Cisco MPLS VPN

Wolfgang Hurst
https://whurst.net

1 Einleitung

1.1 Vorwort

Dieses Dokument ist ein Abbild meiner alten Webseite. Ich hatte damals alles als HTML auf mehreren Seiten auf meiner Webseite. Im Juni 2017 habe ich jedoch meine Webseite umgebaut und dieser Teil ging dann leicht verloren. Deswegen gibt es die einzelnen Seiten nun hier als Kapitel in einem PDF.

1.2 Dieses Dokument und das Ziel

Dieses Dokument beschreibt einen vollständigen Aufbau eines MPLS VPN mit Cisco Geräten. Dazu wird GNS benutzt.

Alle Konfigurationen von allen Geräten sind im Download enthalten. Theoretisch kann man die Topologie zu einem bestimmten Kapitel laden und es sollte Funktionieren. Zumindest war das mal so.

1.3 Voraussetzungen

Sie sollten schon einmal ein Cisco Gerät in der Hand gehalten haben, weiterhin wäre es von Vorteil ein solches Gerät zumindest in den Grundzügen schon konfiguriert zu haben. Wenn Sie noch nie ein Cisco in der Hand hatten, aber dafür andere Hardware Router wie Juniper oder ALU, ist das OK denke ich.

Weiterhin sollten Sie die Grundzüge von MPLS kennen. Zumindest wissen was es tut. BGP und ISIS Kenntnisse wären auch ganz praktisch.

1.4 Verzeichnisse

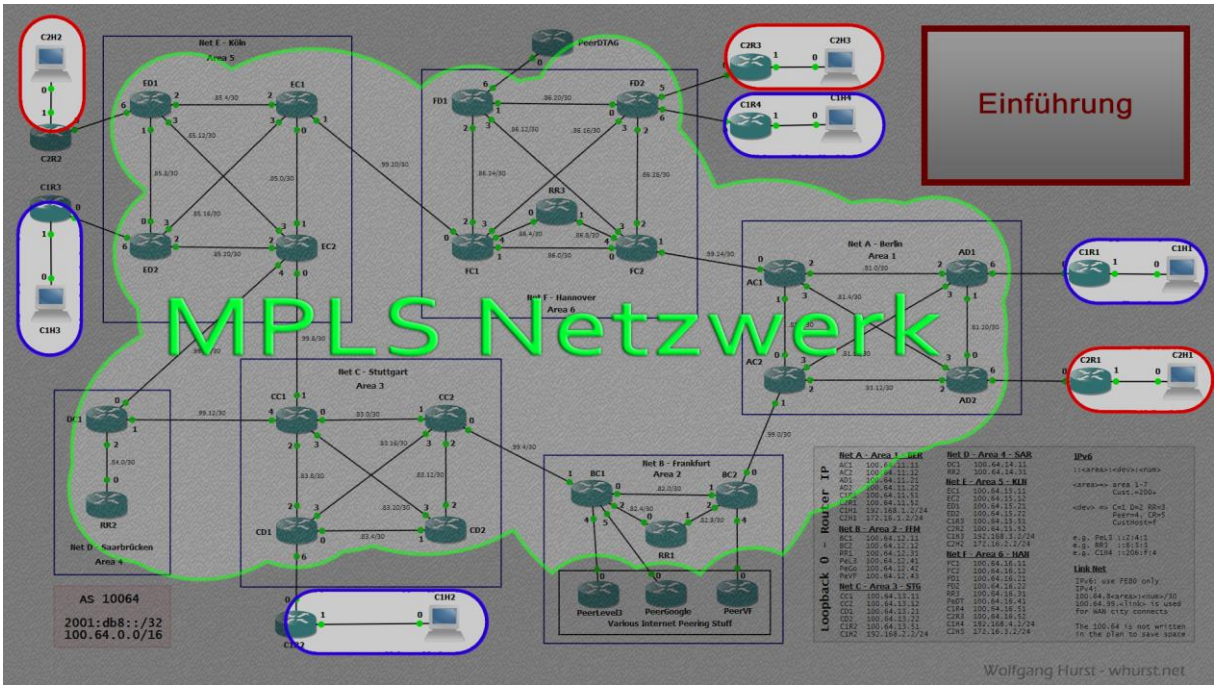
1.4.1 Inhaltsverzeichnis

1.4.2 Abkürzungen

ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
LSR	Label Switch Router
LER	Label Edge Router
MPLS	Multiprotocol Label Switching
OSPF	Open Shortest Path First
PE	Provider Edge
RD	Route Distinguisher
RIP	Routing Information Protocol
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding

2 Basics

2.1 Einführung

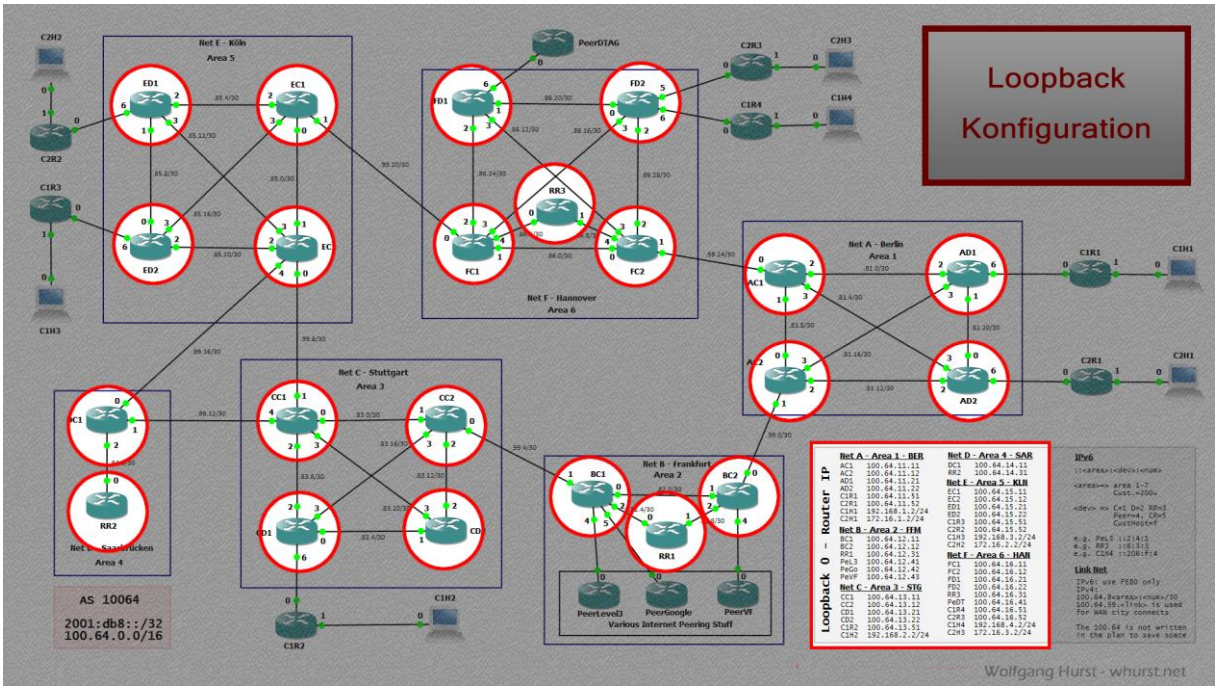


In diesem kleinen Tutorial möchte ich euch gerne zeigen, wie man aus einer Ansammlung von Routern ein Netzwerk aufbaut welches MPLS VPN fähig ist. Das Tutorial selbst ist in kleine Schritte aufgeteilt, so dass man immer folgen kann. Es steigt nicht "irgendwo" ein oder setzt bereits gewisse Konfigurationen voraus.

Wir werden die Router von Null an konfigurieren. Dazu zählt das Loopback, die Linknetze, das IS-IS Routing, das BGP und das MPLS mit dem VPN ... bis zum bitteren Ende ... wo wir tatsächlich einen Kunden mit mehreren Standorten durch das Netzwerk hauen können.

Das Grüne ist unser MPLS VPN und die Blauen bzw. Roten Kunden werden zusammengeschaltet in ein VPN, ohne das die Kunden merken das was Grünes dazwischen ist.

