ping 35.0.0.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 35.0.0.2, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max

= 1/2/4 ms

## Konfiguration des Routings

Da alle Interfaces nun fertig eingerichtet wurden, kann das eigentliche Routing beginnen. In sehr kleinen Netzwerken wie diesem werden normalerweise statische Routen bzw.

Defaultrouten verwendet. Nimmt man jedoch an, dass dieses Netz auch größer sein könnte, so macht dynamisches Routing durchaus Sinn.

## Netzbereich

Dabei ist zu beachten, dass RIPv1 CIDR (Classless Interdomain Routing) unterstützt. Somit müssen folgende Netzbereiche eingehalten werden:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Klasse A** | **Klasse B** | **Klasse C** |
| **1. Oktett der IP** | 0-127 | 128-191 | 192-223 |
| **Binär 1. Oktett** | 0xxx xxxx | 10xx xxxx | 110x xxxx |
| **Subnetzmaske** | 255.0.0.0 | 255.255.0.0 | 255.255.255.0 |

*Anmerkung: x steht für einen variablen Binärwert*

In diesem Beispiel wurde ein Klasse A Netz verwendet, da eine Subnetzmaske von /8 angestrebt wurde.

# RIPv1 Grundkonfiguration

Um nun tatsächlich alle Router und die mit dem Router verbundenen Netzen erreichen zu können wird nun RIPv1 konfiguriert:

Um in den RIP Konfigurationsmodus zu gelangen wird zunächst im Router Konfigurationsmodus der Befehl router rip eingegeben. Anschließend können schon alle dem Router bekannten Netze eingetragen werden:

R130(config-if)#router rip R130(config-router)#network 30.0.0.0

R130(config-router)#network 35.0.0.0

R130(config-router)#network 31.0.0.0

Im Router-Konfigurationsmodus können folgende weitere Befehle eingegeben werden, um die RIPv1 Konfiguration zu optimieren:

R130(config-router)#passive-interface fa0/1 R130(config-router)#no auto-summary

Wie im Konsolenausschnitt betrachtet werden kann, kann ein passive-interface definiert werden, auf dem keine RIP-Anfragen ausgesendet werden. ~ passive-interface fa0/1 Weiters wird auch das automatische Zusammenfassen von Netzen durch den Router ausgeschaltet ~ no auto-summary, was in diesem Fall nur begrenzt Sinn ergibt. Würde man jedoch die Netzbereiche (und dadurch die Subnetzmasken) nicht mit dem ersten Oktett der IP assoziieren, so würde RIP wenn Netze eines bestimmten Übernetzes bekannt sind weiterleiten, dass der Router über das gesamte Übernetz verfügt, was keinen Sinn ergibt.

Wird die eben durchgeführte Konfiguration auf jedem Router vorgenommen, so kann jeder PC jeden anderen pingen. Als beweis wird der PC am anderen Ende des Netzwerks gepingt. Da dieser Ping erfolgt kann angenommen werden, dass RIPv1 funktioniert:

R130#ping 55.0.0.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 55.0.0.2, timeout is 2 seconds:

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

## Debugging

Der Output des Debuggers wird im Normalfall nur an das lokale Terminal weitergeleitet, darum muss der Terminaloutput auch an die Telnet-Session weitergeleitet werden. Um dies zu ermöglichen wird der Befehl terminal monitor im privilegierten Modus eingegeben.

Um nun Debugging selbst zu aktivieren und auf RIP zu spezifizieren wird im privilegierten Modus der Befehl debug ip rip in die Konsole eingegeben. Würde man RIP nicht spezifizieren, sondern debug ip all eingeben, so würden so viele Zeichen gesendet werden, dass man den Debugging-Modus nicht mehr stoppen kann. Das ist also der Grund, warum das Debugging auf RIP eingeschränkt wird.

Der Debug-Modus liefert somit folgenden Output:

