

Linux Networking

Markus Gachnang und Martin Sprecher

25. September 2020

Inhaltsverzeichnis

1	Ausgangslage	2
2	Kochbuch	2
2.1	General	2
2.2	IP Address Assignment	2
2.3	BIRD	3
2.4	IP Forwarding	5
2.5	Script	5
3	Verifizierung	9
3.1	Route Failover	9
3.2	Passive Interfaces	9
3.3	Access Website	9
4	Performanz	9
5	Referenzblatt	10
6	Anhänge	10
6.1	Routing	10
6.2	Firewall	11
6.3	Quellennachweis	11
7	Nachwort	11

1 Ausgangslage

2 Kochbuch

It is expected of you to hand in a step-by-step cookbook for the whole final setup. Explain important commands and reason your decisions. We should be able to fully retrace what you did to be able to assess your work. One cookbook is expected per group.

2.1 General

Change the password ... Also, change the hostname to the name given in LTB.

Wir verbinden uns auf jeden Container und ändern den Inhalt der Datei `/etc/hostname` auf den Namen des Containers. Dafür benützen wir `sudo nano /etc/hostname`, ändern den Namen und speichern mit Ctrl-O und beenden nano mit Ctrl-X. Zusätzlich rufen wir `sudo hostname <newHostName>` auf.

Wir setzen das Passwort des jeweiligen Containers auf seinen Namen mit `sudo passwd ins`.

2.2 IP Address Assignment

Use Netplan to assign the ip addresses to the interfaces. ...

Name	IP
Client	ENS2: 172.16.0.2
R1	ENS2: 172.16.0.1 ENS3: 10.0.1.1
R2	ENS2: 10.0.1.2 ENS3: 10.0.4.1 ENS4: 10.0.2.1
R3	ENS2: 10.0.2.2 ENS3: 10.0.5.1 ENS4: 10.0.3.1
R4	ENS2: 10.0.4.2 ENS3: 10.0.5.2 ENS4: 10.0.100.1
R5	ENS2: 10.0.3.2 ENS3: 192.168.1.1
Server	?: 192.168.1.100
MITM	ENS2: 10.0.100.2

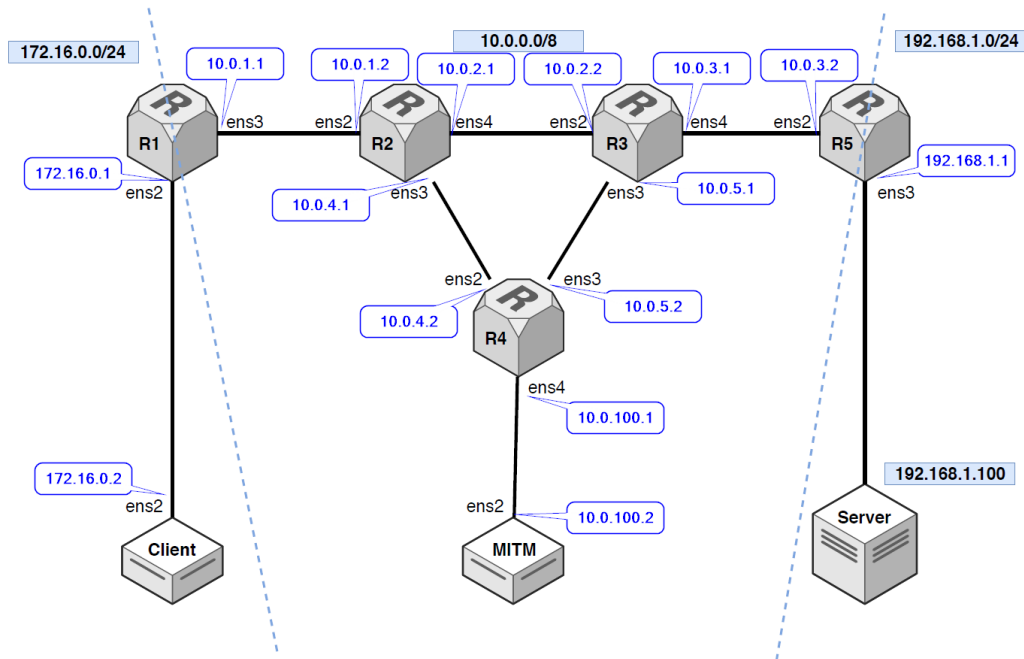


Abbildung 1: Netplan

Die Adressen sind so definiert, dass jede Verbindung ein eigenes Subnet hat. Bird sucht seine Nachbarn anhand des Broadcasts.