2.2 Loopback

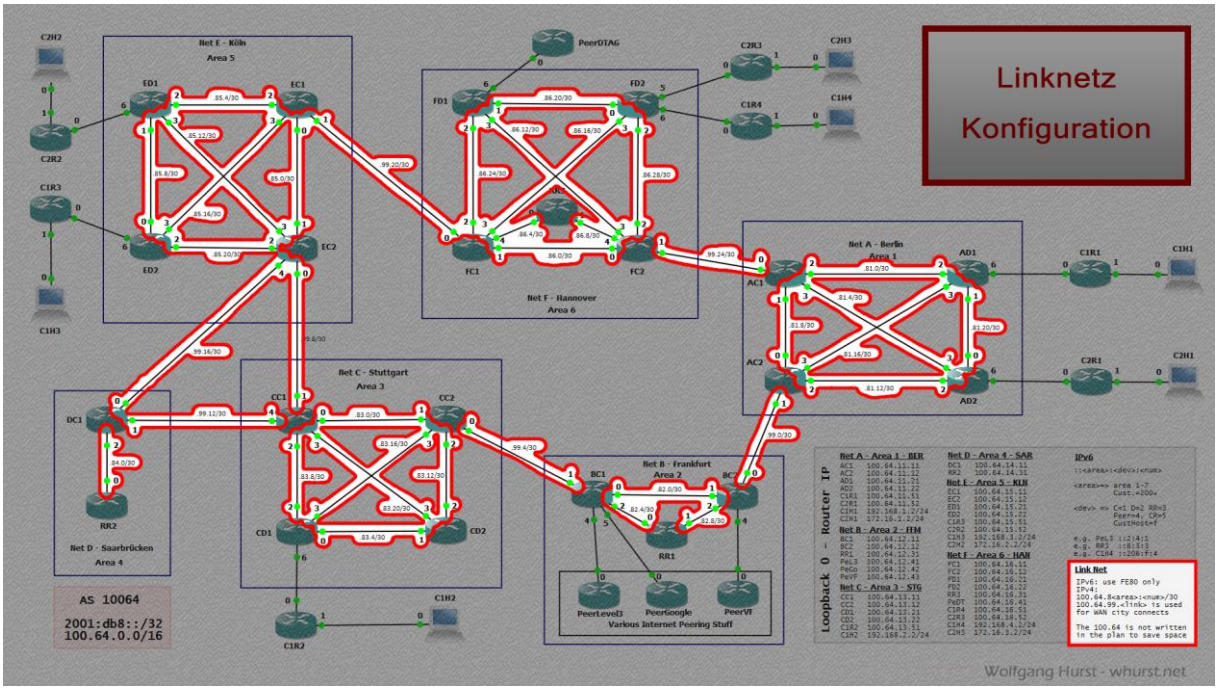


Wir konfigurieren als erstes die IP der Router. Jeder Router hat eine eindeutige IP im ganzen Netzwerk, die Loopback Null. Man kann aber auch eine andere nehmen. Die IP Adresse sollte sich im offiziellen Netzwerk befinden. Man kann auch private IP Adressen nehmen, dann müsste ich aber noch ein Management Netzwerk aufbauen, dazu habe ich aber gerade keine Lust.

Ich habe bereits im Netzdesign die Loopback Null IP Adressen niedergeschrieben. Die müssen nun auf die Router drauf. Die Loopback ist immer eine /32 bzw. eine /128. Da unser Netzwerk IPv4 als auch IPv6 können soll, werden auch beide konfiguriert.

```
Router#conf t
Router(config)#interface loopback 0
Router(config-if)#ip address <ipv4-address> 255.255.255.255
Router(config-if)#ipv6 enable
Router(config-if)#ipv6 address <ipv6-address>/128
```

2.3 Linknetz oder Koppelnetz



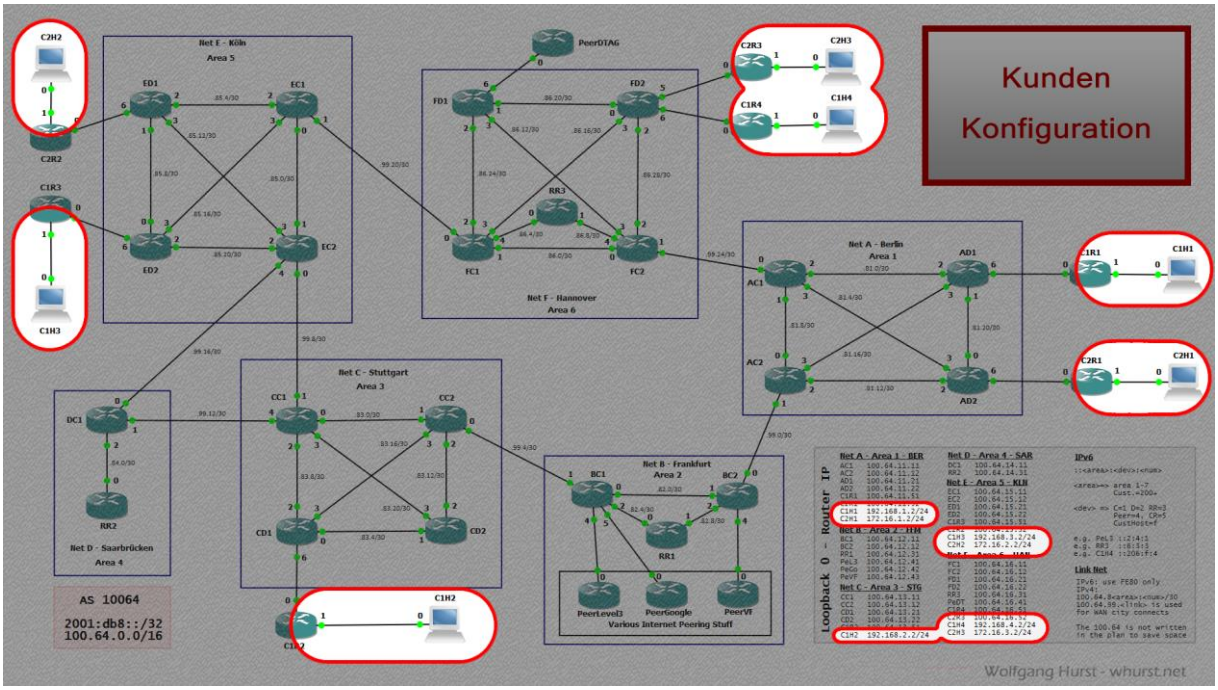
Wir brauchen für die Verbindungen untereinander leider die Konfiguration eines Linknetzes. Jedoch nur für IPv4. Man könnte durchaus jetzt hier private RFC 1918 IP Adressen nehmen, da die IP Adressen effektiv vollkommen unrelevant sind. Jedoch möchte ich darauf verzichten. Spätestens dann, wenn man ein Traceroute macht, würden diese privaten IP Adressen auftauchen.

Bei IPv6 werden wir darauf verzichten, dort arbeiten wir mit den Local-Link FE80 Adressen, diese werden automatisch ausgewürfelt. Da uns, sowohl bei IPv4 als auch bei IPv6, die Adresse der Gegenseite später nicht interessiert, zeige ich hier beide Wege. Einmal mit IPv4 und offiziellen IP Adressen und einmal mit IPv6 FE80 Adressen. Ihr werdet feststellen, dass es mit den IPv6 Adressen viel cooler wird :-)

Hier mal ein Konfigurationsschnipsel für den Core-Router in Area 1 - den AC1:

```
AC1#conf t
AC1(config)#interface GigabitEthernet 0/0
AC1(config-if)#ip address 100.64.99.25 255.255.255.252
AC1(config-if)#ipv6 enable
AC1(config-if)#no shut
AC1(config-if)#interface GigabitEthernet 1/0
AC1(config-if)#ip address 100.64.81.9 255.255.255.252
AC1(config-if)#ipv6 enable
AC1(config-if)#no shut
AC1(config-if)#interface GigabitEthernet 2/0
AC1(config-if)#ip address 100.64.81.1 255.255.255.252
AC1(config-if)#ipv6 enable
AC1(config-if)#no shut
AC1(config-if)#interface GigabitEthernet 3/0
AC1(config-if)#ip address 100.64.81.5 255.255.255.252
AC1(config-if)#ipv6 enable
AC1(config-if)#no shut
```

2.4 Kundenseite



Auch die Kunden Seite konfigurieren wir schon einmal. Die brauchen wir zwar eine ganze Weile nicht, aber hier ist schon einmal ein guter Zeitpunkt. Die Kunden Router stehen mit einem Bein in unserem Netz, dieses Bein werden wir später erst konfigurieren, wenn das MPLS VPN steht. Aber wir wollen schon einmal zwischen dem Kunden Host und dem Router eine Verbindung aufbauen.

Wir nehmen an, der Kunden Router ist auf Kunden Seite mit einem Switch verbunden wo ein /24 drauf liegt. Das habe ich jedoch nicht so im Plan eingezeichnet. Auch werde ich diesen Switch nicht auch noch in die Simulation packen, da er uns effektiv eigentlich nicht interessiert. Wir gehen einfach von der .1 auf der Kundenseite aus und die .2 als Host. Der Host ist hier auch ein C7200 den ich als Host per Icon getarnt hab.

Pro Kunde habe ich mal ein /16 mit privaten IPs benutzt der pro Standort in ein /24 zerfällt. Eine Default Route auf den Hosts zeigt immer Richtung Router. Eine ganz simple Konfiguration.

