



## ***OPEN SHORTEST PATH FIRST***

**Bearbeitet von**

**Suat Algin,  
201210912,  
[salgin@uni-koblenz.de](mailto:salgin@uni-koblenz.de)  
Teil 1-3**

**Lars Blumenstengel,  
201210696,  
[imulb@uni-koblenz.de](mailto:imulb@uni-koblenz.de)  
Teil 4-6**

Praktikum Routingsimulation  
Universität Koblenz-Landau  
Universitätsstrasse 1  
56070 Koblenz

# Inhaltsverzeichnis

<b>1. Einleitung</b>	
1.1 Vorwort.....	2
1.2 Historie.....	2
1.3 OSPF Funktionsweise.....	2
<b>2. Zebra in Vnuml einbinden.....</b>	<b>4</b>
<b>3. EasyOspf Netz.....</b>	<b>5</b>
3.1 Die XML-Datei.....	6
3.2 Zebra: Konfigurationsdateien, Befehle und Bedeutungen.....	8
3.3 Routintabellen, Ping und Tracroute.....	11
<b>4. VortragsNetz.....</b>	<b>15</b>
4.1 Die XML-Datei.....	15
4.2 Die Config-Dateien.....	20
4.3 Kosten und deren Einfluss.....	31
4.4 Die debug-Ausgabe.....	32
<b>5. Fazit.....</b>	<b>34</b>
<b>6. Quellen und Verweise.....</b>	<b>34</b>

# 1. Einleitung

## 1.1 Vorwort

In dieser Ausarbeitung werden wir versuchen die Vorteile des OSPF Protokolls zu verdeutlichen und das Protokoll anhand von Netzwerk Beispielen zu erläutern. Die Beispiele in dieser Ausarbeitung sind alle im Praktikum „Routing Simulation“ entstanden und mit dem VNUML( Virtual Network User Mode Linux)-Tool entwickelt worden.

## 1.2 Historie

OSPF (Open Shortest Path First) wurde 1991 von der “Internet Engineering Task Force” (IETF) entwickelt und löste nach ihrer Spezifizierung 1998 das RIPv1 als Interior Gateway Protocol ab. Das Open in Ospf steht für die Tatsache, dass es keinem Unternehmen gehört, sondern von jeder Organisation verändert oder spezifiziert werden darf. So entstanden sehr viele Variationen von Ospf Software, wie z.B. Ciscos IOS 12.3, Data Connections OSPF, usw.

## 1.3 OSPF Funktionsweise

Ospf ist ein Link State Verfahren und basiert auf dem Shortest Path First Algorithmus von Edsger Dijkstra. Somit berechnet dieser Algorithmus immer den kürzesten Weg zwischen zwei zusammenhängenden Knoten A und B in einem Kanten gewichteten Graphen. Ospf ermöglicht durch aufteilen des Netzwerks in verschiedene Areas eine hierarchische Netzwerk Topologie. Eine Area besteht aus einer Menge von zusammenhängenden Routern, die auch nur eine Link State Database über ihre eigene Area führen. Dadurch werden Informationen eingespart und die einzelnen Routingtabellen (siehe unten) in Grenzen gehalten.

Die Grundlage für die Arbeitsweise des Ospf-Routingprotokolls bildet, wie schon erwähnt, die Topologie-Datenbank, die gemeinsam von den einzelnen Routern aufgebaut wird. Dazu überträgt (flutet) jeder Knoten die Information seiner lokalen Sicht (seine Interfaces und die erreichbaren Nachbarn) zu allen anderen Knoten in seiner Area. Mit dieser Information kann jedes einzelne Gateway einen Baum aufbauen, der die kürzesten Wege zu allen erreichbaren Knoten enthält. Die Wurzel des Baumes ist der Router selbst, der für den Aufbau des Baums den bereits in den vorhergehenden Abschnitten beschriebenen Shortest Path Algorithmus verwendet. Mit Hilfe des Baums, der die kürzesten Wege zu jedem Netzwerk bzw. jedem Endsystem beschreibt, wird dann eine Tabelle erstellt, in der die Routinginformation zusammengefasst werden. Für die Weiterleitung von Datagrammen werden dabei nur das Zielnetz, der nächste Knotenrechner und die Distanz benötigt, die Beschreibung des kompletten Weges ist nicht erforderlich. Um ein Netzwerk über mehrere Areas zu realisieren besitzt Ospf die Möglichkeit spezielle Router zu setzen, die auch Informationen über ihre eigene Area heraus führen. Diese Router werden Area Border Router genannt. Ein großer Vorteil des Ospf Protokolls im Gegensatz zu anderen Protokollen liegt in seiner

Stabilität. Ospf unterstützt gleichzeitig mehrere Verbindungswege gleicher Kosten zu einem Zielnetz und ist in der Lage, den auftretenden Datenverkehr über verschiedene Verbindungswege zu übertragen. Besondere Vorzüge liegen aber auch in der schnellen Konvergenz und wie schon erwähnt, in der sparsamen Verwendung von Bandbreite bei der Erstellung neuer Routingtabellen. Auch die Tatsache das Ospf in der Lage ist, bei Ausfall eines Netzes innerhalb einer Area selbstständig zu reagieren und sich einen neuen Weg zu suchen, um das defekte Netz zu umgehen, war ein wichtiger Vorteil bei der Etablierung dieses Protokolls.

Eine andere Besonderheit des Ospf-Protokolls ist die Unterstützung von IP-Netzen mit untergliederten Subnetzen. Zu diesem Zwecke versieht das Protokoll jede Route mit einer Subnetz-Maske, die einen bestimmten Bereich von Adressen einer bestimmten Route zuordnet. So können durch das Adress-Masken-Paar [0xffff0000, 134.106.0.0] mit einem Eintrag in der Routingtabelle die Wege zu den Netzen im Bereich von 134.106.0.0 bis 134.106.255.255 propagiert werden.

Zusammenfassend kann gesagt werden, dass es sich beim Ospf-Protokoll um ein sehr komplexes und modernes Routing-Protokoll handelt, dessen Merkmale in diesem Rahmen nicht vollständig behandelt werden können. Ospf eignet sich insbesondere für den Einsatz in großen bis sehr großen Netzwerken. In kleineren Netzen sollte es aufgrund seiner Komplexität nicht angewendet werden.

## 2. Zebra in Vnuml einbinden

Wenn die Beispiele von der, in diesem Praktikum entstandenen, Live DVD benutzt werden, ist dieser Abschnitt nicht von Bedeutung

Eine Einführung in Vnuml mit den entsprechenden Installationsanweisungen erhaltet ihr in dem Vortrag „Vnuml“, welches auch in dem Praktikum „Routing Simulation“ entstanden ist. Wenn ihr Vnuml ordnungsgemäß installiert habt, braucht ihr nur noch den Zebra Daemon. Das Zebra Tool kann man unter folgendem Link herunterladen:

<ftp://ftp.zebra.org/pub/zebra>

Nach dem Download muss man sich als root in der Linux-Konsole anmelden und den Pfad, in dem die Zebra-Dateien stehen, öffnen.

1)

Als nächstes konfiguriert man die Installationsdatei in dem man in der Konsole den Befehl **configure** eingibt.

Wenn dieser Prozess ohne Fehlermeldungen beendet wird, kann man zum nächsten Schritt übergehen. Ansonsten werden fehlende Module und Pakete festgestellt und müssen von der Linux DVD nachträglich installiert werden.

Nach dem installieren der Pakete wird der erste Schritt nochmals wiederholt.

2)

Um die entsprechenden Dateien für die Installation vorzubereiten, muss man den Befehl **make** in der Konsole eingeben und auch die Ausgaben sorgfältig beobachten.

3)

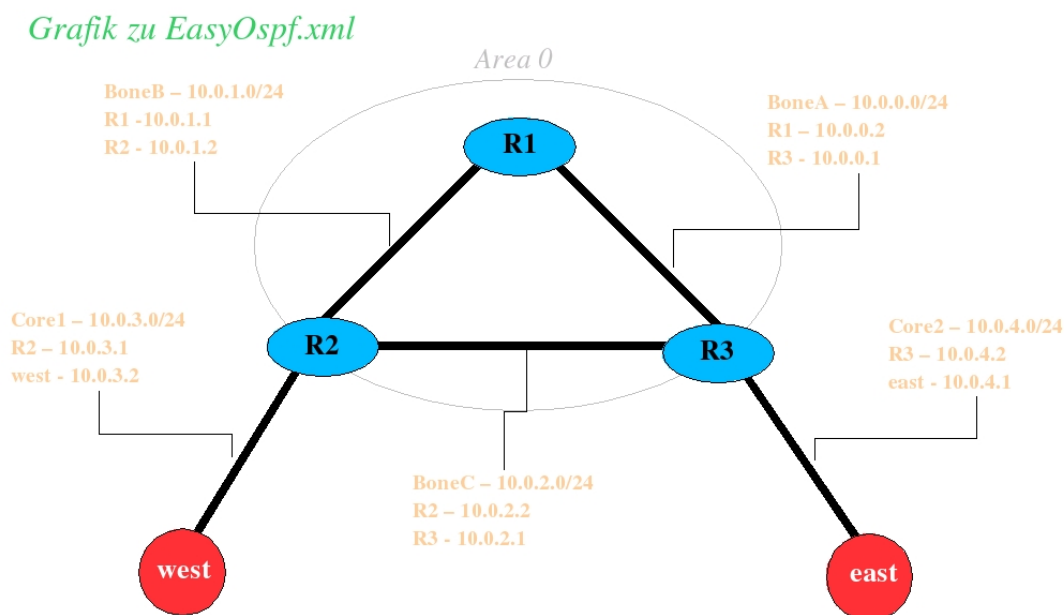
Zur abschließenden Installation fehlt nur noch ein letzter Schritt. Mit dem Befehl **make install** werden die nötigen Dateien in die entsprechenden Verzeichnisse kopiert und der Zebra Deamon ist vollständig installiert.