Wir ändern den netplan mit `sudo nano /etc/netplan/50-cloud-init.yaml`.

```

1 # This file is generated from information provided by
2 # the datasource. Changes to it will not persist across an instance.
3 # To disable cloud-init's network configuration capabilities, write a file
4 # /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
5 # network: {config: disabled}
6 network:
7   version: 2
8   ethernets:
9     ens2:
10       dhcp4: false
11       addresses: [172.16.0.1/24]
12     ens3:
13       dhcp4: false
14       addresses: [10.0.1.1/24]
```

Listing 1: /etc/netplan/50-cloud-init.yaml on R1

Mit `netplan apply` übernehmen wir die neu definierten IP-Adressen.

2.3 BIRD

Now, your routers must run OSPF. ...

- OSPFv2 must run on the routers.
- Find a way to protect the CPU from too much OSPF processing.
- A restart of BIRD should not result in lost routes.

Wir definieren die Configuration von Bird indem wir `"/etc/bird/bird.conf"` anpassen.

```

1 # This file is generated from information provided by $0
2 # Please refer to the documentation in the bird-doc package or BIRD User's
```

```

3 # Guide on http://bird.network.cz/ for more information on configuring BIRD
  and
4 # adding routing protocols.
5
6 # log "/var/log/bird.log" all;
7 log syslog { info, remote, warning, error, auth, fatal, bug };
8
9 # Change this into your BIRD router ID.
10 router id 1.1.1.1;
11
12 # The Device protocol is not a real routing protocol. It doesn't generate
  any
13 # routes and it only serves as a module for getting information about
  network
14 # interfaces from the kernel.
15 protocol device {
16     scan time 10; # Scan interfaces every 10 seconds
17 }
18
19 # The Kernel protocol is not a real routing protocol. Instead of
  communicating
20 # with other routers in the network, it performs synchronization of BIRD's
21 # routing tables with the OS kernel.
22 protocol kernel {
23     metric 64;          # Use explicit kernel route metric to avoid collisions
24                        # with non-BIRD routes in the kernel routing table
25     persist;           # Don't remove routes on BIRD shutdown
26     scan time 20;      # Scan kernel routing table every 20 seconds
27     import none;
28     export all;        # Actually insert routes into the kernel routing table
29 }
30
31
32 protocol rip {
33     export all;
34     import all;
35     interface "*";
36 }
37
38 protocol static {
39     import all;
40
41     172.16.0.0/24 via "ens2"
42 }
43 protocol ospf {
44     tick 10;           # The routing table calculation and clean-up of areas'
                        # databases is not performed when a single link
45                        # state change arrives. To lower the CPU utilization,
                        # it's processed later at periodical intervals of num
46                        # seconds. The default value is 1.
47     import all;
48     #export filter {
49     #     ospf_metric1 = 1000;
50     #     if source = RTS_STATIC then accept; else reject;
51     #};
52
53     area 0 {
54         networks {
55             10.0.0.0/8;
56             172.16.0.0/24;
57             192.168.1.0/24;
58         };
59
60         interface "ens*" {
61             cost 5;
62             type broadcast;

```

```

63         hello 5;
64         retransmit 2;
65         wait 10;
66         dead 20;
67     };
68
69     interface "*" {
70         cost 1000;
71         stub;
72     };
73 };
74 }

```

Listing 2: /etc/bird/bird.conf on R1

Zuerst definieren wir die "router id" nach Vorgabe. Unter "protocol kernen" definieren wir, dass alle routes in die kernel routing table exportiert werden und dort persistent sind, dadurch funktioniert das Routing weiterhin wenn Bird beendet wird.

Unter "protocol static" werden statische routes eingetragen. Nur R1 und R5 haben dort Einträge um ein routing nach Aussen / Innen des Routing-Netzwerkes zu ermöglichen.

Unter "protocol ospf" definieren wir das Verhalten des OSPF. Wir setzen "tick" auf 10, dadurch werden Änderungen innerhalb von 10 Sekunden gesammelt, bevor diese ausgeführt werden. Dadurch wird das Netzwerk weniger geflutet und die CPU ausgelastet.

Wir definieren, dass die "area 0" aus den Netzwerken "10.0.0.0/8", "172.16.0/24" und "192.168.1.0/24" besteht. Dadurch weiss OSPF, welche Adressen geroutet werden sollen. Zusätzlich definieren wir, welche Interface von OSPF verwendet werden sollen. Dort definieren wir auch, in welchem Intervall zum Beispiel ein "hello" geschickt wird. Um ebenfalls das fluten des Netzwerkes zu minimieren, werden gewisse Werte höher gesetzt als der Standardwert.