Hier mal eine Beispiel Konfiguration vom Kundenrouter und Kunden Host 1:

```
C1H1#conf t
C1H1(config)#inter gigabitEthernet 0/0
C1H1(config-if)#ip address 192.168.1.2 255.255.255.0
C1H1(config-if)#no shut
C1H1(config-if)#exit
C1H1(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.1
```

Und den Router:

```
C1R1#conf t
C1R1(config)#interface GigabitEthernet 1/0
C1R1(config-if)#ip address 192.168.1.1 255.255.255.0
C1R1(config-if)#no shut
```

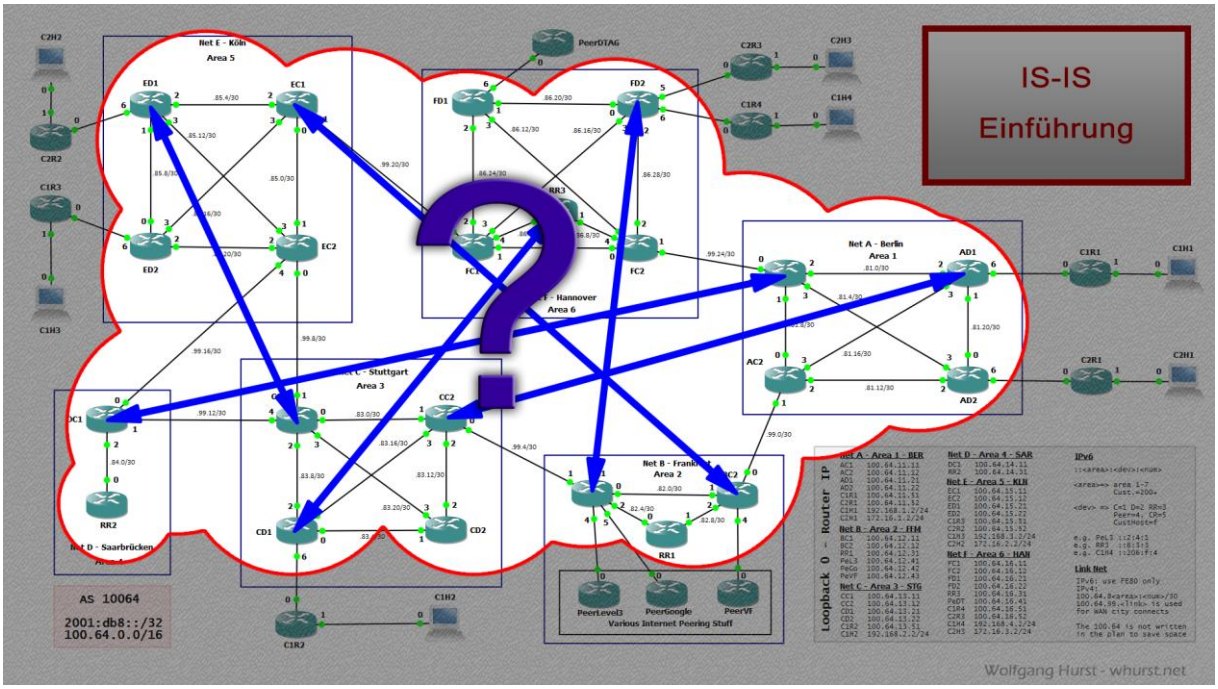
Jetzt können Host und Router zumindest sich untereinander Pingen. Die restlichen Standorte muss man dann noch konfigurieren. Der Zusammenschluss findet dann später statt.

Anmerkung

Leider kann das GNS3 Image, welches ich habe, kein IPv6 VPN. Ich habe daher das gesamte IPv6 beim Kunden nicht konfiguriert

3 IS-IS

3.1 Einführung



Wir brauchen nun ein Routing Protokoll mit dem wir die Systeme erreichen können. Es bringt uns ja nichts nur bis zum nächsten System zu kommen. Es gibt da nun verschiedene Ansätze:

Statische Routen? Ja nun, das ginge zwar, aber danach könnten wir keine Labels austauschen, und die brauchen wir ja. Weiterhin wäre statisches Routen auch nicht so der Hammer.

RIP? RIP ist RIP. RIP braucht man heutzutage nicht mehr wirklich. Man sollte es schon aufgrund der Security aus allen Routern werfen. Aber keiner tut das.

OSPF? Ja mit OSPF könnte man schon mehr anfangen. Ist aber für uns nicht geeignet.

BGP? BGP kann alles was wir brauchen, hat aber ein paar Nachteile. Zum einen ist es nicht besonders schnell, aber das größte Problem wäre die Vermatschung zwischen den Nodes. Der Router würde uns nach einer gewissen Zeit keine Lust mehr haben wollen. Sätze man ein Zentralen Route Reflektor ein, hätten wir das Problem das wir wieder Routen bräuchten um dahin zu kommen.

Bleibt einzig und allein: IS-IS. Was auch gleich mein Lieblingsprotokoll ist.

IS-IS braucht nicht wirklich eine großartige Konfiguration. Es basiert auf Netzbereiche und zwei Levels. Level 2 ist Core Routing und Level 1 das Routing innerhalb eines Netzwerkes. Nur Level 2 Router können Routen aus verschiedenen Netzwerken austauschen. Ein Level 1 Router kann das nicht. Weiterhin bekommt ein Level 1 Router nicht alle Routen aus allen Netzen, sondern bekommt eine Default Route von einem Level 2 Router zugesteckt. Das macht die Routing Tabelle extrem klein, egal wie viele Netzwerke ich zusammen schalte.

Alle Level 1 Router kennen sich untereinander, werden aber durch Level 2 Router quasi abgeschirmt und in "ein Topf" geworfen. Das macht das Routing Protokoll so nett, weil es das tut ohne das ich was tun muss. Schön ...

Wir brauchen aber eine sogenannte Netz-ID. Die lege ich nun wie folgt fest:

```
49.000<area hex>.0000.0000.<ip[3]><ip[4]>.00
```

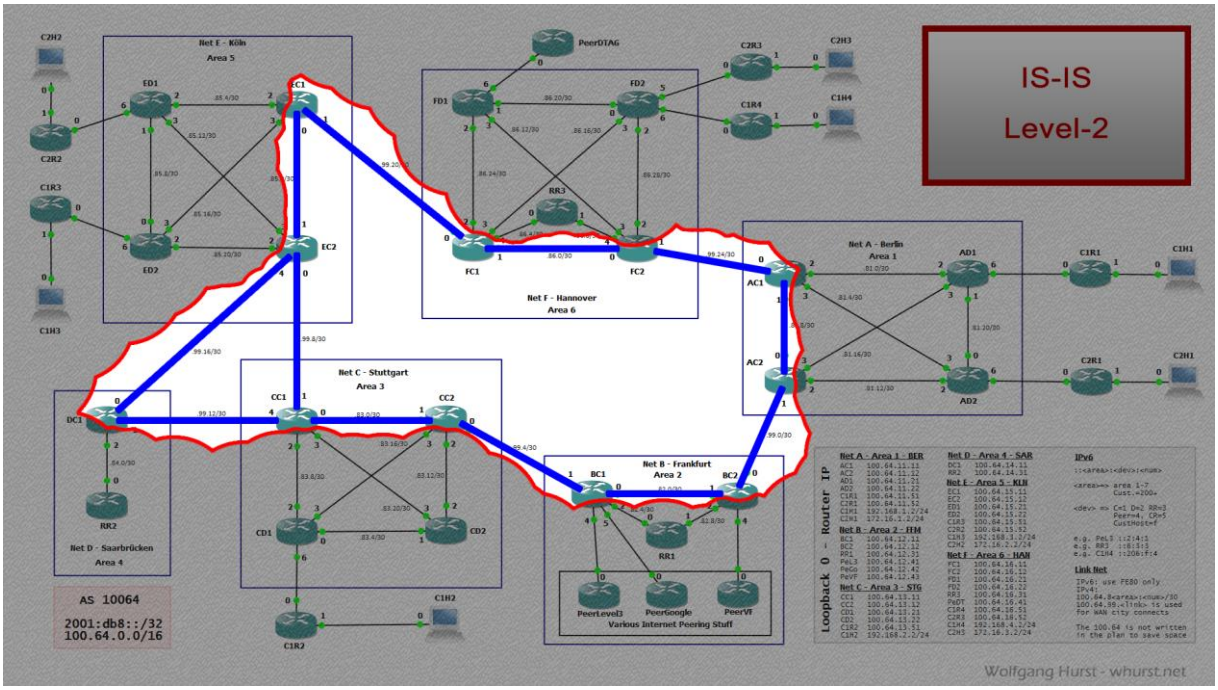
Die Netzwerk Area ist also für Saarbrücken zum Beispiel die `000d`

Die System ID für CC1 wäre `0000.0000.1311`

Jedes System hat eine einzigartige System ID, man kann auch die MAC nehmen, oder man nummeriert die Systeme einfach durch. Das ist vollkommen egal. Mir ist leider aufgefallen das die Version die ich benutze eine Warnung auswirft, wenn es zwei Systeme sieht mit der gleichen System-ID aber in unterschiedlichen Areas. Daher habe ich die IP genommen um dort auch ein Unterschied zu haben. Meldungen sind verschwunden - alles Optimal

Zuerst wollen wir die Core Router konfigurieren und danach die anderen Systeme. Es ist wichtig, dass man explizit den Level pro Link und System angibt, weil sonst kann es ganz fürchterliche Probleme geben, könnt es ja runterladen und dann daran rumspielen.

3.2 Level 2



Auf jedem Core Router sind die Nachbarn jeweils auf dem Interface 0 und 1 angebunden. Bis auf die beiden Ausnahmen EC2 und CC1, da hängt noch einer am Interface 4. An diesen Interfaces müssen wir explizit den IS-IS Typ festlegen, weil sonst funktioniert es nicht.