### 3. Easy OSPF Netz

Im ersten Abschnitt gehen wir zuerst einmal auf die XML- Datei (Abbildung 3.2) ein. Natürlich werden nicht alle XML- Befehle hier erklärt. Die grundlegenden Befehle werden in der Ausarbeitung „VNUML“ ausgiebig behandelt. Wir werden nur auf die, für den Ospf-Daemon, wichtigen Bereiche eingehen und erläutern. Die ausgewählten Bereiche werden entsprechend kenntlich gemacht.

Anhand dieses Beispiels werden wir eine kurze Einleitung zu den Konfigurationsdateien vornehmen und die Routingtabellen erläutern.

Abbildung 3.1



In der Abbildung 3.1 wird die Netzwerktopologie zu Easy Ospf vorgestellt.

Wie man in der Grafik sehen kann besteht dieses Beispiel nur aus einer einzigen Area.

Auf den drei Routern R1, R2, R3 läuft der Ospf –Daemon.

Die beiden Hosts „west“ und „east“ beinhalten als Ausgang nur einen statischen Eintrag und wissen sonst nichts über das Netzwerk.

Erläuterungen zur Grafik:

Die Netze zwischen den Routern sind mit BoneA, BoneB oder BoneC gekennzeichnet.

Um zu verdeutlichen das die Netze, welche zu den Hosts führen, nicht zur Area0 gehören, wurden diese mit Core1 oder Core2 gekennzeichnet.

Wir haben in diesem Beispiel nur Netze mit der Subnetzmaske 255.255.0.0 benutzt, da dieser Adressbereich für unsere Zwecke vollkommen ausreicht.

Die Netze in der Grafik sind entsprechend gekennzeichnet:

z.B.

BoneA – 10.0.0.0/24

R1 – 10.0.0.2

R3 – 10.0.0.1

Die letzten beiden Ausdrücke zeigen das der Router R1 an der Schnittstelle zu R3 die Adresse 10.0.0.2 hat. Dementsprechend hat der Router R3 an der Schnittstelle zu R1 die Adresse 10.0.0.1.

### 3.1 Die XML- Datei

Die Xml-Datei zu EasyOspf sieht wie folgt aus:

```
<vnuml>
  <global>
    <version>1.5</version>
    <simulation_name>EasyOspf</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
  </global>
  <automac>
    <ip_offset>100</ip_offset>
    <host_mapping/>
  </automac>

  <net name="BoneA" mode="uml_switch"/> <!--10.0.0.0/24-->
  <net name="BoneB" mode="uml_switch"/> <!--10.0.1.0/24-->
  <net name="BoneC" mode="uml_switch"/> <!--10.0.2.0/24-->
  <net name="Core1" mode="uml_switch"/> <!--10.0.3.0/24-->
  <net name="Core2" mode="uml_switch"/> <!--10.0.4.0/24-->

  <vm name="R1">
    <filesystem type="cow">/usr/local/share/vnuml/filesystem/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>
    <if id="1" net="BoneA">
      <ipv4 mask="255.255.255.0">10.0.0.2</ipv4>
    </if>
    <if id="2" net="BoneB">
      <ipv4 mask="255.255.255.0">10.0.1.1</ipv4>
    </if>
    <forwarding type="ip"/>
    <filetree root="/etc/zebra" when="start">/usr/local/share/vnuml/examples/EasyOspf/R1</filetree>
    <exec seq="start" type="verbatim">
      for f in /proc/sys/net/ipv4/conf/*/rp filter: do echo 0 > $f; done
    </exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>
  </vm>
  <vm name="R2">
    <filesystem type="cow">/usr/local/share/vnuml/filesystem/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>
    <if id="1" net="BoneA">
      <ipv4 mask="255.255.255.0">10.0.0.1</ipv4>
    </if>
    <if id="2" net="BoneC">
      <ipv4 mask="255.255.255.0">10.0.2.1</ipv4>
    </if>
    <if id="3" net="Core2">
      <ipv4 mask="255.255.255.0">10.0.4.2</ipv4>
    </if>
    <forwarding type="ip"/>
    <filetree root="/etc/zebra" when="start">/usr/local/share/vnuml/examples/EasyOspf/R2
  </filetree>
```

```

<exec seq="start" type="verbatim">
  for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 > $f; done
</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
<exec seq="stop" type="verbatim">killall zebra</exec>
<exec seq="stop" type="verbatim">killall ospfd</exec>
</vm>
<vm name="R3">
  <filesystem type="cow">/usr/local/share/vnuml/filesystem/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>
  <if id="1" net="BoneB">
    <ipv4 mask="255.255.255.0">10.0.1.2</ipv4>
  </if>
  <if id="2" net="BoneC">
    <ipv4 mask="255.255.255.0">10.0.2.2</ipv4>
  </if>
  <if id="3" net="Core1">
    <ipv4 mask="255.255.255.0">10.0.3.1</ipv4>
  </if>
  <forwarding type="ip"/>
  <filetree root="/etc/zebra"
  when="start">/usr/local/share/vnuml/examples/EasyOspf/R3</filetree>
  <exec seq="start" type="verbatim">
    for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 > $f; done
  </exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
  <exec seq="stop" type="verbatim">killall zebra</exec>
  <exec seq="stop" type="verbatim">killall ospfd</exec>
</vm>
<vm name="West">
  <filesystem type="cow">/usr/local/share/vnuml/filesystem/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>
  <if id="1" net="Core1">
    <ipv4 mask="255.255.255.0">10.0.3.2</ipv4>
  </if>
  <route type="inet" gw="10.0.3.1">default</route>
  <forwarding type="ip"/>
</vm>
<vm name="East">
  <filesystem type="cow">/usr/local/share/vnuml/filesystem/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>
  <if id="1" net="Core2">
    <ipv4 mask="255.255.255.0">10.0.3.2</ipv4>
  </if>
  <route type="inet" gw="10.0.4.1">default</route>
  <forwarding type="ip"/>
</vm>
</vnuml>

```

Die fürs Zebra-Daemon wichtigen Teile wurden unterstrichen. Diese Abschnitte sorgen dafür, dass der Zebra-Daemon aufgerufen wird.

```
<filetree root="/etc/zebra" when="start">/usr/local/share/vnuml/examples/EasyOspf/R1</filetree>
```

An dieser Stelle wird das Verzeichnis mit den Konfigurationsdateien (siehe VortragsNetz) bekannt gemacht.

Der Daemon weiß nun, dass er unter diesem Pfad die entsprechenden Dateien vorfindet und greift beim starten (siehe weiter unten) des Protokolls auf die Dateien zu.

Als nächstes wird der virtuellen Maschine die Anweisung gegeben, beim start des Daemons Auf das Verzeichnis mit den „zebra.conf“ und „ospfd.conf“ zuzugreifen und diese unter dem Verzeichnis, /etc/zebra/, zu hinterlegen.

```
<exec seq="start" type="verbatim">
  for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 > $f; done
```

```
</exec>  
<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>  
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>  
<exec seq="stop" type="verbatim">killall zebra</exec>  
<exec seq="stop" type="verbatim">killall ospfd</exec>
```

Die letzten beiden Zeilen stehen für das Herunterfahren des Daemons. Wenn das Szenario gestoppt wird, werden die entsprechenden Dateien (zebra, ospfd) wieder entfernt.

### 3.2 Zebra: Konfigurationsdateien, Befehle und Bedeutungen

Konfigurationsdateien und grundlegende Konfigurationsbefehle:

In einer Konfigurationsdatei werden Debugging-Befehle (siehe VortragsNetz), ein vty-Password, der Ort für die Log-Files festgelegt und die Konfiguration des Routing-Daemons vorgenommen.

Das Standard Verzeichnis für die Konfigurationsdateien lautet:

```
/usr/local/etc/*.conf
```

Die Konfiguration des Daemons wird anhand von drei Dateien vorgenommen. Diese bestehen aus der ospfd.conf, zebra.conf und der vtysh.conf

Beispiel: ospfd.conf von R1

```
! Config by LB+SA  
! Zebra configuration saved from SA  
! 2005/05/30/ 13:50:00  
!  
Hostname R1  
Password xxxx  
!  
interface eth1  
ip ospf cost 20  
!  
interface eth2  
ip ospf cost 10  
!  
debug ospf packet all send  
debug ospf packet all recv  
!  
router ospf  
network 10.0.0.0/30 area 0.0.0.0  
network 10.0.1.0/30 area 0.0.0.0  
log file /tmp/ospfd.log  
!  
line vty  
!
```

Wenn man etwas in der Konfigurationsdatei auskommentieren möchte, macht man dies mit einem Ausrufezeichen. Alle Zeilen mit einem Ausrufezeichen am Anfang haben keine Auswirkung auf die Datei.

Zuerst wird die Datei mit dem Befehl „hostname R1“ dem Router R1 zugewiesen. Um die Datei vor unbefugten Zugriffen zu schützen, kann man ein Passwort vergeben. Die Schnittstellen können anhand der Befehle,

```
interface eth1
ip ospf cost 20
```

mit verschiedenen Kosten belegt werden. Der debug-Befehl speichert die Ospfpakete, welche gesendet oder empfangen wurden, in dem Verzeichnis der unter „log file“ einige Zeilen drunter angegeben wird.

Im nächsten Schritt wird dem Router das Ospfprotokoll zugewiesen und die Netze mit dem der Router verbunden ist aufgeführt. Dabei ist es wichtig, die Subnetzmasken auch so zu wählen, wie man sie in der XML-Datei deklariert hat. Die Area in dem sich das Netz befinden wird am Ende der Zeile festgelegt.

In der „zebra.conf“ und „vtysh.conf“ kann man die Verzeichnisse für die Speicherung der Log-Dateien festlegen.

Nähere Informationen zur Konfigurationsdatei sind im zweiten Beispiel (VortragsNetz) dieser Ausarbeitung zu finden.

Nun können wir das Szenario mal starten. Dazu öffnen wir die Konsole mit root-Rechten und gehen in das Verzeichnis mit der EasyOspf.xml Datei.