|  |  |  |  |
| --- | --- | --- | --- |
| \*Feb | 27 | 14:30:19.114: | RIP: sending v1 update to 255.255.255.255 via Loopback0 (31.0.0.1) |
| \*Feb | 27 | 14:30:19.114: | RIP: build update entries |
| \*Feb | 27 | 14:30:19.114: | network 30.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:19.114: | network 35.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:19.114: | network 40.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:19.114: | network 41.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:19.114: | network 45.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:19.114: | network 50.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:19.114: | network 51.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:19.114: | network 55.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:19.114: | network 61.0.0.0 metric 4 |
| \*Feb | 27 | 14:30:21.586: | RIP: sending v1 update to 255.255.255.255 via Serial0/0/0 (35.0.0.1) |
| \*Feb | 27 | 14:30:21.586: | RIP: build update entries |
| \*Feb | 27 | 14:30:21.586: | network 30.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:21.586: | network 31.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:31.618: | RIP: received v1 update from 35.0.0.2 on Serial0/0/0 |
| \*Feb | 27 | 14:30:31.618: | 40.0.0.0 in 1 hops |
| \*Feb | 27 | 14:30:31.618: | 41.0.0.0 in 1 hops |
| \*Feb | 27 | 14:30:31.618: | 45.0.0.0 in 1 hops |
| \*Feb | 27 | 14:30:31.618: | 50.0.0.0 in 2 hops |
| \*Feb | 27 | 14:30:31.618: | 51.0.0.0 in 2 hops |
| \*Feb | 27 | 14:30:31.618: | 55.0.0.0 in 2 hops |
| \*Feb | 27 | 14:30:31.618: | 61.0.0.0 in 3 hops |
| \*Feb | 27 | 14:30:41.282: | RIP: sending v1 update to 255.255.255.255 via FastEthernet0/0 (30.0.0.1) |
| \*Feb | 27 | 14:30:41.282: | RIP: build update entries |
| \*Feb | 27 | 14:30:41.282: | network 31.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:41.282: | network 35.0.0.0 metric 1 |
| \*Feb | 27 | 14:30:41.282: | network 40.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:41.282: | network 41.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:41.282: | network 45.0.0.0 metric 2 |
| \*Feb | 27 | 14:30:41.282: | network 50.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:41.282: | network 51.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:41.282: | network 55.0.0.0 metric 3 |
| \*Feb | 27 | 14:30:41.282: | network 61.0.0.0 metric 4 |

Man kann erkennen, dass alle anderen Netze von anderen Routern weitergeleitet wurden und dass eine Metrik für jedes Netz eingetragen wurde. Weiters kann erkannt werden auf welchem interface wann RIP Updates empfangen werden und wann auf welchem Interface Updates gesendet werden. Da alle vorhin konfigurierten Netze aufgelistet werden ist die Funktion von RIP bewiesen.

Zusätzlich kann erkannt werden, dass RIP Updates Broadcastpakete sind.

*Nach dem Debugging kann der monitor Modus wieder mit terminal no monitor*

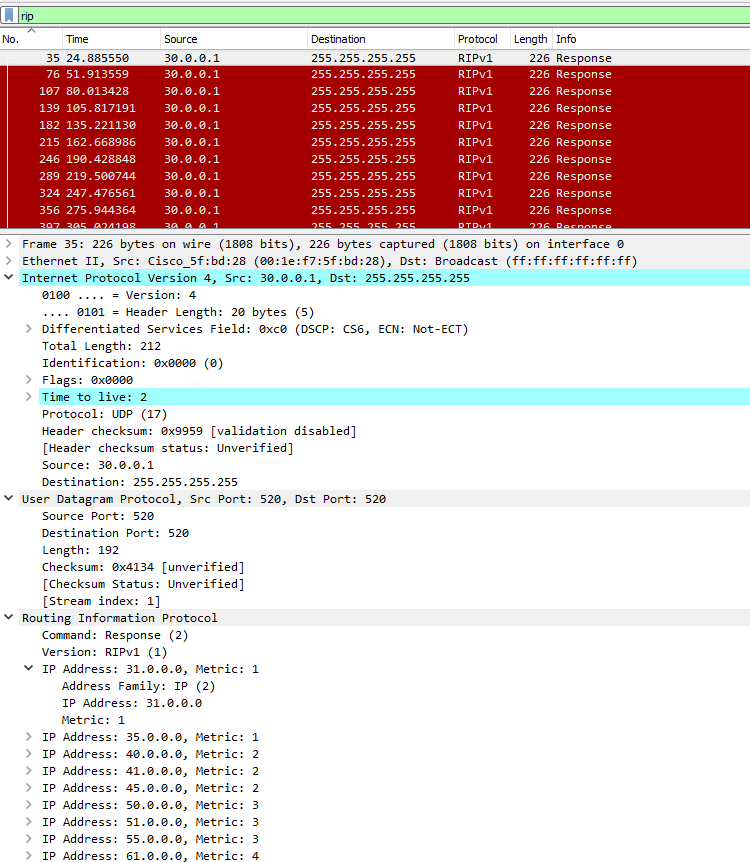
*ausgeschaltet werden.*

**Anmerkung zum Ablauf der Übung**

Das Netzwerk 60.0.0.0 existiert nicht, weil es für den entsprechenden Übungspartner keinen Sinn gemacht hätte dieses Netzwerk hinzuzufügen, da er durch ein Firewall- Konfigurationsproblem keine RIP-Pakete auf seinem Rechner loggen konnte und er über sein eigenes Gerät den Router konfigurierte. Somit symbolisiert sein virtuelles Interface 61.0.0.0 ebenfalls das Netz 60.0.0.0.