Alle Core Router sind weiterhin Level-2-1 Router, das ist zwar bei Cisco Default, aber wenn man es reinschreibt ist das besser, man sieht es nachher nicht, weil Default und so. Aber wer weiß was in 30 Jahren Default ist ...

Wir brauchen aber unbedingt unsere Loopback Null IP Adresse mit im IS-IS. Es gibt nun zwei Wege das zu tun:

1. Man konfiguriert das Loopback Null mit `ip router isis`, wie die anderen Interfaces auch. Das hat aber den Nachteil das der Router dann auf dem Loopback Interface ebenfalls anfängt IS-IS Hello Pakete zu senden, was ziemlich sinnlos ist. Aber er tut es. Es ist unbedingt Weg Nummer zwei zu wählen.

2. Man konfiguriert das Interface mit `passive-interface` im IS-IS Routing Context. Das hört sich absurd an, ist es auch! Es ist leider historisch so, dass ein passive-interface mit in das Routing genommen wird, aber keine IS-IS Hellos versendet. Merke: `passive-interface` schaltet NUR die Hellos aus, aber NICHT die Aufnahme des Interfaces in das Routing.

Das ist leider etwas verwirrend, weil ja eigentlich "passive-interface" bedeutet, dass man ein Interface vollkommen ignorieren will. Aber hier ist das halt anders. Weiterhin funktioniert das nur, wenn man vorher bereits Interfaces via `ip router isis` hinzugefügt hat!

Nun gut ... kommen wir zur Konfiguration eines, bzw. aller Core Router:

```
CoreRouter#conf t
CoreRouter(config)#router isis
CoreRouter(config-router)#is-type level-1-2
CoreRouter(config-router)#net 49.000<area>.0000.0000.<sys id>.00
CoreRouter(config-router)#exit
CoreRouter(config)#interface GigabitEthernet 0/0
```

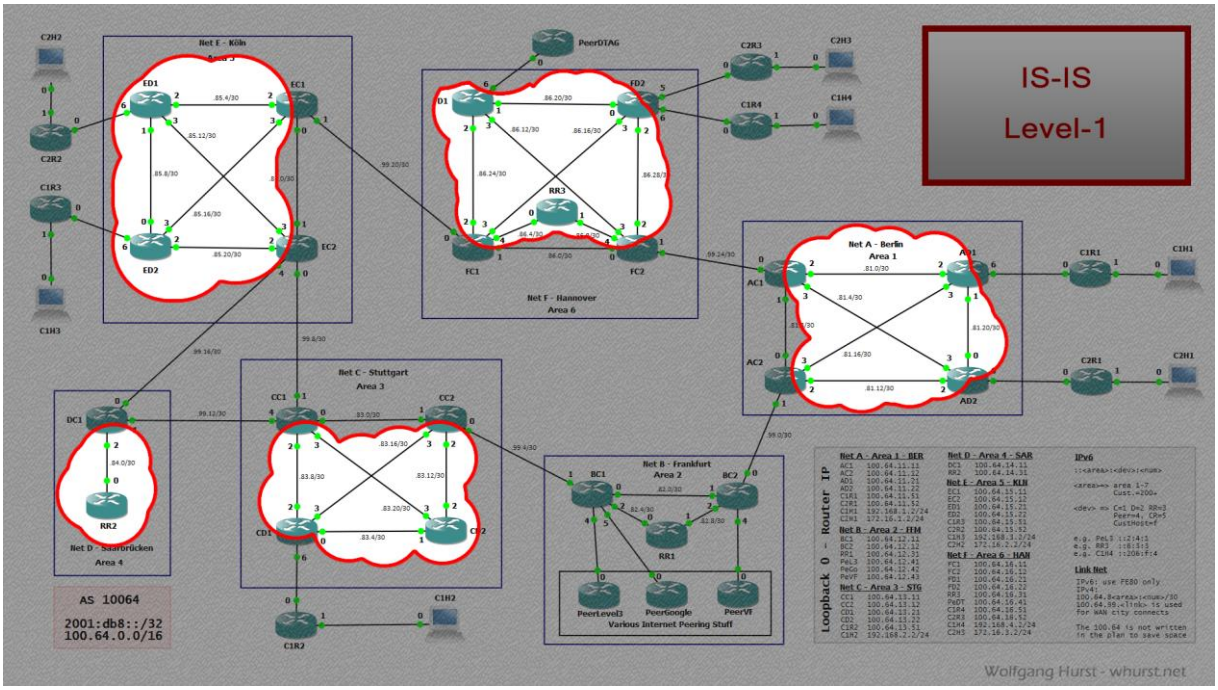
```
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-2-only
CoreRouter(config-if)#interface GigabitEthernet 1/0
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-2-only
CoreRouter(config-if)#exit
CoreRouter(config)#router isis
CoreRouter(config-router)#passive-interface loopback 0
```

`ip router isis` Aktiviert das IS-IS Protokoll für IPv4 auf diesem Interface

`ipv6 router isis` Aktiviert das IS-IS Protokoll für IPv6 auf diesem Interface

`isis circuit-type level-2-only` Kennzeichnet das Interface als Layer-2 Link

3.3 Level 1



Das Level 1 Routing geschieht nun in zwei Teilen. Auf dem Core Router muss das Interface als Level 1 konfiguriert sein und zum anderen müssen die Systeme konfiguriert werden. Wir wollen zuerst alle Systeme innerhalb des Level 1 Netzwerkes konfigurieren. Dann kann man die "noch" kleinen Routing Tabellen sich anschauen.

Um ein System zum Level 1 Router zumachen sind folgende Befehle nötig, die Kundenrouter gehören nicht dazu:

```
L1#conf t
L1(config)#router isis
L1(config-router)#net 49.000<area>.0000.0000.<sys id>.00
L1(config-router)#is-type level-1
L1(config-router)#exit
L1(config)#interface Gigabit 0/0
L1(config-if)#ip router isis
L1(config-if)#ipv6 router isis
L1(config-if)#interface Gigabit 1/0
L1(config-if)#ip router isis
L1(config-if)#ipv6 router isis
L1(config-if)#interface Gigabit 2/0
L1(config-if)#ip router isis
L1(config-if)#ipv6 router isis
L1(config-if)#interface Gigabit 3/0
L1(config-if)#ip router isis
L1(config-if)#ipv6 router isis
L1(config-if)#exit
L1(config)#router isis
L1(config-router)#passive-interface loopback 0
```

Nicht alle haben die ganzen Interfaces, der RR2 z.B. hat nur das eine und RR1 und RR3 haben nur das Interface 0 und 1.

Super wichtig ist aber `is-type level-1` im `router isis` Context. Wenn man das vergisst hat man später ein echtes Problem.

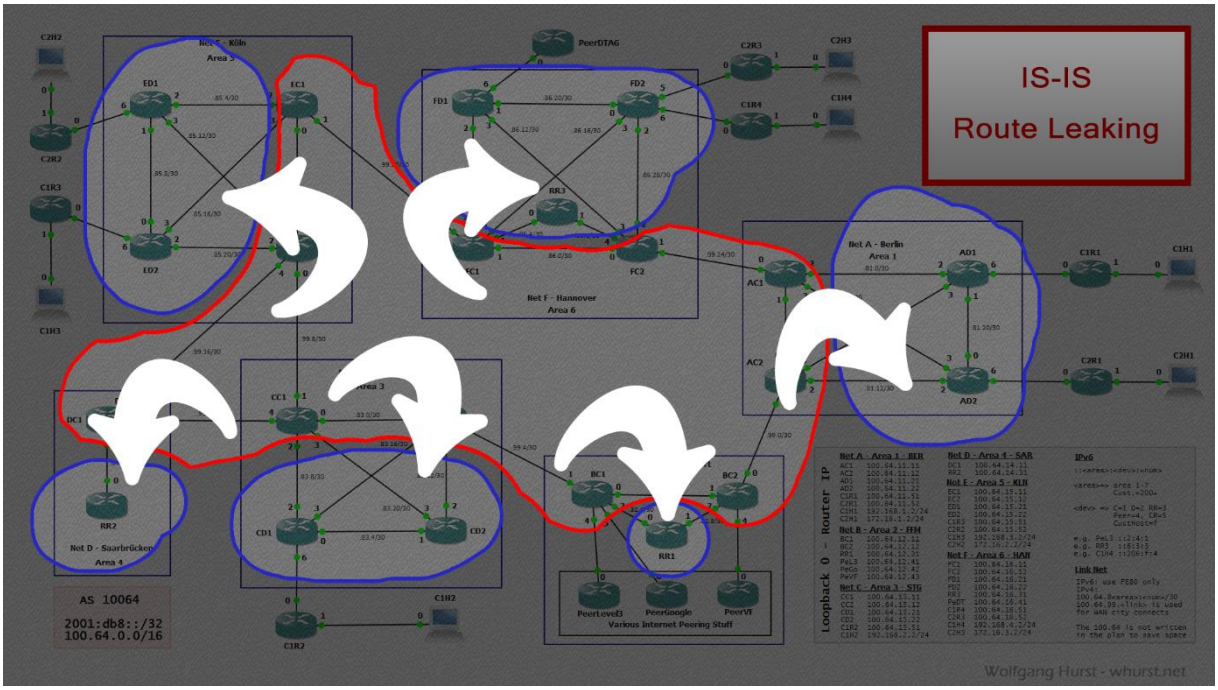
Man kann in den Router aber bereits jetzt schon ein paar Routen sehen. Was jetzt noch fehlt sind die Core Router und dessen Interfaces:

```
CoreRouter#conf t
CoreRouter(config)#interface Gigabit 2/0
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-1
CoreRouter(config-if)#interface Gigabit 3/0
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-1
CoreRouter(config-if)#interface Gigabit 4/0
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-1
CoreRouter(config-if)#interface Gigabit 5/0
CoreRouter(config-if)#ip router isis
CoreRouter(config-if)#ipv6 router isis
CoreRouter(config-if)#isis circuit-type level-1
```

Super wichtig ist hier der Befehl `isis circuit-type level-1`, damit der Router weiß das es Level 1 ist.