Das Szenario wird mit dem vnumlparser-befehl gestartet oder auch wieder herunter gefahren:

```
vnumlparser.pl -t EasyOspf.xml -vB          ← Start
```

```
vnumlparser.pl -d EasyOspf.xml -vB          ← Stopp
```

In vereinzelt Fällen kann es dazukommen, dass das Szenario nicht mehr richtig herunter fährt. In so einem Fall kann man die folgenden Befehle nutzen:

```
vnumlparser.pl -P EasyOspf.xml -v          ← Schnelles beenden
```

```
vnumlparser.pl -f EasyOspf.xml -v          ← Kill
```

Wenn das Szenario ordnungsgemäß geladen wurde, kann man den Zebra-Daemon starten/stoppen:

```
vnumlparser.pl -x start@EasyOspf.xml
```

```
vnumlparser.pl -x stop@EasyOspf.xml
```

Nachdem die XML-Datei und der Daemon gestartet wurden, kann man sich mit dem entsprechenden Befehl in der Konsole, in den Ospf-Daemon, einloggen und die Ospf-Informationen auslesen:

z.B.

```
Telnet R1 2604
```

Mit diesem Befehl greift man auf den Ospf-Daemon des Routers R1 zu.

Nun gibt es verschiedene Befehle um die Ospf Informationen abzurufen.

**show ip ospf:** Dieser Befehl zeigt die Informationen die der Router enthält

Beispiel:

```
R1: show ip ospf
OSPF Routing Process, Router ID: 10.0.0.200
Supports only single TOS (TOS0) routes
This implementation conforms to RFC2328
RFC1583Compatibility flag is disabled
SPF schedule delay 1 secs, Hold time between two SPFs 1 secs
Refresh timer 10 secs
Number of external LSA 0
Number of areas attached to this router: 1

Area ID: 0.0.0.0 (Backbone)
Number of interfaces in this area: Total: 2, Active: 2
Number of fully adjacent neighbors in this area: 2
Area has no authentication
SPF algorithm executed 5 times
Number of LSA 8
```

Die wichtigsten Daten bei diesem Befehl sind zum Beispiel, das der Router die ID 10.0.0.200 hat und konform zum RFC-Standard ist.

Die Verzögerung des „Shortest Path First“ Algorithmus beträgt 1 Sekunde und die Wartezeit zwischen zwei SPFs auch eine Sekunde.

Die Nummer der externen Link State Acknowledgement beträgt null.

Der Router ist auch nur mit einer Area verbunden.

Unter Area ID sieht man, das die Area0 auch die Backbone ist und das der Router zwei aktive Schnittstellen zu seinen 2 direkten Nachbarn besitzt.

Die Authentifikation der Area ist ausgeschaltet und der SPF-Algorithmus wurde schon fünf mal ausgeführt.

**show ip ospf neighbor:** Mit diesem Befehl werden die direkten Nachbarn des Routers aufgelistet

Beispiel:

```
R1:show ip ospf neighbor
Neighbor ID  Pri  State           Dead Time  Address      Interface
10.0.2.200   1  Full/DR        00:00:33  10.0.0.1    eth1:10.0.0.2
10.0.1.200   1  Full/DR        00:00:33  10.0.1.2    eth2:10.0.1.1
```

Hier sieht man die Router IDs, der mit dem Router R1 verbundenen Router.

Da beide Router die Priorität 1 haben, sind haben sie den Status Full/DR.

Die Dead Time läuft immer von 40 Sekunden runter und befindet sich im Moment bei 33 Sekunden. Wenn diese Zeit abläuft wird die Verbindung als ausgefallen betrachtet.

Der Router ist mit den Schnittstellen eth1 und eth2 an die zwei Router angeschlossen, wobei die angeschlossenen Router die Adressen 10.0.0.1 und 10.0.1.2 haben.

**show ip ospf database:** Der Befehl bewirkt das die Ospf Database aufgelistet wird

(siehe im Zusammenhang VortragsNetz)

**show ip ospf route:** Damit kann man sich die kompletten Informationen des gesamten Netzes anzeigen lassen

Beispiele:

R1: show ip ospf route

```
===== OSPF network routing table =====
N 10.0.0.0/24 [10] area: 0.0.0.0
    directly attached to eth1
N 10.0.1.0/24 [10] area: 0.0.0.0
    directly attached to eth2
N 10.0.2.0/24 [20] area: 0.0.0.0
    via 10.0.0.1, eth1
    via 10.0.1.2, eth2
N IA 10.0.3.0/24 [20] area: 0.0.0.0
    via 10.0.1.2, eth2
N IA 10.0.4.0/24 [20] area: 0.0.0.0
    via 10.0.0.1, eth1

===== OSPF router routing table =====
R 10.0.1.200 [10] area: 0.0.0.0, ABR
    via 10.0.1.2, eth2
R 10.0.2.200 [10] area: 0.0.0.0, ABR
    via 10.0.0.1, eth1

===== OSPF external routing table =====
```

In der „OSPF network routing table“ stehen alle Netzwerke, die sich innerhalb der Area des Routers R1 befinden. Dabei sind die Kosten innerhalb der Klammern (z.B. [10]) angegeben. Auch die Schnittstellen oder Gateways können direkt abgelesen werden.

Unter „OSPF router routing table“ sind die zwei Router angegeben, die direkt mit dem Router R1 verbunden sind. Ein wichtiges Merkmal ist auch das die zwei Router Area Border Router sind.

### 3.3 Routingtabellen, Ping, Traceroute

Nachdem wir das Szenario hochgefahren haben, können wir auf die einzelnen Router und Hosts zugreifen.

Der Befehl um auf eine virtuelle Maschine zuzugreifen lautet:

z.B.

```
ssh R1
```

Mit diesem Befehl greifen wir mit dem vorher erzeugten ssh-Schlüssel auf Router R1 zu.

```
ssh west
```

Mit diesem Befehl greifen wir auf den Host west zu.

Nun können wir uns mit dem Befehl „route“ die einzelnen Routingtabellen der Router ansehen.

z.B.

R1: route

Destination	Gateway	Metric	Interface
192.168.1.144	*	0	eth0
10.0.0.0	*	0	eth1
10.0.1.0	*	0	eth2

R2: route

192.168.1.148	*	0	eth0
10.0.4.0	*	0	eth3
10.0.0.0	*	0	eth1
10.0.2.0	*	0	eth2

West: route

192.168.1.156	*	0	eth0
10.0.3.0	*	0	eth1
default	10.0.3.1	0	eth1

Wie man sieht wissen die Router nicht besonders viel. Ihre Routingtabellen beinhalten nur die direkt verbundenen Netze und nicht mehr. Das hängt damit zusammen, dass wir das Szenario gestartet haben, aber der Ospf-Daemon noch nicht läuft.

Der Stern hinter dem jeweiligen Netz bedeutet, dass der Router direkt mit dem Netz verbunden ist.

Die Kosten zu den jeweiligen Routern haben den Wert 0, da man die Werte, ohne Informationsaustausch nicht berechnen kann. Die Schnittstellen des Routers mit den verbundenen Netzen werden unter Interfaces aufgelistet.

Die 192.168.1er Netze sind die Verbindungen zum virtuellen Host. Dieser virtuelle Host hat eine Verbindung zu allen Routern und Hosts. Er ist sozusagen die globale Einheit, indem alles zusammen läuft.

Die Default-Route beim Host West, wurde statisch gesetzt und befindet deswegen in der Routingtabelle.

Um nochmals zu verdeutlichen das sich die Router untereinander nicht verständigen können, senden wir einen Ping oder einen Traceroute von einem beliebigen Router aus.

In unserem Beispiel vom Router R1 zum Router R2:

R1: ping 10.0.3.2

```
connect: Network is unreachable
```

Oder ein Traceroute:

Dieser Befehl ist ein nützliches Werkzeug um die Konfiguration des Netzes zu überprüfen und die Wege der Pakete zu kontrollieren.

R1: traceroute 10.0.3.2

traceroute: Warning findsaddr:netlink error: Network is unreachable

Wir können jetzt den Ospf-Daemon starten und uns die Routingtabellen noch mal anschauen:

```
vnumlparser.pl -x start@EasyOspf.xml
```

```
ssh R1
```

```
R1: route
```

Destination		Metric	Interface
192.168.1.144	*		eth0
10.0.4.0	10.0.0.1		eth1
10.0.0.0	*		eth1
10.0.1.0	*		eth2
10.0.2.0	10.0.0.1		eth1
10.0.3.0	10.0.1.2		eth2

```
ssh R2
```

```
R2: route
```

Destination	Gateway	Genmask	Flag	Metric	Interface
192.168.1.148	*	255.255.255.252	U	0	eth0
10.0.4.0	*	255.255.255.255	U	0	eth3
10.0.0.0	*	255.255.255.255	U	0	eth1
10.0.1.0	10.0.0.2	255.255.255.255	UG	20	eth1
10.0.2.0	*	255.255.255.255	U	0	eth2
10.0.3.0	10.0.2.2	255.255.255.255	UG	20	eth2

```
ssh west
```

```
west: route
```

Destination		Metric	Interface
192.168.1.156	*		eth0
10.0.3.0	*		eth1
default	10.0.3.1		eth1

Wie man sieht, wissen die Router jetzt deutlich mehr. Sie wissen nun über alle Netze, die das Szenario beinhaltet, bescheid.

Unter Gateway sind die Schnittstellen eingetragen über die, die Pakete gesendet werden. Die Subnetzmasken befinden sich unter Genmask. Das Flag zeigt an, ob das Netz direkt mit dem Router verbunden ist oder über ein direkt verbundenes Gateway gesendet werden muss.

Nun können wir nochmals einen Ping und einen Traceroute senden.

```
R1: ping 10.0.3.2
```

```
64 Bytes from 10.0.3.2 icmp_seq=1 ttl=63 time=27,3ms
```

Diese Ausgabe bestätigt, dass der Ping beim Empfänger angekommen ist.

R1: traceroute 10.0.3.2

1	10.0.1.2 (10.0.1.2)	ms	ms	ms
2	10.0.3.2 (10.0.3.2)	ms	ms	ms

Dieser Traceroute zeigt uns den genauen Weg, der durchlaufen wird, wenn wir ein Packet zur Adresse 10.0.3.2 senden.