Zum Schluss definieren wir, dass alle restlichen Interfaces "stub" sind und daher zu ignorieren sind.

Wir haben eine Vorlage aus dem Internet übernommen [1] und die einzelnen Einstellungen anhand von [2] überprüft und angepasst.

2.4 IP Forwarding

"IP-Forwarding" ist standardgemäss ausgeschaltet. Mit `sysctl net.ipv4.ip_forward=1` aktivieren wird das Weiterleiten der Pakete erlaubt.

2.5 Script

Da wir keine Ahnung von "Bird" hatten, wurde das Anpassen und Testen der Configuration aufwendig, weshalb wir kurzer Hand ein Script erzeugten, welche die zuvor definierten Einstellungen anwendet:

```

1  #!/bin/bash
2
3  setup_hostname()
4  {
5      echo "Set hostname to $1"
6      echo "$1" > '/etc/hostname'
7      hostname $1
8  }
9
10 setup_ip()
11 {
12     echo "Setup netplan"
13     cat <<EOF > '/etc/netplan/50-cloud-init.yaml'
14 # This file is generated from information provided by $0

```

```

15 # Changes to it will not persist across an instance.
16 # To disable cloud-init's network configuration capabilities, write a file
17 # /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
18 # network: {config: disabled}
19 network:
20     version: 2
21     ethernets:
22 EOF
23     if [ $# -ge "1" ]; then
24         echo " ens2 = $1"
25         cat <<EOF >> '/etc/netplan/50-cloud-init.yaml'
26         ens2:
27             dhcp4: false
28             addresses: [$1]
29 EOF
30     fi
31     if [ $# -ge "2" ]; then
32         echo " ens3 = $2"
33         cat <<EOF >> '/etc/netplan/50-cloud-init.yaml'
34         ens3:
35             dhcp4: false
36             addresses: [$2]
37 EOF
38     fi
39     if [ $# -ge "3" ]; then
40         echo " ens4 = $3"
41         cat <<EOF >> '/etc/netplan/50-cloud-init.yaml'
42         ens4:
43             dhcp4: false
44             addresses: [$3]
45 EOF
46     fi
47 }
48
49 setup_bird()
50 {
51     echo "Setup bird"
52     cat <<EOF >> '/etc/bird/bird.conf'
53     # This file is generated from information provided by $0
54     # Please refer to the documentation in the bird-doc package or BIRD User's
55     # Guide on http://bird.network.cz/ for more information on configuring BIRD
56     and
57     # adding routing protocols.
58     # log "/var/log/bird.log" all;
59     log syslog { info, remote, warning, error, auth, fatal, bug };
60
61     # Change this into your BIRD router ID.
62     router id $1;
63
64     # The Device protocol is not a real routing protocol. It doesn't generate
65     any
66     # routes and it only serves as a module for getting information about
67     network
68     # interfaces from the kernel.
69     protocol device {
70         scan time 10; # Scan interfaces every 10 seconds
71     }
72
73     # The Kernel protocol is not a real routing protocol. Instead of
74     communicating
75     # with other routers in the network, it performs synchronization of BIRD's
76     # routing tables with the OS kernel.
77     protocol kernel {
78         metric 64; # Use explicit kernel route metric to avoid collisions
79                 # with non-BIRD routes in the kernel routing table

```

```

77     persist;           # Don't remove routes on BIRD shutdown
78     scan time 20;      # Scan kernel routing table every 20 seconds
79     import none;
80     export all;        # Actually insert routes into the kernel routing table
81 }
82
83
84 protocol rip {
85     export all;
86     import all;
87     interface "*";
88 }
89
90 protocol static {
91     import all;
92
93 EOF
94
95     for var in "$@"
96     do
97         echo " $var"
98         if [ "$var" != "$1" ]; then
99             echo "         route $var;" >> '/etc/bird/bird.conf'
100         fi
101     done
102
103     cat <<EOF >> '/etc/bird/bird.conf'
104 }
105
106 protocol ospf {
107     tick 10;           # The routing table calculation and clean-up of areas'
                        # databases is not performed when a single link
108                        # state change arrives. To lower the CPU utilization,
                        # it's processed later at periodical intervals of num
109                        # seconds. The default value is 1.
110
111     import all;
112     #export filter {
113     #    ospf_metric1 = 1000;
114     #    if source = RTS_STATIC then accept; else reject;
115     #};
116
117     area 0 {
118         networks {
119             10.0.0.0/8;
120             172.16.0.0/24;
121             192.168.1.0/24;
122         };
123
124         interface "ens*" {
125             cost 5;
126             type broadcast;
127             hello 5;
128             retransmit 2;
129             wait 10;
130             dead 20;
131         };
132
133         interface "*" {
134             cost 1000;
135             stub;
136         };
137     }
138
139 EOF
140     sysctl net.ipv4.ip_forward=1