Nach einer Schrecksekunde kann man nun fröhlich Kreuz und Quer im Netz rum Pingen und Tracen. Achtet mal beim Traceroute auf den Output zwischen IPv4 und IPv6 ... herrlich ...

3.4 Route-Leaking



In unserem Level 1 Netzwerk sehen wir zurzeit nur unsere beiden Core Router und bekommen eine Default Route gesendet. Das würde theoretisch ausreichen um erfolgreich Routen zu können. Es ist aber nicht genug um BGP und MPLS fahren zu können. Jeder Router im gesamten Netzwerk muss alle anderen Router in der Routingtabelle haben.

Genau das erreichen wir nun durch einen Bruch im IS-IS, indem wir zusätzlich der Default Route noch alle anderen Routen in das Level 1 reinwerfen.

Auf den Core-Routern benötigen wir nun noch eine Redistribution:

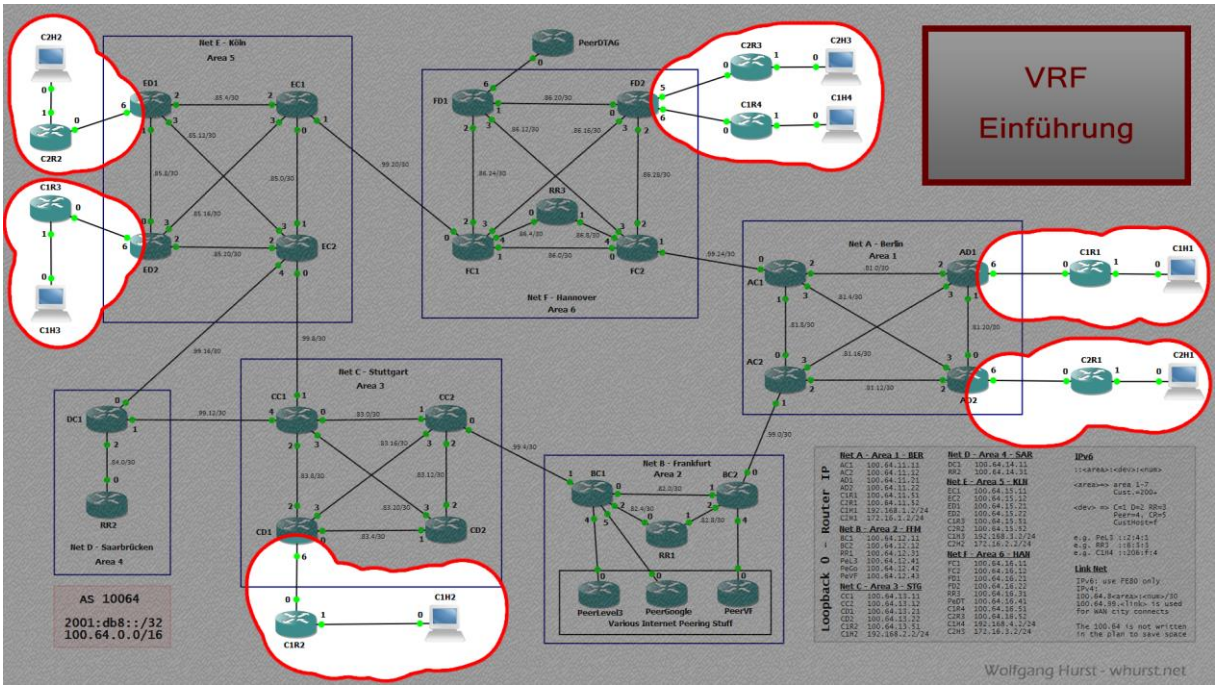
```
CoreRouter#conf t
CoreRouter(config)#access-list 101 permit ip any any
CoreRouter(config)#router isis
CoreRouter(config-router)#redistribute isis ip level-2 into level-1 distribute-list 101
CoreRouter(config-router)#address-family ipv6
CoreRouter(config-router-af)#redistribute isis ip level-2 into level-1 distribute-list 101
```

Die Access Liste ist zugegeben "etwas" großzügig ... Im richtigen echten Netzwerken macht man so etwas natürlich nicht. Genauso wenig würde man nicht ohne Identifikation arbeiten ... aber wir sind ja hier in einer Simulation und in einem Tutorial, da darf man das :-)

Damit füllt sich jetzt auch die Routing Tabelle aller Level 1 Router mit allen Loopback IPs. Das ist zwar etwas unübersichtlich für uns, aber die Maschine weiß schon was sie tut

4 VRF

4.1 Einführung



Ein VRF ist ein Virtueller Router im Router. Wir benötigen für jeden Kunden ein VRF. Der Name des VRF kann variieren, man sollte das aber nicht tun, weil man sonst sehr schnell durcheinanderkommt.

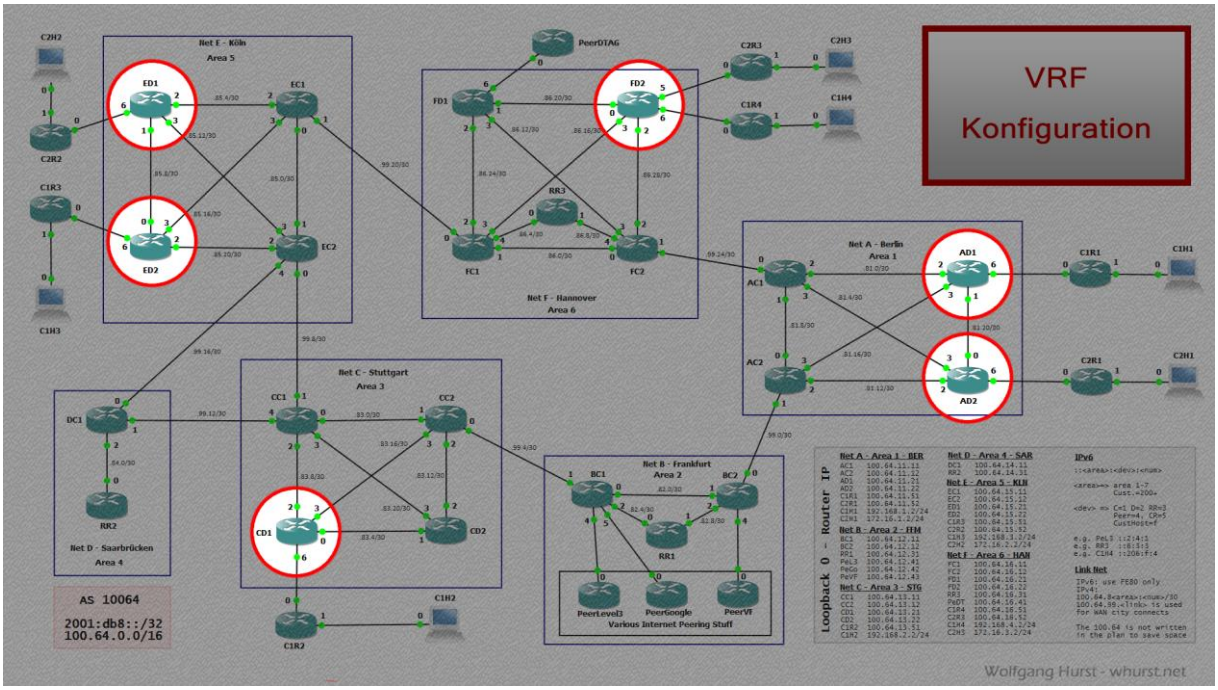
Das Interface, wo dieser Kunde angebunden ist, wird dann in die virtuelle Routerinstanz geschoben. Nachdem man das Interface dann in das VRF geschoben hat, kann man die IP auf dem Interface auch nicht mehr aus dem Globalen Routing erreichen. Wäre ja auch doof, wenn.

Auf dem Kunden Router allerdings bekommt man das nicht mit. Man kann auf dem Linknetz ganz normal Pingen und so, ob das Interface im VRF ist oder nicht, merkt man auf der Kundenseite überhaupt nicht. Ja ok - stimmt nicht ganz ... wenn man das Interface in das VRF wirft, kommt man von der Kundenseite nicht mehr in das Globale Netz und kann nichts mehr Pingen. Aber das wollen wir ja, sonst wäre es ja kein VPN

Das Linknetz zwischen unserem PE und dem Kundenrouter, ist sowohl für den Kunden, als auch für uns vollkommen unrelevant. Wir könnten in jedem Standort das gleiche Netz verwenden. Auch alle Kunden können das Netz verwenden. Es spielt keine Rolle. Das Netz ist zum einen nur Lokal im VRF für das Routing in das Kundennetzwerk wichtig, zum anderen taucht das Netzwerk auch nicht im VPN oder im Globalen Routing auf. Von daher können wir hier immer das gleiche Netz verteilen. Ich habe das 10'er Netz genommen, das war gerade noch frei.