Er besagt das das Paket zuerst an die Schnittstelle 10.0.1.2 gesendet wird und danach an der Zieladresse 10.0.3.2 ankommt.

West: traceroute 10.0.4.1

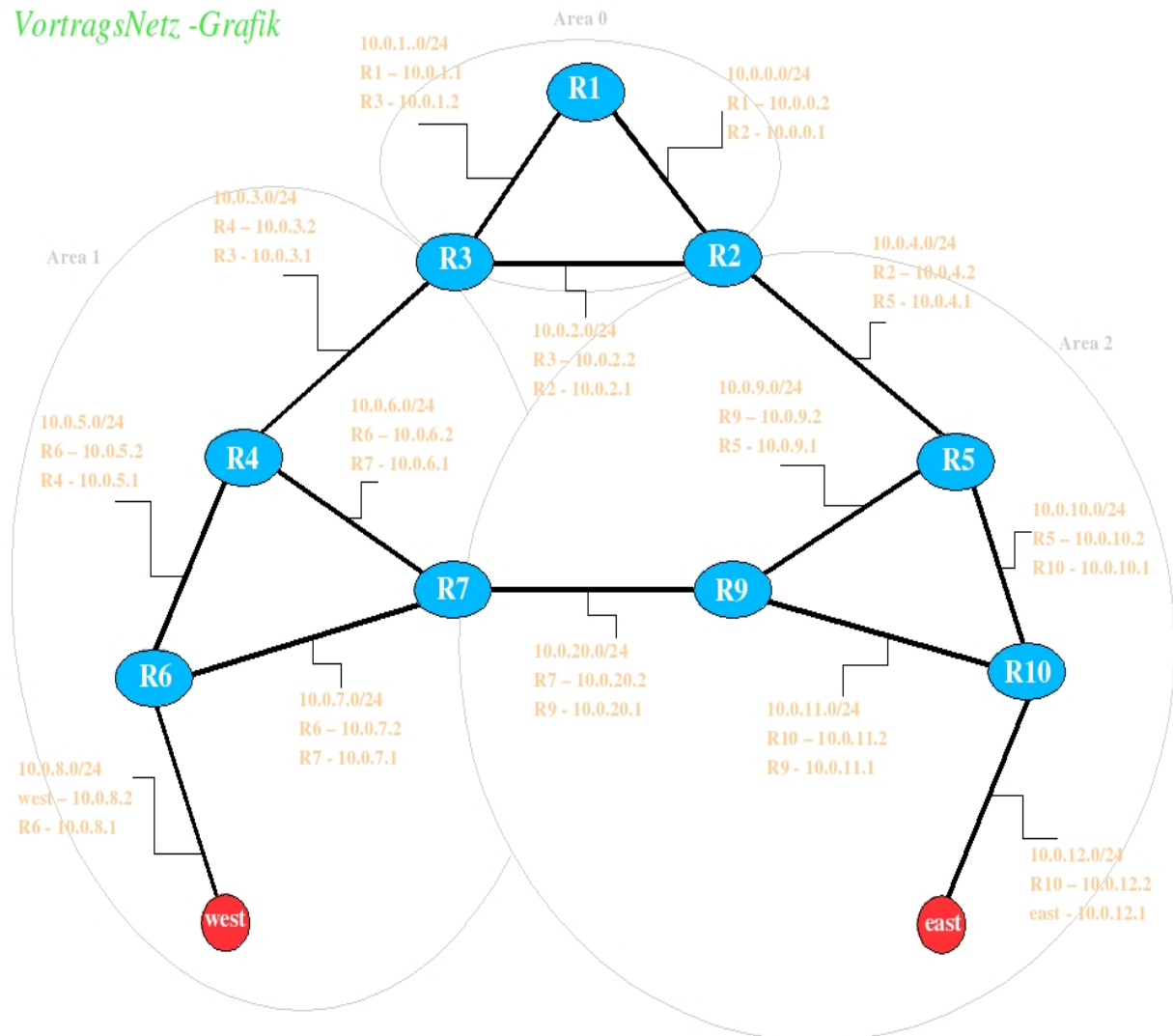
1	10.0.3.1 (10.0.3.1)	ms	ms	ms
2	10.0.2.1 (10.0.2.1)	ms	ms	ms
3	10.0.4.1 (10.0.4.1)	ms	ms	ms

Auch an dieser Stelle kann man den genauen Weg anhand der Grafik zurückverfolgen.

## 4. VortragsNetz

Um die Thematik etwas zu vertiefen, stellen wir nun noch unser zweites Netz vor. Es gelten hier natürlich die selben show-Befehle, sowie dieselbe Handhabung, wie oben bereits erwähnt.

### VortragsNetz -Grafik



### 4.1 Die Xml-Datei

Hier die von uns geschriebene XML-Datei zum obigen Schaubild, in der die grau geschriebenen Passagen als auskommentiert zu verstehen sind. Hierbei handelt es sich um die Router West und East, weil im Rahmen dieses Praktikums die sozusagen am Ende sich befindenden Router nicht auf einem Daemon laufen sollten.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/local/share/xml/vnuml/vnuml.dtd">

<vnuml>
  <global>
    <version>1.5</version>
    <simulation_name>VortragsNetz</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
    <automac/>
  </global>
</vnuml>
```

```

    <ip_offset>100</ip_offset>
    <host_mapping/>
</global>

<net name="BoneA" mode="uml_switch"/> <!--10.0.0.0/24-->
<net name="BoneB" mode="uml_switch"/> <!--10.0.1.0/24-->
<net name="BoneC" mode="uml_switch"/> <!--10.0.2.0/24-->
<net name="Core1" mode="uml_switch"/> <!--10.0.3.0/24-->
<net name="Core2" mode="uml_switch"/> <!--10.0.4.0/24-->
<net name="Core3" mode="uml_switch"/> <!--10.0.5.0/24-->
<net name="Core4" mode="uml_switch"/> <!--10.0.6.0/24-->
<net name="Core5" mode="uml_switch"/> <!--10.0.7.0/24-->
<net name="Core6" mode="uml_switch"/> <!--10.0.8.0/24-->
<net name="Core7" mode="uml_switch"/> <!--10.0.9.0/24-->
<net name="Core8" mode="uml_switch"/> <!--10.0.10.0/24-->
<net name="Core9" mode="uml_switch"/> <!--10.0.11.0/24-->
<net name="Core10" mode="uml_switch"/> <!--10.0.12.0/24-->
<net name="Core11" mode="uml_switch"/> <!--10.0.20.0/24-->

```

```

    <!--Backbone-->

```

```

<vm name="R1">

```

```

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

    <if id="1" net="BoneA">
        <ipv4 mask="255.255.255.0">10.0.0.2</ipv4>
    </if>
    <if id="2" net="BoneB">
        <ipv4 mask="255.255.255.0">10.0.1.1</ipv4>
    </if>

    <forwarding type="ip"/>

    <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R1</filetree>
    <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/*_filter; do echo 0 &gt; $f; done</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>

```

```

</vm>

```

```

<vm name="R2">

```

```

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

    <if id="1" net="BoneA">
        <ipv4 mask="255.255.255.0">10.0.0.1</ipv4>
    </if>
    <if id="2" net="BoneC">
        <ipv4 mask="255.255.255.0">10.0.2.1</ipv4>
    </if>
    <if id="3" net="Core2">
        <ipv4 mask="255.255.255.0">10.0.4.2</ipv4>
    </if>

    <forwarding type="ip"/>

    <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R2</filetree>
    <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/*_filter; do echo 0 &gt; $f; done</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>

```

```

</vm>

```

```

<vm name="R3">

```

```

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

```

```

<if id="1" net="BoneB">
    <ipv4 mask="255.255.255.0">10.0.1.2</ipv4>
</if>
<if id="2" net="BoneC">
    <ipv4 mask="255.255.255.0">10.0.2.2</ipv4>
</if>
<if id="3" net="Core1">
    <ipv4 mask="255.255.255.0">10.0.3.1</ipv4>
</if>

<forwarding type="ip"/>

<filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R3</filetree>
<exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
<exec seq="stop" type="verbatim">killall zebra</exec>
<exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

<!--Aarea1-->

<vm name="R4">

<filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
<mem>30M</mem>
<kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

<if id="1" net="Core1">
    <ipv4 mask="255.255.255.0">10.0.3.2</ipv4>
</if>
<if id="2" net="Core3">
    <ipv4 mask="255.255.255.0">10.0.5.1</ipv4>
</if>
<if id="3" net="Core4">
    <ipv4 mask="255.255.255.0">10.0.6.2</ipv4>
</if>

<forwarding type="ip"/>

<filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R4</filetree>
<exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
<exec seq="stop" type="verbatim">killall zebra</exec>
<exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

<vm name="R6">

<filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
<mem>30M</mem>
<kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

<if id="1" net="Core3">
    <ipv4 mask="255.255.255.0">10.0.5.2</ipv4>
</if>
<if id="2" net="Core5">
    <ipv4 mask="255.255.255.0">10.0.7.2</ipv4>
</if>
<if id="3" net="Core6">
    <ipv4 mask="255.255.255.0">10.0.8.1</ipv4>
</if>

<forwarding type="ip"/>

<filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R6</filetree>
<exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
<exec seq="stop" type="verbatim">killall zebra</exec>
<exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

```

```

<vm name="R7">

  <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

  <if id="1" net="Core4">
    <ipv4 mask="255.255.255.0">10.0.6.1</ipv4>
  </if>
  <if id="2" net="Core5">
    <ipv4 mask="255.255.255.0">10.0.7.1</ipv4>
  </if>
  <if id="3" net="Core11">
    <ipv4 mask="255.255.255.0">10.0.20.2</ipv4>
  </if>

  <forwarding type="ip"/>

  <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R7</filetree>
  <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
  <exec seq="stop" type="verbatim">killall zebra</exec>
  <exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

```