# RIPv1 Analyse mit Wireshark

Da dem Router erlaubt wurde die Pakete auch an fa0/0 zu senden, welches das Netzwerk ist, in dem sich auch der PC befindet, kann der PC auch mithilfe eines Netzwerksniffers (in diesem Fall Wireshark) die Pakete analysieren, indem man den Filter auf „rip“ setzt:



*Abbildung 2 RIPv1 Wireshark output*

Es fällt auf, das RIPv1 Pakete Broadcast sind und auf IP und UDP basieren. Somit kann man sagen, dass RIPv1 ein Layer 7 Protokoll ist.

Weiters werden die Pakete von den Routern als Broadcasts und nicht als Multicasts ausgesendet und um die Reichweite der Pakete einzudämmen wird die „Time to live“ auf 2 gesetzt. Dies ist möglich, da ein Router seine gesamte Netzwerktabelle (enthält auch die empfangenen Routen) an seine Interfaces sendet, woran möglicherweise weitere Router angeschlossen sind die wiederum die bekannten Routen zu den eigenen Netzen hinzufügen usw..

Der trivialste erkennbare Fakt ist jedoch, dass RIPv1 keine Subnetzmasken überträgt, sondern lediglich Classful subnetting unterstützt, was auch im Sniffer durch das Fehlen der jeweiligen Subnetzmasken für die Netze erkannt werden kann.

Zusätzlich zu den bereits genannten Fakten kann erkannt werden, dass RIPv1 eine Checksumme verwendet, um die Integrität sicherzustellen und auf Port 520 laufen und dass keine Möglichkeiten zur Authentifizierung vorgesehen sind – somit sind Angriffe auf das Routing des Netzwerks leicht möglich.

Die Metrik (auch Kosten) solle den Aufwand, um ein bestimmtes Netz zu erreichen in Form einer Zahl darstellen. Dadurch muss für jedes Netzwerk eine Metrik festgelegt werden, die in RIP nur durch die Hop-count (die Anzahl der Router zwischen Netzen) bestimmt wird. Bei optimierten Routingprotokollen bestimmt die Metrik auch Paketverlust, Datendurchsatz und weitere Netzfaktoren. Bei RIP werden Routingeinträge auf maximal 16 Hops beschränkt, um in großen Netzwerken nicht zu viele Einträge zu erzeugen.

# RIPv2

Um RIPv2 zu aktivieren wird im Konfigurationsmodus der Befehl router rip eingegeben, um in den RIP-Konfigurationsmodus zu gelangen. Nun wird der Befehl version 2 ausgeführt werden, um zu RIPv2 zu wechseln. Wurden bereits Netzwerke für RIP bekanntgegeben, so müssen diese nicht mehr angegeben werden – die für RIPv1 konfigurierten Netzwerke werden somit in RIPv2 übernommen:

R130(config)#router rip R130(config-router)#version 2

## Debugging

Aktiviert man nun wieder das RIP debugging, so liefert der Router über Telnet (so lange terminal monitor noch aktiviert ist) so bekommt man mit, dass RIP Pakete nun keine Broadcasts, sondern Multicasts sind und dass zu den jeweiligen Netzwerken auch Subnetzmasken eingetragen werden:

|  |  |
| --- | --- |
| \*Mar | 6 14:15:20.502: RIP: sending v2 update to **224.0.0.9** via Loopback0 (31.0.0.1) |
| \*Mar | 6 14:15:20.502: RIP: build update entries |
| \*Mar | 6 14:15:20.502: 30.0.0.0**/8** via 0.0.0.0, metric 1, tag 0 |
| \*Mar | 6 14:15:20.502: 35.0.0.0/8 via 0.0.0.0, metric 1, tag 0 |
| \*Mar | 6 14:15:20.502: 40.0.0.0/8 via 0.0.0.0, metric 2, tag 0 |
| \*Mar | 6 14:15:20.502: 45.0.0.0/8 via 0.0.0.0, metric 2, tag 0 |
| \*Mar | 6 14:15:20.502: 50.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:20.502: 51.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:20.502: 55.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:20.502: 60.0.0.0/8 via 0.0.0.0, metric 4, tag 0 |
| \*Mar | 6 14:15:20.502: 61.0.0.0/8 via 0.0.0.0, metric 4, tag 0 |
| \*Mar | 6 14:15:20.502: RIP: ignored v2 packet from 31.0.0.1 (sourced from one of our addresses) |
| \*Mar | 6 14:15:20.970: RIP: ignored v2 update from bad source 40.0.0.1 on FastEthernet0/0 |
| \*Mar | 6 14:15:33.134: RIP: received v2 update from 35.0.0.2 on Serial0/0/0 |
| \*Mar | 6 14:15:33.134: 40.0.0.0/8 via 0.0.0.0 in 1 hops |
| \*Mar | 6 14:15:33.134: 45.0.0.0/8 via 0.0.0.0 in 1 hops |
| \*Mar | 6 14:15:33.134: 50.0.0.0/8 via 0.0.0.0 in 2 hops |
| \*Mar | 6 14:15:33.134: 51.0.0.0/8 via 0.0.0.0 in 2 hops |
| \*Mar | 6 14:15:33.134: 55.0.0.0/8 via 0.0.0.0 in 2 hops |
| \*Mar | 6 14:15:33.134: 60.0.0.0/8 via 0.0.0.0 in 3 hops |
| \*Mar | 6 14:15:33.134: 61.0.0.0/8 via 0.0.0.0 in 3 hops |
| \*Mar | 6 14:15:38.174: RIP: sending v2 update to **224.0.0.9** via FastEthernet0/0 (30.0.0.1) |
| \*Mar | 6 14:15:38.174: RIP: build update entries |
| \*Mar | 6 14:15:38.174: 31.0.0.0/8 via 0.0.0.0, metric 1, tag 0 |
| \*Mar | 6 14:15:38.174: 35.0.0.0/8 via 0.0.0.0, metric 1, tag 0 |
| \*Mar | 6 14:15:38.174: 40.0.0.0/8 via 0.0.0.0, metric 2, tag 0 |
| \*Mar | 6 14:15:38.174: 45.0.0.0/8 via 0.0.0.0, metric 2, tag 0 |
| \*Mar | 6 14:15:38.174: 50.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:38.174: 51.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:38.174: 55.0.0.0/8 via 0.0.0.0, metric 3, tag 0 |
| \*Mar | 6 14:15:38.174: 60.0.0.0/8 via 0.0.0.0, metric 4, tag 0 |
| \*Mar | 6 14:15:38.174: 61.0.0.0/8 via 0.0.0.0, metric 4, tag 0 |
| \*Mar | 6 14:15:38.238: RIP: sending v2 update to 224.0.0.9 via Serial0/0/0 (35.0.0.1) |
| \*Mar | 6 14:15:38.238: RIP: build update entries |
| \*Mar | 6 14:15:38.238: 30.0.0.0/8 via 0.0.0.0, metric 1, tag 0 |
| \*Mar | 6 14:15:38.238: 31.0.0.0/8 via 0.0.0.0, metric 1, tag 0 |

Diese Updates finden wie bei RIPv1 periodisch etwa im 30 Sekundentakt statt. Die geloggten Zieladressen für die Netzwerke betragen jeweils 0.0.0.0. Dies lässt sich mithilfe des RFC 1723 erklären:

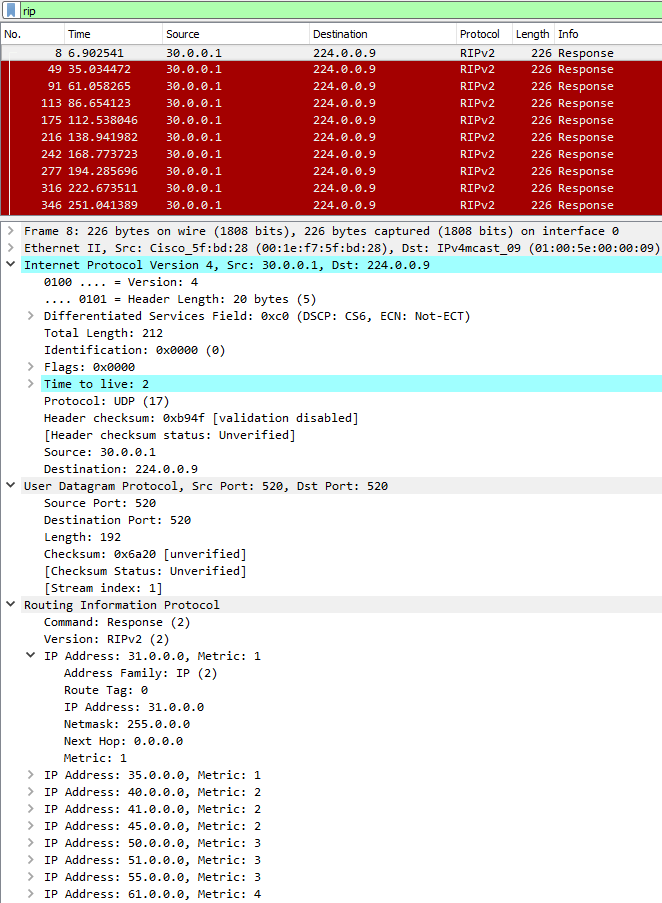
*“The immediate next hop IP address to which packets to the destination specified by this route entry should be forwarded. Specifying a value of 0.0.0.0 in this field indicates that routing should be via*

*the originator of the RIP advertisement. An address specified as a next hop must, per force, be directly reachable on the logical subnet over which the advertisement is made.” – RFC 1723*

Kurz gesagt bedeutet dies, dass der Absender des RIP Pakets als next hop für das jeweilige Netz in die Routing Tabelle eingetragen wird.

## RIPv2 Analyse mit Wireshark

Filtert man nun in Wireshark ebenfalls nach „rip“, so erscheinen nun die RIPv2 Pakete, welche auch der PC empfangen konnte (also die von Router 130). Sofort fällt auf, dass, wie auch im Debug-Modus die Destination-IP nun ein Multicast und kein Broadcast ist und dass in der Spalte Protocol nun RIPv2 angezeigt wird:



*Abbildung 3 Wireshark RIPv2 output*

## Unterschied RIPv1/RIPv2

In RIPv2 wird weiterhin IP mit UDP auf Port 520 verwendet, die TTL ist ident und die Erreichbarkeitsgrenze von 15 Hops bleibt ebenfalls erhalten.

Die ausschlaggebenden Unterschiede zu RIPv1 sind jedoch die variable Subnetzmaske, die für die jeweiligen Netzwerke mitübertragen wird, um classless Routing zu ermöglichen und die Adressierung, da in RIPv2 Multicast anstatt von Broadcast Paketen verwendet werden. Ein weiterer nicht zu unterschätzender Vorteil von RIPv2 ist, dass dieses Multipathfähig ist und somit zusätzlich zwischen Netzen mit gleicher Metrik unterscheiden kann, um die Netzlast besser zu verteilen.

# Routing Tabelle

Durch die im Netzwerk gesendeten Pakete wurde die Routing-Tabelle entsprechend aktualisiert, dabei wurde mit dem Befehl show ip route die Routing-Tabelle aufgerufen:

R130(config-router)#do show ip route

Codes: L - local, C - connected, S - static, **R - RIP**, M – mobile **…**

**…**

Gateway of last resort is not set 10.0.0.0/16 is subnetted, 1 subnets

S 10.20.0.0 [1/0] via 172.31.0.1

30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks C 30.0.0.0/8 is directly connected, FastEthernet0/0 L 30.0.0.1/32 is directly connected, FastEthernet0/0

31.0.0.0/8 is variably subnetted, 2 subnets, 2 masks C 31.0.0.0/8 is directly connected, Loopback0

L 31.0.0.1/32 is directly connected, Loopback0 35.0.0.0/8 is variably subnetted, 2 subnets, 2 masks

C 35.0.0.0/8 is directly connected, Serial0/0/0 L 35.0.0.1/32 is directly connected, Serial0/0/0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **R** | **40.0.0.0/8** | **[120/1]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **41.0.0.0/8** | **[120/1]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **45.0.0.0/8** | **[120/1]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **50.0.0.0/8** | **[120/2]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **51.0.0.0/8** | **[120/2]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **55.0.0.0/8** | **[120/2]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |
| **R** | **61.0.0.0/8** | **[120/3]** | **via** | **35.0.0.2,** | **00:00:21,** | **Serial0/0/0** |