Ich verwende aber pro Standort ein anderes 10'er Netz. man kann so viel besser Debuggen und auch zwischen den Routern mal ein Ping absetzen. Es ist auch besser für das Debugging hier in der Simulation.

4.2 Konfiguration



Wir benötigen die Konfiguration eines VRF auf allen Routern, wo der Kunde eine Anbindung hat. Man sollte den VRF Namen vereinheitlichen, der muss zwar nicht gleich sein, aber man sollte überall den gleichen Namen verwenden.

Die Konfiguration des VRF ist relativ einfach, ich habe jetzt mal eine Description hinzugefügt, damit überhaupt was drinsteht:

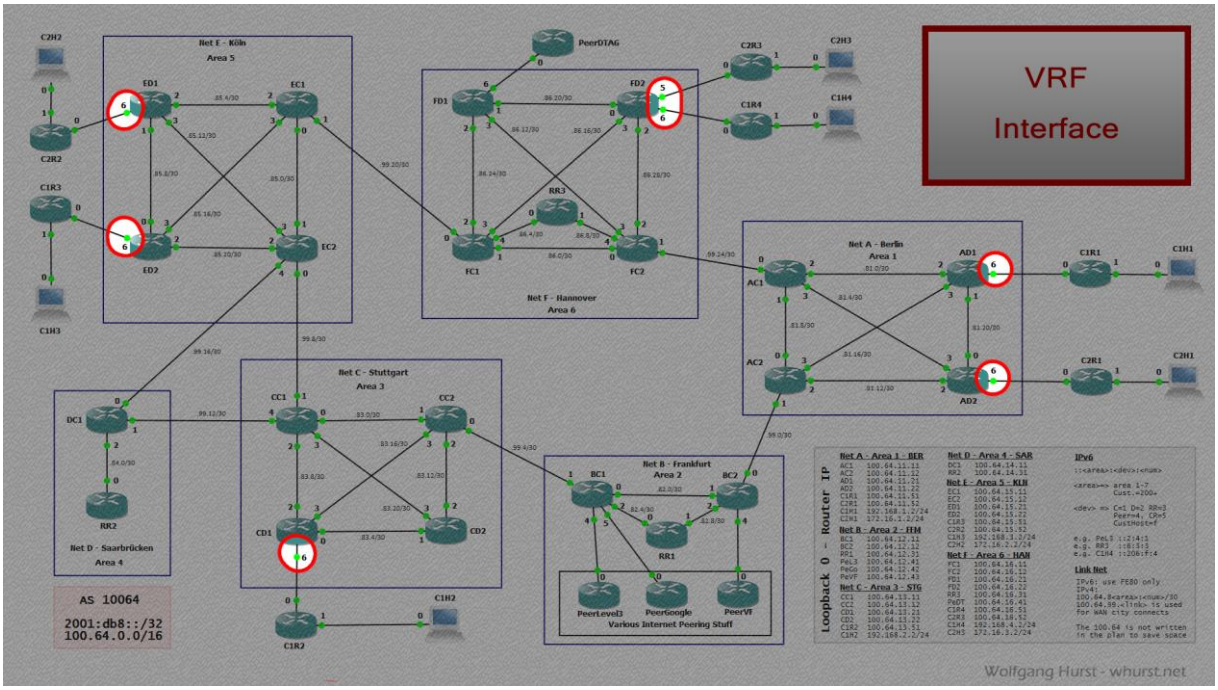
```
AD1#conf t
AD1(config)#ip vrf vpn01
AD1(config-vrf)#description VPN Customer 01
```

Auf dem FD2 Router, brauchen wir zwei VRFs, weil dort hängen zwei Kunden drauf:

```
FD2#conf t
FD2(config)#ip vrf vpn01
FD2(config-vrf)#description VPN Customer 01
FD2(config-vrf)#exit
FD2(config)#ip vrf vpn02
FD2(config-vrf)#description VPN Customer 02
```

Und das war es auch erst einmal schon

4.3 Interface



Nun geht man auf das Interface wo der Kunden angebunden ist. Man wirft das Interface in den Virtuellen Router und konfiguriert die Linknetz IP. Wie schon angedeutet verwende ich überall das 10'er.

Sollte der Kunde mit zwei Leitungen am gleichen Router hängen, dann muss sich logischerweise auch das Linknetz unterscheiden. Diesen Fall deckt mein Netzplan jetzt nicht ab.

Hier ein Beispiel vom AD1:

```
AD1#conf t
AD1(config)#interface GigabitEthernet 6/0
AD1(config-if)#ip vrf forwarding vpn01
AD1(config-if)#ip address 10.1.1.1 255.255.255.0
AD1(config-if)#no shut
```

Jetzt geht man auf den Kunden Router und richtet dort das Interface ganz normal ein, so wir ein ganz normales Interface:

```
C1R1#conf t
C1R1(config)#interface gigabitEthernet 0/0
C1R1(config-if)#ip address 10.1.1.2 255.255.255.0
C1R1(config-if)#no sh
```

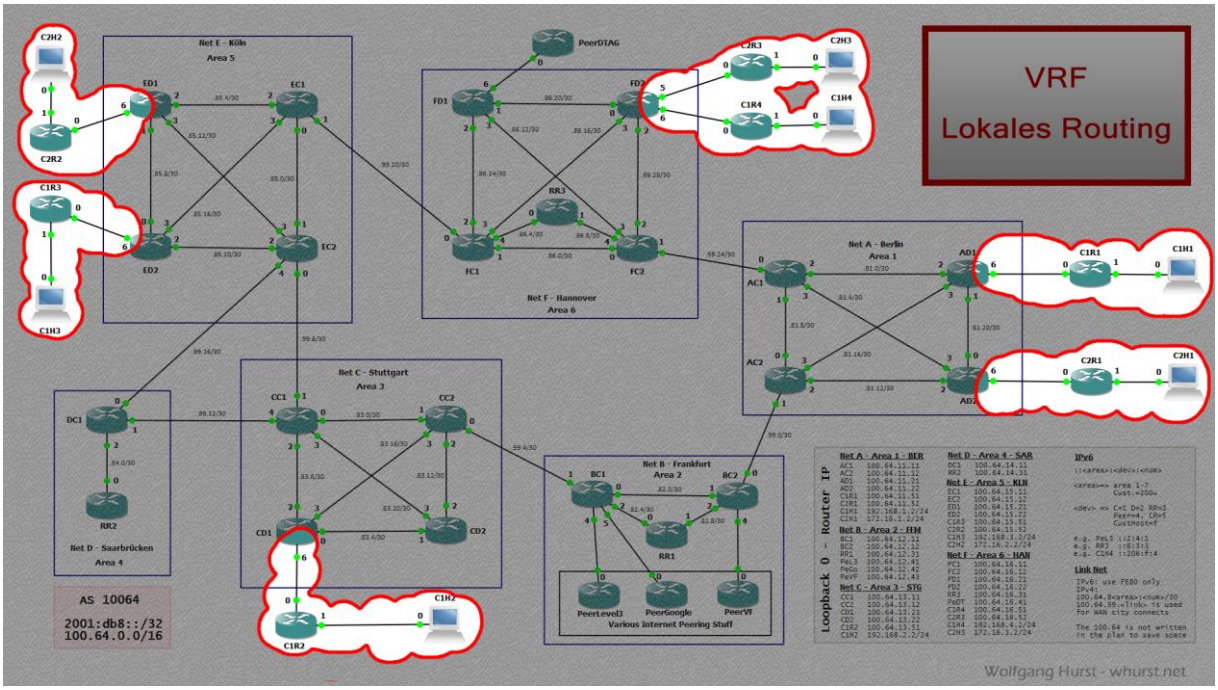
Danach kann man vom Kunden Router aus unseren PE Pingen. Umgekehrt geht das nur, wenn man beim Ping das VRF angibt:

```
AD1#ping vrf vpn01 10.1.1.2
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/85/120 ms
```

Anmerkung

Das (also mein) GNS Image kann leider kein IPv6 VPN. Von daher fehlt IPv6 beim Kunden gänzlich.

4.4 Lokales Routing



Wir müssen im VRF und auch auf dem Kundenrouter entsprechendes Routing konfigurieren. Der Kunde selbst verwendet keinerlei dynamische Routing Protokolle und wir tun das, was wir bereits auf dem Host gemacht haben.

Auf dem Kundenrouter konfigurieren wir eine Default Route zum PE.

```
C1R2#conf t
C1R2(config)#ip route 0.0.0.0 0.0.0.0 10.1.1.1
```

Auf dem PE konfigurieren wir innerhalb des VRF eine Route in das Kundennetzwerk:

```
AD1#conf t
AD1(config)#ip route vrf vpn01 192.168.1.0 255.255.255.0 10.1.1.2
```

Nachdem man das Lokale Routing eingerichtet hat, kann z.B. der Host das PE Pingen.