```

<vm name="West">

  <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

  <if id="1" net="Core6">
    <ipv4 mask="255.255.255.0">10.0.8.2</ipv4>
  </if>
  <route type="inet" gw="10.0.8.1">default</route>

  <forwarding type="ip"/>

  <!--
  <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R8</filetree>
  <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
  <exec seq="stop" type="verbatim">killall zebra</exec>
  <exec seq="stop" type="verbatim">killall ospfd</exec>
  -->

</vm>

```

<!--Aerea2-->

```

<vm name="R5">

  <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
  <mem>30M</mem>
  <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

  <if id="1" net="Core2">
    <ipv4 mask="255.255.255.0">10.0.4.1</ipv4>
  </if>
  <if id="2" net="Core7">
    <ipv4 mask="255.255.255.0">10.0.9.1</ipv4>
  </if>
  <if id="3" net="Core8">
    <ipv4 mask="255.255.255.0">10.0.10.2</ipv4>
  </if>

  <forwarding type="ip"/>

  <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R5</filetree>
  <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
  <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>

```

```

    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

<vm name="R9">

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

    <if id="1" net="Core7">
        <ipv4 mask="255.255.255.0">10.0.9.2</ipv4>
    </if>
    <if id="2" net="Core9">
        <ipv4 mask="255.255.255.0">10.0.11.2</ipv4>
    </if>
    <if id="3" net="Core11">
        <ipv4 mask="255.255.255.0">10.0.20.1</ipv4>
    </if>

    <forwarding type="ip"/>

    <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R9</filetree>
    <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

<vm name="R10">

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

    <if id="1" net="Core8">
        <ipv4 mask="255.255.255.0">10.0.10.1</ipv4>
    </if>
    <if id="2" net="Core9">
        <ipv4 mask="255.255.255.0">10.0.11.1</ipv4>
    </if>
    <if id="3" net="Core10">
        <ipv4 mask="255.255.255.0">10.0.12.2</ipv4>
    </if>

    <forwarding type="ip"/>

    <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum-Beispiele/VortragsNetz/R10</filetree>
    <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
    <exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
    <exec seq="stop" type="verbatim">killall zebra</exec>
    <exec seq="stop" type="verbatim">killall ospfd</exec>

</vm>

<vm name="East">

    <filesystem type="cow">/usr/local/share/vnuml/filesystems/root_fs_tutorial-0.2.3</filesystem>
    <mem>30M</mem>
    <kernel>/usr/local/share/vnuml/kernels/linux-2.6.10-1m/linux-2.6.10-1m</kernel>

    <if id="1" net="Core10">
        <ipv4 mask="255.255.255.0">10.0.12.1</ipv4>
    </if>

    <route type="inet" gw="10.0.12.2">default</route>

    <forwarding type="ip"/>

    <!--
    <filetree root="/etc/zebra" when="start">/home/linblumi/Praktikum
    -Beispiele/VortragsNetz/R11</filetree>
    <exec seq="start" type="verbatim">for f in /proc/sys/net/ipv4/conf/*/rp_filter; do echo 0 &gt; $f; done</exec>

```

```

<exec seq="start" type="verbatim">/usr/local/sbin/zebra -f /etc/zebra/zebra.conf -d</exec>
<exec seq="start" type="verbatim">/usr/local/sbin/ospfd -f /etc/zebra/ospfd.conf -d</exec>
<exec seq="stop" type="verbatim">killall zebra</exec>
<exec seq="stop" type="verbatim">killall ospfd</exec>
-->

```

```
</vm>
```

```
</vnuml>
```

## 4.2 Die config-Dateien

Als Grundlage für unsere Konfigurationsdateien, ospfd.conf, zebra.conf und vtysh.conf, haben wir die bei Zebra enthaltenen Beispieldateien benutzt und angepasst auf unsere Bedürfnisse im Rahmen dieses Praktikums. Die nicht genutzten Interfaces haben wir im Rahmen dieses Praktikum nicht weiter beleuchtet. Für grössere Netze, wie hier, bietet sich noch der Befehl `ospf router-id` an, um jedem Router eine feste ip-Adresse zuzuweisen, die höchste hier wäre jeweils `x.x.x.255` aber diese wird als Multicast-Adresse benutzt und somit haben wir uns für die `x.x.x.200` entschieden.

In der Backbone, auch Area0, befinden sich folgende Router:

R1:

Die ospfd.conf:

```

!
hostnameR1
password xxxx
log file /tmp/ospfd.log
log stdout
!
!
debug ospf packet all send
!
! interface dummy0
!
interface eth1
 ip ospf cost 30
!
interface eth2
 ip ospf cost 10
!
! interface eth3
!
! interface gre0
!
! interface lo
!
! interface sit0
!
! interface teql0
!
! interface tunl0
!
router ospf
 ospf router-id 10.0.0.200
! ospf rfc1583compatibility
 network 10.0.0.0/24 area 0.0.0.0

```

```

network 10.0.1.0/24 area 0.0.0.0
!
line vty
!
```

Der Befehl **router ospf** macht R1 erst zum Ospf-Router. Unter dem werden dann die Netzwerke entsprechend der XML-Datei deklariert, mit ihrer Area Zugehörigkeit. Die Intherfacebeschreibung **interface** entspricht ebenfalls der Vergabe der Ids in der XML-Datei, d.h. id=1 ist hier eth1. Die Kostenzuweisung erfolgt direkt an jedem Interface(**ip ospf cost**). Wie zu sehen, durch den Befehl **ospf router-id** kann man nun die ID des Routers festlegen. Dies taucht dann später in der database und in anderen ospf-show-Befehlen auf und dient der besseren Orientierung. Man sollte auch einen Zusammenhang mit den angeschlossenen Netzen herstellen, darum entsprechen die Ids nicht immer den Routernamen.

Die zebra.conf:

```

!
hostname R1
password xxxx
! enable password zebra
!
! Interface's description.
!
! interface lo
!  description test of desc.
!
! interface sit0
!  multicast
!
! Static default route sample.
!
! ip route 0.0.0.0/0 203.181.89.241
!
!
log file zebra.log
log file /tmp/zebra.log
!
```

Die vtysh.conf:

```

!
! vtysh sample configuratin file
!
! username kunihiro nopassword
log file /var/log/zebra/vtysh.log
!
```

Im Folgenden nun beschäftigen wir uns nur noch mit der ospfd.conf, da die zebra.conf und die vtysh.conf die Funktionalität des Netzwerkes nicht behindern bzw. Änderungen, im speziellen in der zebra.conf, stets zu Fehlermeldungen führten und die vtysh.conf momentan keine weiteren Befehle besitzt bzw. nicht implementiert sind.

R2:

```

!
hostname R2
password xxxx
log file /tmp/ospfd.log
```

```

log stdout
!
debug ospf packet all send
!
!
! interface dummy0
!
interface eth1
  ip ospf cost 20
!
interface eth2
  ip ospf cost 40
!
interface eth3
  ip ospf cost 50
!
!
! interface gre0
!
! interface lo
!
! interface sit0
!
! interface teql0
!
! interface tunl0
!
router ospf
  ospf router-id 10.0.2.200
! ospf rfc1583compatibility
  network 10.0.0.0/24 area 0.0.0.0
  network 10.0.2.0/24 area 0.0.0.0
  network 10.0.4.0/24 area 2
!
line vty
!

```

R3:

```

!
hostname R3
  password xxxx
  log file /tmp/ospfd.log
  log stdout
!
  debug ospf packet all recv
!
!
!interface dummy0
!
interface eth1
  ip ospf cost 30
!
interface eth2
  ip ospf cost 10
!
interface eth3
  ip ospf cost 50
!
!interface gre0
!
!interface lo

```

```

!
!interface sit0
!
!interface teql0
!
!interface tunl0
!
router ospf
  ospf router-id 10.0.1.200
! ospf rfc1583compatibility
  network 10.0.1.0/24 area 0.0.0.0
  network 10.0.2.0/24 area 0.0.0.0
  network 10.0.3.0/24 area 1
!
line vty
!

```

Wie vielleicht schon gemerkt haben wir dieses Netz aus dem zuvor schon vorgestellten EasyOspf-Netz entwickelt und es um zwei richtige Areas erweitert, um das Area-Prinzip zu nutzen. Die Definition von den Areas erfolgt durch das Zuordnen dieser an die Netze, die an einem Router angeschlossen sind. Dabei muss man beachten, dass ein Netz nicht auf der einen Seite eine andere Area-ID zugewiesen bekommt als auf der anderen. Nun zu Area1 mit folgenden Routern:

```

R4:
!
  hostname R4
  password xxxx
  log file /tmp/ospfd.log
  log stdout
!
  debug ospf packet all recv
!
!
! interface dummy0
!
interface eth1
  ip ospf cost 40
!
interface eth2
  ip ospf cost 20
!
interface eth3
  ip ospf cost 30
!
! interface gre0
!
! interface lo
!
! interface sit0
!
! interface teql0
!
! interface tunl0
!
router ospf
  ospf router-id 10.0.3.200
! ospf rfc1583compatibility
  network 10.0.3.0/24 area 1
  network 10.0.5.0/24 area 1

```

```
network 10.0.6.0/24 area 1
!  
line vty
!
```

R6

```
!  
hostname R6  
password xxxx  
log file /tmp/ospfd.log  
log stdout  
!  
debug ospf packet all recv  
!  
!interface dummy0  
!  
interface eth1  
ip ospf cost 10  
!  
interface eth2  
ip ospf cost 30  
!  
interface eth3  
ip ospf cost 60  
!  
! interface gre0  
!  
! interface lo  
!  
! interface sit0  
!  
! interface teql0  
!  
! interface tunl0  
!  
router ospf  
ospf router-id 10.0.5.200  
! ospf rfc1583compatibility  
network 10.0.5.0/24 area 1  
network 10.0.7.0/24 area 1  
network 10.0.8.0/24 area 1  
!  
line vty  
!
```

R7:

```
!  
hostname R7  
password xxxx  
log file /tmp/ospfd.log  
log stdout  
!  
debug ospf packet all send  
!  
!interface dummy0  
!  
interface eth1  
ip ospf cost 20  
!  
interface eth2
```

```

    ip ospf cost 10
    !
interface eth3
    ip ospf cost 70
    !
!interface gre0
!
!interface lo
!
!interface sit0
!
!interface teql0
!
!interface tunl0
!
router ospf
    ospf router-id 10.0.6.200
! ospf rfc1583compatibility
    network 10.0.6.0/24 area 1
    network 10.0.7.0/24 area 1
    network 10.0.20.0/24 area 2
!
line vty
!

```

West:

Wie oben schon erwähnt sollte auf dem Router West kein Daemon laufen. Wer es dennoch ausprobieren möchte, braucht nur in der XML-Datei die Kommentarzeichen löschen. Die Konfigurationsdateien befinden sich im Ordner R8 und sind schon angepasst. (Vorsicht, um die Änderungen in der XML-Datei wirksam werden zu lassen, muss das Szenario gestoppt bzw. heruntergefahren werden, um es dann wieder neu zu starten)

Area2 besitzt folgende Router:

R5:

```

!
hostname R5
password xxxx
log file /tmp/ospfd.log
log stdout
!
debug ospf packet all send
!
!interface dummy0
!
interface eth1
    ip ospf cost 20
!
interface eth2
    ip ospf cost 10
!
interface eth3
    ip ospf cost 10
!
!interface gre0
!
!interface lo

```

```
!  
!interface sit0  
!  
!interface teql0  
!  
!interface tunl0  
!  
router ospf  
  ospf router-id 10.0.4.200  
! ospf rfc1583compatibility  
  network 10.0.4.0/24 area 2  
  network 10.0.9.0/24 area 2  
  network 10.0.10.0/24 area 2  
!  
line vty  
!
```

R9:

```
!  
hostname R9  
password xxxx  
log file /tmp/ospfd.log  
log stdout  
!  
debug ospf packet all send  
!  
!interface dummy0  
!  
interface eth1  
  ip ospf cost 10  
!  
interface eth2  
  ip ospf cost 30  
!  
interface eth3  
  ip ospf cost 50  
!  
!interface gre0  
!  
!interface lo  
!  
!interface sit0  
!  
!interface teql0  
!  
!interface tunl0  
!  
router ospf  
  ospf router-id 10.0.9.200  
! ospf rfc1583compatibility  
  network 10.0.9.0/24 area 2  
  network 10.0.11.0/24 area 2  
  network 10.0.20.0/24 area 2  
!  
line vty  
!
```

R10:

```
!  
hostname R10
```

```

password xxxx
log file /tmp/ospfd.log
log stdout
!
debug ospf packet all send
!
! interface dummy0
!
interface eth1
 ip ospf cost 20
!
interface eth2
 ip ospf cost 30
!
interface eth3
 ip ospf cost 40
!
!interface gre0
!
!interface lo
!
!interface sit0
!
!interface teql0
!
!interface tunl0
!
router ospf
 ospf router-id 10.0.10.200
! ospf rfc1583compatibility
 network 10.0.10.0/24 area 2
 network 10.0.11.0/24 area 2
 network 10.0.12.0/24 area 2
!
line vty
!
```

East:

Für den Router East gilt das gleiche, wie oben bei West schon beschrieben. Hier befinden sich die Konfigurationsdateien im Ordner R11.

Nun eine Zusammenfassung der Router-ids und die Area-Zugehörigkeit:

Router	Id	Area
R1	10.0.0.200	0
R2	10.0.2.200	0 und 2 somit ABR
R3	10.0.1.200	0 und 1 somit ABR
R4	10.0.3.200	1
R5	10.0.4.200	2
R6	10.0.5.200	1
R7	10.0.6.200	1 und 2 somit ABR
R9	10.0.9.200	2
R10	10.0.10.200	2

West gehört zur Area1 und East zur Area2.

Zum Abschluss nun noch zwei Beispiele zur Database eines Routers in einer Area (R10) und eines ABR (R3).

```
R10> show ip ospf database
```

```
OSPF Router with ID (10.0.10.200)
```

```
Router Link States (Area 0.0.0.2)
```

Link ID	ADV Router	Age	Seq#	CkSum	Link count
10.0.2.200	10.0.2.200	45	0x8000000a	0xd975	1
10.0.4.200	10.0.4.200	46	0x80000009	0x6b7a	3
10.0.6.200	10.0.6.200	47	0x80000009	0x5eb5	1
10.0.9.200	10.0.9.200	45	0x80000009	0xccb9	3
10.0.10.200	10.0.10.200	44	0x80000009	0x3d6f	3

```
Net Link States (Area 0.0.0.2)
```

Link ID	ADV Router	Age	Seq#	CkSum
10.0.4.1	10.0.4.200	51	0x80000007	0x576b
10.0.9.2	10.0.9.200	50	0x80000007	0x4e62
10.0.10.1	10.0.10.200	49	0x80000007	0x3b73
10.0.11.1	10.0.10.200	49	0x80000007	0x7137
10.0.20.1	10.0.9.200	50	0x80000007	0xe4bf

```
Summary Link States (Area 0.0.0.2)
```

Link ID	ADV Router	Age	Seq#	CkSum	Route
10.0.0.0	10.0.2.200	844	0x80000007	0x88db	10.0.0.0/24
10.0.1.0	10.0.2.200	854	0x80000007	0xe177	10.0.1.0/24
10.0.2.0	10.0.2.200	1526	0x80000006	0x3d12	10.0.2.0/24
10.0.3.0	10.0.2.200	494	0x80000007	0xc163	10.0.3.0/24
10.0.3.0	10.0.6.200	900	0x80000007	0xdc58	10.0.3.0/24
10.0.5.0	10.0.2.200	1496	0x80000006	0x7699	10.0.5.0/24
10.0.5.0	10.0.6.200	1300	0x80000006	0x3725	10.0.5.0/24
10.0.6.0	10.0.2.200	474	0x80000007	0xcd36	10.0.6.0/24
10.0.6.0	10.0.6.200	1701	0x80000006	0x2c2f	10.0.6.0/24
10.0.7.0	10.0.2.200	544	0x80000007	0x27d1	10.0.7.0/24
10.0.7.0	10.0.6.200	860	0x80000007	0xbaa8	10.0.7.0/24
10.0.8.0	10.0.2.200	1134	0x80000006	0xaf21	10.0.8.0/24
10.0.8.0	10.0.6.200	129	0x80000007	0x0a1c	10.0.8.0/24

Wie oben zu sehen ist verfügt der Router 10 nur über Einträge aus der Area2 in der er sich befindet. Die Router Link States Tabelle gibt an, welche Router zu dieser Area gehören. Der Link Count zählt die Anzahl der Netze mit denen die jeweiligen Router innerhalb der Area verbunden sind. Die Net Link States Tabelle zählt alle zur Area gehörigen Netze auf und auch welchem Router sie zugeordnet sind. In der Summary Link States Tabelle stehen nun alle sich außerhalb der Area befindlichen Netze und über welchen Router, in dem Fall über welche ABRs, er dort hingelangt. Die Age Angabe verdeutlicht den Aktualisierungs- und Updateprozess.