```



```

141     ip route flush table main
142     bird -p
143     birdc down
144     # bird -R
145     systemctl start bird
146     birdc show status
147     systemctl status bird
148 }
149
150 setup()
151 {
152     echo "Start setup for '$1'"
153     case $1 in
154         Client)
155             setup_hostname "Client"
156             setup_ip "172.16.0.2/24"
157             echo "                gateway4: 172.16.0.1" >> '/etc/netplan/50-
                cloud-init.yaml'
158             ;;
159         MITM)
160             setup_hostname "MITM"
161             setup_ip "10.0.100.2"
162             echo "                gateway4: 10.0.100.1" >> '/etc/netplan/50-
                cloud-init.yaml'
163             ;;
164         R1)
165             setup_hostname "R1"
166             setup_ip "172.16.0.1/24" "10.0.1.1/24"
167             setup_bird "1.1.1.1" "172.16.0.0/24 via \"ens2\""
168             # "10.0.0.0/8 via \"ens3\""
169             ;;
170         R2)
171             setup_hostname "R2"
172             setup_ip "10.0.1.2/24" "10.0.4.1/24" "10.0.2.1/24"
173             setup_bird "2.2.2.2"
174             # "10.0.1.0/24 via \"ens2\"" "10.0.4.0/24 via \"ens3\""
                "10.0.3.0/24 via \"ens4\""
175             ;;
176         R3)
177             setup_hostname "R3"
178             setup_ip "10.0.2.2/24" "10.0.5.1/24" "10.0.3.1/24"
179             setup_bird "3.3.3.3"
180             # "10.0.2.0/24 via \"ens2\"" "10.0.4.0/24 via \"ens3\""
                "10.0.5.0/24 via \"ens4\""
181             ;;
182         R4)
183             setup_hostname "R4"
184             setup_ip "10.0.4.2/24" "10.0.5.2/24" "10.0.100.1/24"
185             setup_bird "4.4.4.4"
186             ;;
187         R5)
188             setup_hostname "R5"
189             setup_ip "10.0.3.2/24" "192.168.1.1/24"
190             setup_bird "5.5.5.5" "192.168.1.0/24 via \"ens3\""
191             # "10.0.0.0/8 via \"ens2\""
192             ;;
193         *)
194             echo "name is unknown..."
195             exit 1
196             ;;
197     esac
198     netplan apply
199 }
200
201 echo "CldInf Networker"
202 if [ "$EUID" -ne 0 ]; then

```

```

203     echo "Please run as root"
204     exit 1
205 elif [ $# -lt "1" ]; then
206     hostname=$(hostname)
207     case $hostname in
208         Client|MITM|R1|R2|R3|R4|R5)
209         setup $hostname
210         ;;
211         *)
212         echo "Usage: $0 <name>"
213         echo " name = Client, MITM, [R1 .. R5]"
214         exit 1
215         ;;
216     esac
217 else
218     setup $1
219 fi

```

Listing 3: /doConfig.sh

3 Verifizierung

Verify your routing implementation. Explain which exact commands you used for each verification step and show how they provide prove that your setup works.

3.1 Route Failover

Verify that your setup works. Prove that a route failover takes place in case of a route outage. To do that, a well known tool can be used.

3.2 Passive Interfaces

Show that no OSPF packets are sent into the client and server networks, too. `tcpdump` and `tshark` are good tools for that, sniff on the suspicious interfaces and filter for OSPFv2 packets. To be sure that your filter works, sniff on an interface where you expect OSPF-messages, too.

Wir haben versucht, anhand von Wireshark mit "Sshdump" und "Ciscodump" auf den Interfaces zu lauschen, erhielten aber keine Pakete. Wir versuchten ebenfalls mit `tcpdump` die Pakete aufzufangen. Dies klappte zwar, aber das pcap-File wollte sich einfach nicht übertragen lassen. Wir sind anhand von [3] vorgegangen.

3.3 Access Website

Finally, you must be able to access the webpage from the webserver. You can use `curl` or `wget` for that. The webserver listens on port 8080.

4 Performanz

Provide the measurement before and after the appliance of the `tc` commands.
Provide the exact used `tc` commands.