172.31.0.0/16 is variably subnetted, 2 subnets, 2 masks C 172.31.0.0/16 is directly connected, FastEthernet0/1 L 172.31.4.12/32 is directly connected, FastEthernet0/1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S | 192.168.11.0/24 | [1/0] | via | 172.31.0.1 |
| S | 192.168.16.0/24 | [1/0] | via | 172.31.0.1 |
| S | 192.168.50.0/24 | [1/0] | via | 172.31.0.1 |

**R** steht dabei für RIP-Einträge. Man kann erkennen, dass es sich bei diesen Einträgen um RIPv2 Einträge handelt, da die Subnetzmaske /8 aufgelistet ist. Bei RIPv1 und RIPv2 wird in der Tabelle für jeden RIP-Eintrag eine Netzwerk-IP, die entsprechende Next-Hop IP das entsprechende Interface eingetragen. Zusätzlich dazu wird die Zeit seit der letzten Aktualisierung angegeben, welche in diesem Fall 21 Sekunden beträgt.

# Zusätzliche Fragen

### Zu welcher Kategorie von Routingprotokollen zählt RIP?

RIP Zählt zu den **Distanzvektorprotokollen**: Diese basieren im Gegensatz zu Linkstate basierenden Protokollen darauf, dass ein Router nicht an alle anderen Router seinen Zustand sendet, sondern lediglich die Routing-Tabelle, mit der der jeweiligen Nachbarrouter rekursiv abgeglichen wird. Somit kann auch eine Metrik errechnet werden, die sich einfach inkrementell erhöht. Ergibt sich nun ein kürzerer weg zu einer Ziel-Adresse, so wird dieser entsprechend in die Routing-Tabelle aufgenommen und der Weg mit der schlechteren Metrik wird entfernt.

### Welche Werte bilden bei RIP die Metrik?

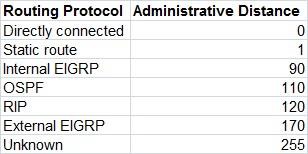
Die Metrik gibt an wie aufwendig ein Weg durchs Netz ist. RIP ermittelt die Metrik nur durch die Hop-Count also wie viele Router die bis zum Ziel durchlaufen werden müssen.

Unglücklicherweise bleibt dadurch die Bandbreite unberücksichtigt.

### Welche „Adminstrative Distanz“ hat RIP per Default auf Cisco Routern?

Falls unterschiedliche Routing-Protokolle für die selbe Ziel-IP unterschiedliche Pfade vorsehen entscheidet die „Adminstrative Distanz“ bei Cisco Routern, welchem Routing- Protokoll tatsächlich Glauben geschenkt wird. Je niedriger die korrespondierende 8-Bit-Zahl, desto glaubwürdiger die Route. Folgende Tabelle gewährt einen kurzen Überblick über

„Adminstrative Distanzen“:



*Abbildung 4 Administrative Distanzen (unvollständig)*

Die Antwort auf die Frage lautet somit: 120.

# Verwendete Geräte, Software, Unterlagen

### Geräte:

* HTL-Ybbs: PC 130-150 und Laborrouter 130-160

### Software:

* PuTTY: Telnet-Zugriff
* Wireshark: Netzwerksniffer

### Unterlagen:

* Buch: HERDT-Verlag: Netzwerke – Protokolle & Dienste

# Weitere Anmerkungen zum Übungsverlauf + Erkenntnisse

Generell verlief die Übung reibungslos, da RIPv1 und RIPv2 direkt nach der Erstellung des Netzplans beim ersten Versuch funktionierten. RIP scheint selbst für kleine Netze eine zwar provisorische aber auch praktische Lösung zu sein.

Nichtsdestotrotz erschien es uns überflüssig die weiterzuleitenden Netze selbst für RIP mithilfe des network <ip> Befehls anzugeben. Es wäre viel sinnvoller Netzwerke, die man kennt, automatisch weiterzuleiten und optional ausschließen zu können als dazu gezwungen zu werden alle weiterzuleitenden Netze einzugeben, was durch mögliches Vertippen eine Fehlerquelle darstellen könnte.

Generell ist es geschickt, ein schnell aufzusetzendes Routing-Protokoll wie RIP in seiner Toolbox zu haben anstatt andauernd statische Routen eingeben zu müssen, welche bei Netzwerkänderungen erneuert werden müssen, was RIP automatisch erledigt.

# Quellen

* Buch: HERDT-Verlag: Netzwerke – Protokolle & Dienste
* <https://de.wikipedia.org/wiki/Routing_Information_Protocol>
* <https://www.computerweekly.com/de/definition/Routing-Information-Protocol-RIP>
* NWTK-Unterricht 3BHIT – IT-HTL Ybbs