```
R3> show ip ospf database
```

```
OSPF Router with ID (10.0.1.200)
```

```
Router Link States (Area 0.0.0.0)
```

Link ID	ADV Router	Age	Seq#	CkSum	Link count
10.0.0.200	10.0.0.200	1667	0x80000007	0xda66	2
10.0.1.200	10.0.1.200	1665	0x80000007	0x1e1b	2
10.0.2.200	10.0.2.200	1667	0x80000007	0x9691	2

Net Link States (Area 0.0.0.0)

Link ID	ADV Router	Age	Seq#	CkSum
10.0.0.1	10.0.2.200	1672	0x80000005	0x6965
10.0.1.2	10.0.1.200	1670	0x80000005	0x5678
10.0.2.1	10.0.2.200	1667	0x80000005	0x606b

Summary Link States (Area 0.0.0.0)

Link ID	ADV Router	Age	Seq#	CkSum	Route
10.0.3.0	10.0.1.200	1042	0x80000005	0x9fa6	10.0.3.0/24
10.0.4.0	10.0.2.200	1444	0x80000005	0x8db6	10.0.4.0/24
10.0.5.0	10.0.1.200	611	0x80000006	0x50de	10.0.5.0/24
10.0.6.0	10.0.1.200	1342	0x80000005	0xab79	10.0.6.0/24
10.0.7.0	10.0.1.200	831	0x80000005	0x0515	10.0.7.0/24
10.0.8.0	10.0.1.200	79	0x80000006	0x8966	10.0.8.0/24
10.0.9.0	10.0.2.200	93	0x80000006	0xb87b	10.0.9.0/24
10.0.10.0	10.0.2.200	1264	0x80000005	0xaf84	10.0.10.0/24
10.0.11.0	10.0.2.200	283	0x80000006	0xcf44	10.0.11.0/24
10.0.12.0	10.0.2.200	1374	0x80000005	0x2bde	10.0.12.0/24
10.0.20.0	10.0.2.200	343	0x80000006	0x35c1	10.0.20.0/24

Router Link States (Area 0.0.0.1)

Link ID	ADV Router	Age	Seq#	CkSum	Link count
10.0.1.200	10.0.1.200	1665	0x80000007	0xd77e	1
10.0.3.200	10.0.3.200	1665	0x80000007	0x9334	3
10.0.5.200	10.0.5.200	1666	0x80000007	0xb809	3
10.0.6.200	10.0.6.200	1665	0x80000007	0x32f5	2

Net Link States (Area 0.0.0.1)

Link ID	ADV Router	Age	Seq#	CkSum
10.0.3.2	10.0.3.200	1670	0x80000005	0x5572
10.0.5.2	10.0.5.200	1670	0x80000005	0x5d62
10.0.6.1	10.0.6.200	1670	0x80000005	0x5667
10.0.7.1	10.0.6.200	1669	0x80000005	0x6555

Summary Link States (Area 0.0.0.1)

Link ID	ADV Router	Age	Seq#	CkSum	Route
10.0.0.0	10.0.1.200	1042	0x80000005	0xf765	10.0.0.0/24
10.0.1.0	10.0.1.200	781	0x80000006	0xea70	10.0.1.0/24
10.0.2.0	10.0.1.200	721	0x80000006	0x1757	10.0.2.0/24
10.0.4.0	10.0.1.200	1072	0x80000005	0xf842	10.0.4.0/24
10.0.4.0	10.0.6.200	1270	0x80000005	0x67a6	10.0.4.0/24
10.0.9.0	10.0.1.200	300	0x80000006	0x2407	10.0.9.0/24
10.0.9.0	10.0.6.200	289	0x80000006	0x65b6	10.0.9.0/24
10.0.10.0	10.0.1.200	941	0x80000005	0x1b10	10.0.10.0/24
10.0.10.0	10.0.6.200	1400	0x80000005	0xc051	10.0.10.0/24
10.0.11.0	10.0.1.200	1852	0x80000005	0x3dce	10.0.11.0/24
10.0.11.0	10.0.6.200	1300	0x80000005	0x1aec	10.0.11.0/24
10.0.12.0	10.0.1.200	390	0x80000006	0x946b	10.0.12.0/24
10.0.12.0	10.0.6.200	1149	0x80000005	0x3cab	10.0.12.0/24
10.0.20.0	10.0.1.200	229	0x80000006	0xa04d	10.0.20.0/24
10.0.20.0	10.0.6.200	1470	0x80000005	0x8992	10.0.20.0/24

Wie oben schon erwähnt, hat der ABR R3 die Einträge von zwei Netzen in seiner Database, welche er nun an beide, an ihn angeschlossenen Netze weitergibt. Dadurch aktualisiert jeder Router seine Routingtabelle, um das Wissen von Netzen auch außerhalb seiner eigenen Area.

Beispiel R10:

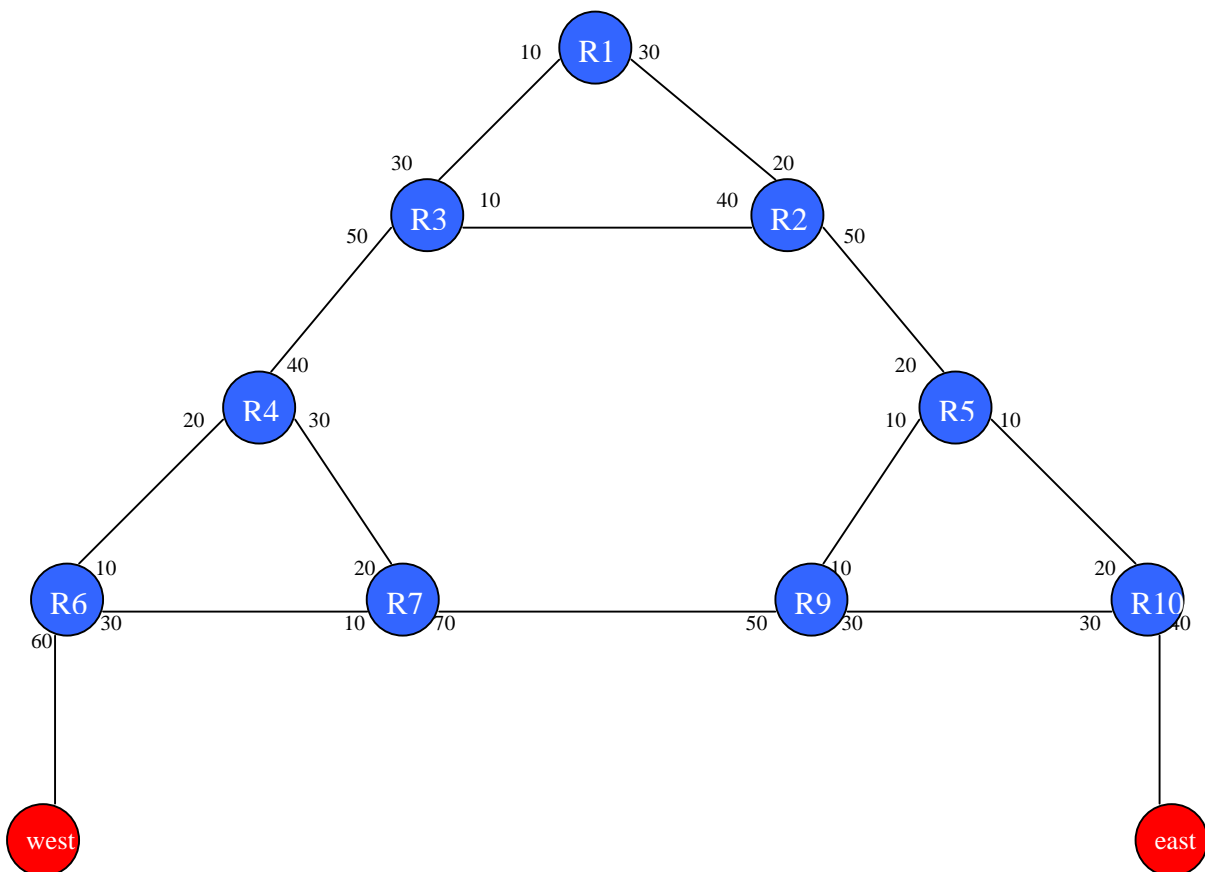
```
R10> show ip ospf route
===== OSPF network routing table =====
N IA 10.0.0.0/24      [60] area: 0.0.0.2
                    via 10.0.10.2, eth1
N IA 10.0.1.0/24     [70] area: 0.0.0.2
                    via 10.0.10.2, eth1
N IA 10.0.2.0/24     [80] area: 0.0.0.2
                    via 10.0.10.2, eth1
N IA 10.0.3.0/24     [120] area: 0.0.0.2
                    via 10.0.10.2, eth1
N   10.0.4.0/24      [40] area: 0.0.0.2
                    via 10.0.10.2, eth1
N IA 10.0.5.0/24     [100] area: 0.0.0.2
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
N IA 10.0.6.0/24     [100] area: 0.0.0.2
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
N IA 10.0.7.0/24     [90] area: 0.0.0.2
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
N IA 10.0.8.0/24     [150] area: 0.0.0.2
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
N   10.0.9.0/24      [30] area: 0.0.0.2
                    via 10.0.10.2, eth1
N   10.0.10.0/24     [20] area: 0.0.0.2
                    directly attached to eth1
N   10.0.11.0/24     [30] area: 0.0.0.2
                    directly attached to eth2
N   10.0.12.0/24     [40] area: 0.0.0.2
                    directly attached to eth3
N   10.0.20.0/24     [80] area: 0.0.0.2
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
===== OSPF router routing table =====
R   10.0.2.200       [40] area: 0.0.0.2, ABR
                    via 10.0.10.2, eth1
R   10.0.6.200       [80] area: 0.0.0.2, ABR
                    via 10.0.11.2, eth2
                    via 10.0.10.2, eth1
===== OSPF external routing table =====
```

Dieses Beispiel soll zeigen, wie eine vollständige Routingtabelle nach Abschluss des Database-Austausches aussieht.

### 4.3 Kosten und deren Einfluß

Jetzt kommen wir zur Kostenverteilung im Speziellen. Die Kosten bei Ospf berechnen sich wie folgt:  $\text{cost} = 100.000.000/\text{bandwidth in bps}$ , so ergibt sich für eine 10M Ethernet-Leitung die Kosten von 10 und für eine T1-Leitung die Kosten von 64 (Gleichung aus dem „CISCO OSPF DESIGN GUIDE“ <http://www.cisco.com/warp/public/104/1.html>).

Wir benutzen hier aber Zebra und wir haben die Möglichkeit die Kosten direkt in der ospfd.conf an den Interfaces zu setzen. Hier nun eine Grafik, nur mit der Kostenverteilung, wie wir sie in den Konfigurationsdateien gesetzt haben.



Der Ospf-Algorithmus sucht sich nun anhand der verteilten Kosten den billigsten Weg. Ospf bedeutet zwar, Open Shortest Path First, also sollte man annehmen den „kürzesten“ aber hier ist mit dem Kürzesten der Billigste gemeint. Auch wenn von West nach East der weg über R6, R7, R9 und R10 physikalisch der kürzeste Weg ist, so ist er aber nicht der billigste Weg durch dieses Netz. Hierbei gibt es zu beachten, dass der ausgehende Port über die Kosten der Route entscheidet, nicht der Eingehende.

Überlegen wir jetzt mal wie man von West nach East kommt.

Als Erstes R6 mit 30 nach R7, von da mit 70 nach R9 und wiederum mit 30 nach R10, wo dann noch 40 nach East verbleiben, macht zusammen 170. Der Nächste Weg wäre R6 mit 10 zu R4 und von dort mit 30 zu R7, dann wieder 70 aber wir sind jetzt schon „teurer“ als der

erste Versuch, weil wir schon 40 haben bevor wir zu R7 kommen(zuvor nur 30). Gehen wir nun bis R9 (100) wie im ersten Versuch. Nun folgen wir aber den Weg nach R5 mit 10 und von dort mit 10 zu R10. Plus den Rest ergibt sich nun 160. Dieser Weg wäre nun billiger als der Erste.

Aber der Traceroute von West nach East ergibt folgende Ausgabe:

```
West:~# traceroute 10.0.12.1
  traceroute to 10.0.12.1 (10.0.12.1), 30 hops max, 38 byte packets
  1 10.0.8.1 (10.0.8.1) 2.006 ms 4.964 ms 3.895 ms
  2 10.0.5.1 (10.0.5.1) 32.720 ms 5.768 ms 3.158 ms
  3 10.0.3.1 (10.0.3.1) 25.642 ms 6.888 ms 3.552 ms
  4 10.0.0.1 (10.0.0.1) 44.710 ms 7.618 ms 5.807 ms
  5 10.0.9.1 (10.0.9.1) 97.160 ms 7.506 ms 5.892 ms
  6 10.0.11.1 (10.0.11.1) 28.248 ms 9.032 ms 5.040 ms
  7 10.0.12.1 (10.0.12.1) 42.353 ms 12.292 ms 6.150 ms
```

Die Frage ist nun: Warum geht er diesen Weg?

Erst mal schauen wir ob der Weg günstiger ist. Er verläuft wie folgt: von R6 mit 10 nach R4, von dort mit 40 nach R3, von dort mit 10 nach R2, weiter geht's mit 50 nach R5 und von da aus mit 10 zu R10, wo noch 40 bis East verbleiben. Dies macht zusammen auch 160. Nun haben wir zwei Wege, die die Kosten von 160 haben. Nun kann man sich fragen warum nimmt er beim Routing nicht den unteren Weg, welcher auch noch um einen Hop kürzer wäre. Die Antwort darauf liegt in der Eigenschaft von Ospf selbst, immer den besseren, also kürzeren, Weg zu gehen. Denn auf diesem, von ihm gewählten Weg sind alle Netze, von den Kosten her, kleiner als die 70 bei R7. Desweiteren fällt auf, dass bis Schritt 3 immer der „richtige“ Port angezeigt wird, danach es aber etwas schwer fällt den genauen Weg zu verfolgen. Die Antwort hier ist auch wieder die Kostenverteilung an den Interfaces, denn der Port 10.0.0.1 liegt an R2, um den Weg anzuzeigen, müsste R2 aber über den Port 10.0.2.1 antworten. Dies geschieht aber nicht, weil R2 aus Kostengründen über R1 routet und somit seine Antwort über den in Punkt 4 genannten Port schickt. Genauso verhält es sich mit der Ausgabe bei den Punkten 5 für R5 und 6 für R10

Als zweites Beispiel sehen wir uns noch den Traceroute von East nach West an. Dieser ergibt:

```
East:~# traceroute 10.0.8.2
  traceroute to 10.0.8.2 (10.0.8.2), 30 hops max, 38 byte packets
  1 10.0.12.2 (10.0.12.2) 3.984 ms 9.684 ms 2.605 ms
  2 10.0.9.2 (10.0.9.2) 26.295 ms 6.474 ms 4.034 ms
  3 10.0.20.2 (10.0.20.2) 39.113 ms 6.740 ms 4.506 ms
  4 10.0.5.2 (10.0.5.2) 10.463 ms 8.083 ms 6.148 ms
  5 10.0.8.2 (10.0.8.2) 6.436 ms 8.490 ms 6.178 ms
```

#### 4.4 Die debug-Ausgabe

Zum Abschluss nun noch ein kurzer Blick auf die ospfd.log-Datei. Wie weiter oben schon erwähnt wird dies durch die log-Befehle in der ospfd.conf gesetzt. Durch den Befehl **debug ospf packet** wird festgelegt welche Pakete man in die log-Datei schreiben möchte. Hier kann man auch entscheiden, welche Pakete es sein sollen, z. Bsp. nur die Hello-Pakete mit **hello** oder die Link State Update-Pakete mit **ls-update** und zum Schluss dann noch ob es die jeweils eingehenden oder ausgehenden Pakete sein sollen, mit **send** bzw. **recv**.

Als Beispiel: **debug ospf packet hello send** heißt alle von diesem Router gesendeten Hello-Pakete werden aufgezeichnet.

Standardmäßig ist auf jeder Maschine ein tmp-Verzeichnis vorhanden. Um dorthin zu gelangen, muss man zuerst sich auf die Maschine einloggen mit ssh. Dann geht man mit `cd /tmp` in das Verzeichnis und mit `vi ospfd.log` wird dann die log-Datei auf die Konsole ausgegeben. Die Tastenkombination `Strg+z` führt einen wieder in das Verzeichnis zurück.

Beispiel an R10(ein kurzer Auszug):

Hier haben wir uns alle gesendeten Pakete aufzeichnen lassen.

```
2005/09/08 15:15:38 OSPF: interface 10.0.10.1 join AllSPFRouters Multicast group.
2005/09/08 15:15:38 OSPF: interface 10.0.11.1 join AllSPFRouters Multicast group.
2005/09/08 15:15:38 OSPF: interface 10.0.12.2 join AllSPFRouters Multicast group.
```

Jetzt werden alle angeschlossenen Router in die SPF Multicast Gruppe aufgenommen.

```
2005/09/08 15:15:39 OSPF: Hello sent to [224.0.0.5] via [eth1:10.0.10.1].
2005/09/08 15:15:39 OSPF: Hello sent to [224.0.0.5] via [eth2:10.0.11.1].
2005/09/08 15:15:39 OSPF: Hello sent to [224.0.0.5] via [eth3:10.0.12.2].
2005/09/08 15:15:41 OSPF: Hello sent to [224.0.0.5] via [eth1:10.0.10.1].
2005/09/08 15:15:49 OSPF: Hello sent to [224.0.0.5] via [eth2:10.0.11.1].
2005/09/08 15:15:49 OSPF: Hello sent to [224.0.0.5] via [eth3:10.0.12.2].
2005/09/08 15:15:51 OSPF: Hello sent to [224.0.0.5] via [eth1:10.0.10.1].
2005/09/08 15:15:59 OSPF: Hello sent to [224.0.0.5] via [eth2:10.0.11.1].
2005/09/08 15:15:59 OSPF: Hello sent to [224.0.0.5] via [eth3:10.0.12.2].
2005/09/08 15:16:01 OSPF: Hello sent to [224.0.0.5] via [eth1:10.0.10.1].
2005/09/08 15:16:09 OSPF: Hello sent to [224.0.0.5] via [eth2:10.0.11.1].
2005/09/08 15:16:09 OSPF: Hello sent to [224.0.0.5] via [eth3:10.0.12.2].
2005/09/08 15:16:11 OSPF: Hello sent to [224.0.0.5] via [eth1:10.0.10.1].
```

Hier sind die gesendeten Hello-Pakete zu sehen. Ospf benutzt dazu eine eigene IP-Adresse [224.0.0.5].

```
2005/09/08 15:16:16 OSPF: Packet[DD]: Neighbor state is 2-Way, packet discarded.
2005/09/08 15:16:17 OSPF: Packet[DD]: Neighbor state is 2-Way, packet discarded.
```

Hier beginnt nun der Databaseaustausch.

```
2005/09/08 15:16:18 OSPF: DR-Election[1st]: Backup 10.0.10.1
2005/09/08 15:16:18 OSPF: DR-Election[1st]: DR 10.0.10.1
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: Backup 10.0.10.2
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: DR 10.0.10.1
2005/09/08 15:16:18 OSPF: interface 10.0.10.1 join AllDRouters Multicast group.
2005/09/08 15:16:18 OSPF: Database Description sent to [10.0.10.2] via [eth1:10.0.10.1].
2005/09/08 15:16:18 OSPF: DR-Election[1st]: Backup 10.0.11.1
2005/09/08 15:16:18 OSPF: DR-Election[1st]: DR 10.0.11.1
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: Backup 10.0.11.2
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: DR 10.0.11.1
2005/09/08 15:16:18 OSPF: interface 10.0.11.1 join AllDRouters Multicast group.
2005/09/08 15:16:18 OSPF: Database Description sent to [10.0.11.2] via [eth2:10.0.11.1].
2005/09/08 15:16:18 OSPF: Packet[DD]: Negotiation done (Master).
2005/09/08 15:16:18 OSPF: DR-Election[1st]: Backup 10.0.12.2
2005/09/08 15:16:18 OSPF: DR-Election[1st]: DR 10.0.12.2
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: Backup 0.0.0.0
2005/09/08 15:16:18 OSPF: DR-Election[2nd]: DR 10.0.12.2
2005/09/08 15:16:18 OSPF: interface 10.0.12.2 join AllDRouters Multicast group.
2005/09/08 15:16:18 OSPF: Packet[DD]: Negotiation done (Master).
```

2005/09/08 15:16:18 OSPF: Database Description sent to [10.0.10.2] via [eth1:10.0.10.1].  
2005/09/08 15:16:18 OSPF: Database Description sent to [10.0.11.2] via [eth2:10.0.11.1].  
2005/09/08 15:16:18 OSPF: Database Description sent to [10.0.10.2] via [eth1:10.0.10.1].

Am Ende dieses Vorgangs erhält R10 dann seine Routingtabelle und auch seine aktualisierte Database, außerdem wird festgelegt welcher Router den Status Backup oder DR bekommt. Letztendlich fällt der Port 10.0.12.1 aus der Multicast Gruppe, weil East ja kein Ospf-router ist, d.h. R10 sendet an East keine DD(Database Description)-Pakete.

## 5. Fazit

Ospf ist ein sehr interessantes und auch effizientes Protokoll und besitzt gegenüber anderen Protokollen mehrere Vorteile, schneller, konvergiert schneller und erlaubt größere Netze. Trotzdem gibt es auch hier leider keine richtige einheitliche Norm für die Deklarationsbefehle, siehe Zebra und Cisco beispielsweise. Darüber hinaus, was wir hier behandelt haben, gibt es noch viel mehr Eigenschaften und Einstellungen für Ospf, die wir in diesem Praktikum nicht alle erfassen konnten. Zebra bietet aber einen guten Einstieg in dieses Metier aber es fehlt stellenweise noch der volle Leistungsumfang, soll heißen dass nicht alle unter <http://www.zebra.org/zebra/index.html> aufgelisteten Befehle auch implementiert sind. Beim Erstellen eines Ospf-Netzes, wie auch bei anderen Programmierungen, sollte man darauf achten den richtigen Zusammenhang zwischen der XML und der ospfd.conf-Datei bereitzustellen. Denn Syntaxfehler werden gemeldet aber leider keine semantischen Fehler und so kann eine Fehlersuche mitunter sehr lange dauern.

## 6. Quellen

[www.zebra.org](http://www.zebra.org) die Zebra Seite

[www.cisco.com](http://www.cisco.com) die Cisco Seite

<http://jungla.dit.upm.es/~vnuml> die VNUML Seite

und Fachliteratur