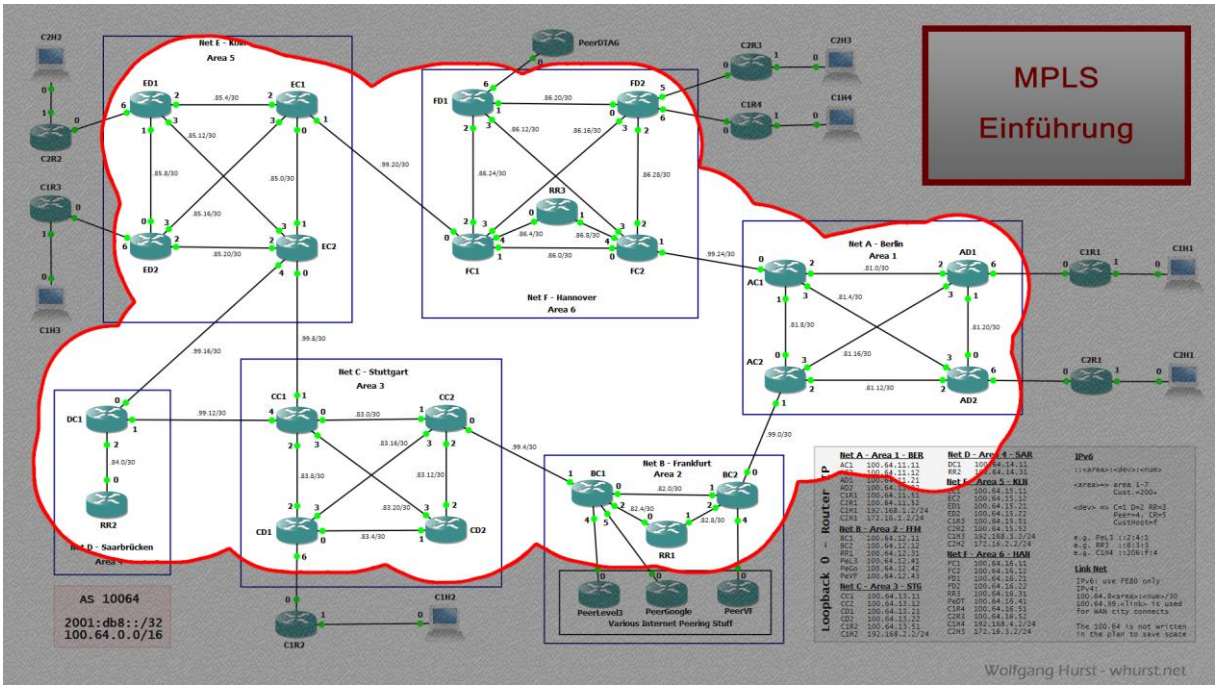
```
C1H1#ping 10.1.1.1
Sending 5, 100-byte ICMP Echos to 10.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 112/166/228 ms
```

Anmerkung

Leider ist das Image das ich verwende noch nicht in der Lage statische IPv6 Routen innerhalb eines VRF zu setzen. Das ist schade, weil wir und so die IPv6 Konnektivität zum Kunden abscheiden. Ich schau mal ob ich nicht doch noch ein Image auftreiben kann, was das kann ... Irgendwie ist auch das Image noch nicht so 100% IPv6 Ready ... na mal schauen. IPv6 habe ich aus diesem Grunde nicht konfiguriert, auch nicht im Kunden Netz

5 MPLS

5.1 Einführung



Jetzt kommen gleich zwei Eigenschaften zeitgleich. Beide funktionieren nur gegenseitig, bzw. in Abhängigkeit zueinander. Nämlich MPLS und BGP.

Ich will erst einmal MPLS machen, weil das geht im Core auch erst einmal ohne BGP.

MPLS ist eine Kapselung von Paketen. Ähnlich wie man es von XoverY Protokollen her kennt. MPLS umschießt das eigentliche IP Paket und hängt ein Label vor das Paket. Dieses Label definiert den Path durch ein MPLS Netzwerk.

Normalerweise muss jeder Router bei jedem Paket in die Routingtabelle schauen, damit er weiß wem er das Paket geben muss. Bei MPLS tut er das nicht mehr, sondern verwendet ein Label.

Damit das aber überhaupt funktioniert, muss der Router erst einmal wissen wer hier überhaupt welche Netzwerke und welche Labels anbietet. Dazu wird LDP verwendet. Mit LDP werden Labels auf einem Link ausgetauscht.

Der Router selbst baut intern eine Tabelle auf mit Label und Netzwerk. Er gibt seine Tabelle an seinen Nachbarn weiter. Der hat jetzt auch eine Tabelle, aber vermutlich die gleichen Labels. Das ist aber kein Problem. Es gibt zwei Tabellen. Eine Inputtabelle und eine Output Tabelle. Der Router bekommt ein Paket mit dem Label 17. Er schaut in seine Inputtabelle und sieht Label 17 bedeutet ich muss es an das Interface 2 weiterleiten mit dem Label 12. Er ändert also das Label und sendet es weiter. Daher auch der Name: Label Switching.

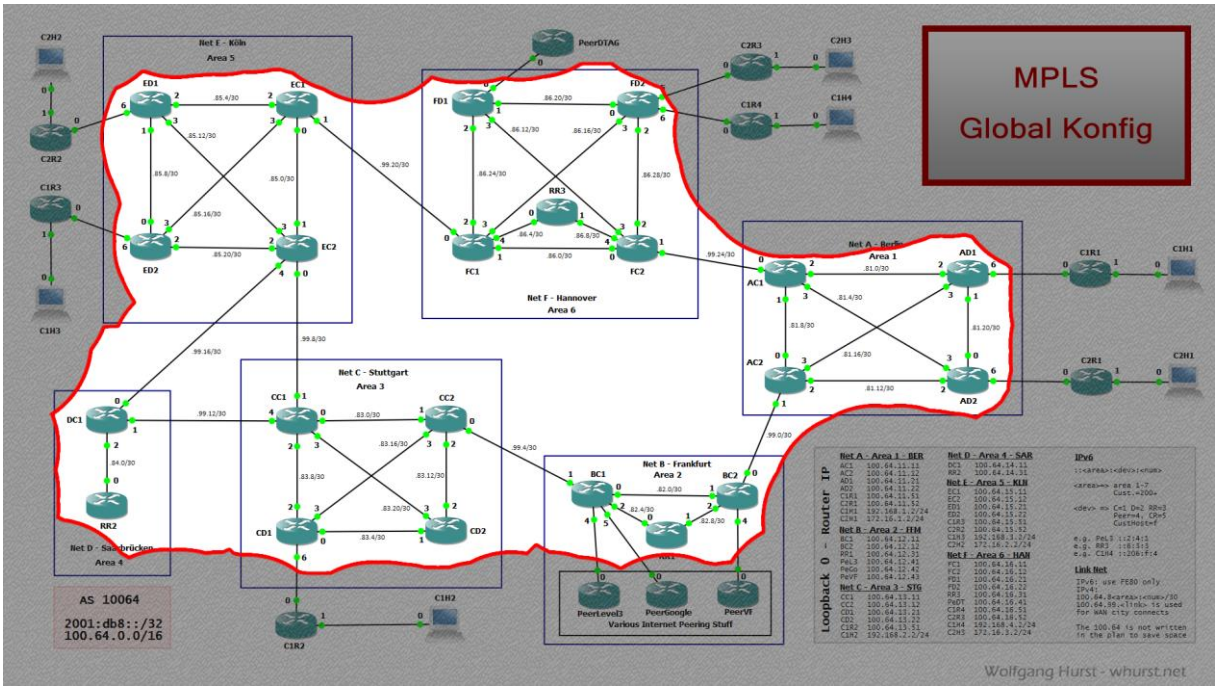
Damit aber ein solches Paket überhaupt erst einmal ein Label bekommen kann, benötigt man ein Ingress-Router (LER). Der Ingress Router nimmt ein IP Paket entgegen und sucht die Ziel IP in seiner Routingtabelle. Danach sucht er ein passendes Label und sendet es seinem Nachbarn mit dem Label.

Danach wird das Paket durch das MPLS durchgehauen bis es an der letzten Station ankommt. Der sogenannte Egress Router (LER). Der entfernt einfach das Label und stellt das Paket zu.

Das Schöne an dieser Methode ist, das kein Router im MPLS Netzwerk eigentlich weiß, was in dem Paket drin ist. Und keiner weiß an wen es eigentlich geht. Das ist aber auch den Routern im Netz vollkommen egal. Durch diese Technik allerdings kann man einfach alles durch ein Netzwerk hauen, sofern der Ingress und der Egress Router wissen was sie tun. Man kann ATM Pakete oder PSTN durch das MPLS hauen, wenn man will. Oder eigenentwickelte Protokolle, ohne dass einer der Router dazwischen (LSR) das mitbekommt. Der sieht nur das Label, mehr kennt er nicht.

Für unser VPN wollen wir auf das MPLS aufsetzen, weil wir genau das tun werden was ich gerade beschrieben hab. Früher zu Zeiten der Trommeln hat man noch mit seltsamen verschlüsselten Tunneln sich ein abgebrochen. Das ist nun vorbei.