5 Referenzblatt

We also expect you to hand in a reference sheet for all the net-work commands used in this lab. Just list every command and its function. This reference must not be longer than one page.

Befehl	Funktion
<code>sudo nano <Pfad></code>	Öffnen eines Files im Texteditor
<code>sudo hostname <newHostName></code>	Zum ändern des Hostnames
<code>sudo passwd</code>	Ändert das Passwort des aktuellen Benutzers.
<code>sudo birdc</code>	Zum kommunizieren mit einem laufenden BIRD. Einzelne Befehle, die hier nicht aufgeführt sind, sind auf [4] zu finden.
<code>sudo birdc show status</code>	Anzeigen vom router status, BIRD version, Laufzeit und Zeitpunkt von der letzten Rekonfiguration.
<code>sudo birdc show interfaces</code>	Anzeigen der Liste von allen Interfaces. Zeigt für jedes Interface, Typ, Status, MTU und die zugewiesene Adresse an.
<code>sudo birdc show ospf interface</code>	Anzeigen von detaillierten Informationen über die OSPF Interfaces.
<code>birdc show ospf neighbors</code>	Anzeigen der Liste mit allen OSPF Nachbarn deren Zustand.
<code>birdc show ospf state</code>	Anzeigen von detaillierten Informationen über OSPF Bereiche basierend auf der link-state database. Es zeigt die Netzwerk Topologie, stub Netzwerke, zusammengeführte Netzwerke und Router von anderen Areas und externen Routen. Ausserdem zeigt es erreichbare Netzwerk Knoten an.
<code>ping -c <num> <ip></code>	sendet num mal einen Ping an die ip
<code>tcptraceroute <ip></code>	Zum Anzeigen vom Pfad im Netzwerk vom Host, auf dem die Traceroute ausgeführt wird, und der angegebenen IP, sowie dem Ort, an dem die Route, falls vorhanden, nicht abgeschlossen werden kann und allen Hops bis zum Ziel.
<code>blub</code>	blob
<code>blub</code>	blob
<code>blub</code>	blob

6 Anhänge

6.1 Routing

- Your delivered report must include the new usernames and pass-words of the hosts.
- Your delivery must contain the created netplan files and an ad-dress plan.
- Attach the BIRD config files to your cookbook and explain how you achieve the minimal requirements.
- Verify your routing implementation. Explain which exact commands you used for each verification step and show how they provide prove that your setup works.

Erarbeitung der BIRD-config ist im 2.3 erläutert. Verifikation der Routing unter 3.

6.2 Firewall

The next step is to implement a firewall in your network. On Linux, netfilter was the kernel firewall for a long time. To configure netfilter, tools like iptables, ip6tables, arptables and so on were used. But for this task, you'll have to use the more modern nftables and its nft utility to configure it. First, check all open ports of the webserver from your client with nmap. Your firewall solution must fulfill the following requirements:

- You must use nftables and its utility nft. Do not use iptables.
- The firewall must run on one of the routers.
- The website is only accessible from the clients network (172.16.0.0/24).
- Pings must not be answered.
- nmap must not show any open ports. Check if your firewall works by executing nmap again.

We're interested in the used `nmap` command, where the firewall runs and why you've chosen that location. Print the ruleset of the firewall and attach it to your report.

6.3 Quellennachweis

- [1] Ondrej Sury, "Ospf_example · wiki · labs / bird internet routing daemon · gitlab," 25.09.2020. [Online]. Available: https://gitlab.nic.cz/labs/bird/-/wikis/OSPF_example#:~:text=OSPF%20example,ptp%20links%20to%20other%20routers.
- [2] Tomas Filip - HIPPO © 2004 and C. F. ALL: WEBarium, "The bird internet routing daemon project," 25.09.2020. [Online]. Available: https://bird.network.cz/?get_doc&v=20&f=bird-6.html
- [3] A. Phillips, "How to run a remote packet capture with wireshark and tcpdump," *Comparitech*, 27.12.2018. [Online]. Available: <https://www.comparitech.com/net-admin/tcpdump-capture-wireshark/>
- [4] Tomas Filip - HIPPO © 2004 and C. F. ALL: WEBarium, "The bird internet routing daemon project," 25.09.2020. [Online]. Available: https://bird.network.cz/?get_doc&v=20&f=bird-4.html

7 Nachwort

Kein Mitglied unserer Gruppe hat CN2 besucht. Alleine bis wir 2.3 zum laufen kriegten haben wir gemeinsam mehr als 30 Stunden aufwenden müssen.