5.2 Globale Konfiguration

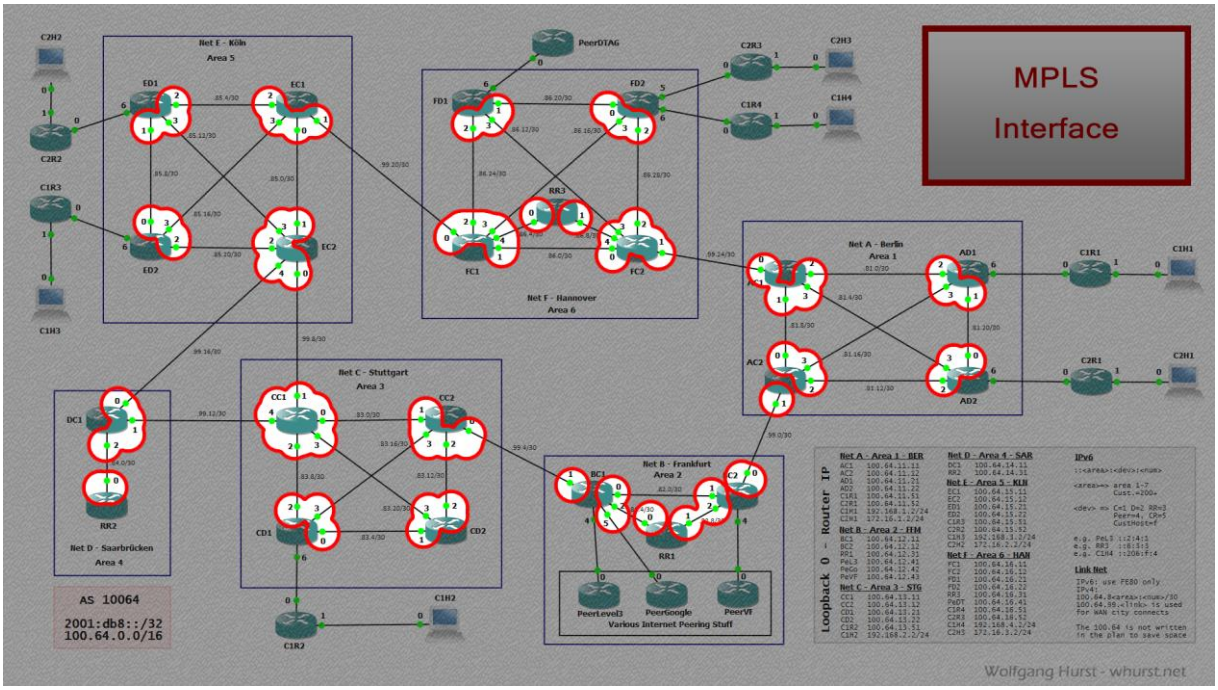


Diesen Schritt können wir fast überspringen. Die folgenden Befehle sind bei Cisco bereits per Default aktiviert. Aber zur Sicherheit:

```
Router#conf t
Router(config)#mpls ip
Router(config)#mpls ipv6 source-interface loopback 0
Router(config)#mpls label protocol ldp
```

Wobei ich nicht sicher bin, ob es für IPv6 auch eingeschaltet ist.

5.3 Interface Konfiguration



Wenn man nun auf ein Interface geht und dort mit `mpls ip` das Interface konfiguriert, fängt der Router sofort an dort LDP Pakete zu senden und zu empfangen.

Nimmt man mal als Beispiel AC1 und AC2 und aktiviert dort das MPLS, sieht man wenig später in der MPLS Forwarding Tabelle das der AC1 das Linknetz 99.0 zum Beispiel erreicht in dem er ein Label hinzufügt für AC2. Das Netz würde er dann mit einem Label erreichen können. Man sieht aber auch, dass er für den AD2 zum Beispiel schon ein Label reserviert hat. Das Label 17 in diesem Fall. Das bedeutet, wenn nun einer von außen kommt der braucht nur zu sagen: "Hier das ist für die 17" ...

```
AC1#sh mpls forwarding-table
Local  Outgoing    Prefix            Bytes tag  Outgoing    Next Hop
tag    tag or VC   or Tunnel Id      switched   interface
16     Untagged    100.64.11.12/32   0          Gi2/0       100.64.81.2
17     16          100.64.12.12/32   0          Gi1/0       100.64.81.10
18     18          100.64.12.31/32   0          Gi1/0       100.64.81.10
19     Untagged    100.64.11.21/32   0          Gi2/0       100.64.81.2
20     Pop tag     100.64.99.0/30    0          Gi1/0       100.64.81.10
21     Untagged    100.64.81.20/30   0          Gi2/0       100.64.81.2
22     Untagged    100.64.81.16/30   0          Gi2/0       100.64.81.2
23     23          100.64.82.4/30    0          Gi1/0       100.64.81.10
24     25          100.64.82.0/30    0          Gi1/0       100.64.81.10
25     Untagged    100.64.81.12/30   0          Gi2/0       100.64.81.2
26     26          100.64.82.8/30    0          Gi1/0       100.64.81.10
```

Mit zwei Routern macht das aber kein Spaß, wir werden nun alle Router, bis auf die Kunden Router, mit MPLS versorgen:

```
Router#conf t
Router#mpls ip
Router(config)#mpls ipv6 source-interface loopback 0
Router(config)#mpls label protocol ldp
Router(config)#interface GigabitEthernet 0/0
Router(config-if)#mpls ip
Router(config-if)#interface GigabitEthernet 1/0
Router(config-if)#mpls ip
Router(config-if)#interface GigabitEthernet 2/0
```

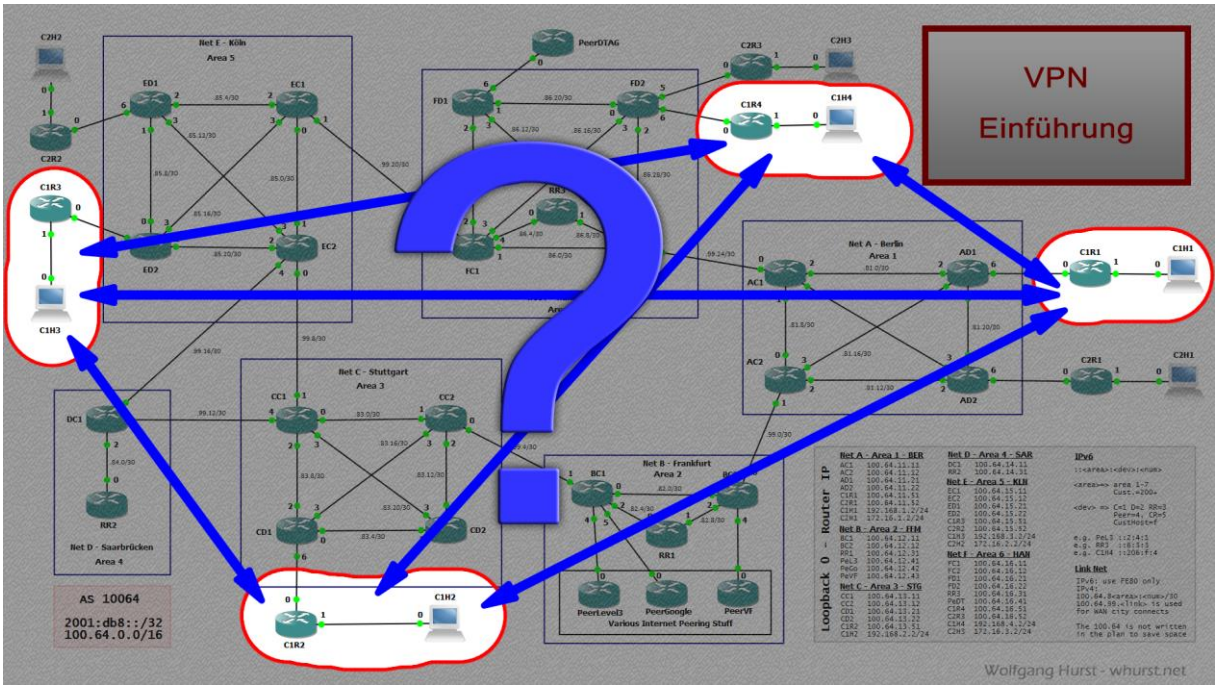
```
Router(config-if)#mpls ip
Router(config-if)#interface GigabitEthernet 3/0
Router(config-if)#mpls ip
Router(config-if)#interface GigabitEthernet 4/0
Router(config-if)#mpls ip
```

Bitte darauf achten, dass die Interfaces zu den Peers und Kunden Routern nicht mit MPLS versorgt werden.

Wenn man das überall ausrollt, bekommt man eine schöne Label Liste. Wenn man nun mal ein Traceroute macht oder ein PING snifft, wird man sehen das er jetzt bereits schon überall Labels benutzt. Das bedeutet unsere Core Kommunikation läuft bereits jetzt schon im Label Switch verfahren.

6 VPN

6.1 Einführung



Jetzt wird es interessant. Nun verbinden wir unsere noch lokalen VRF über das MPLS zu einem echten VPN.

Damit das aber funktioniert sind ein paar Informationen hilfreich. Damit ein Standort den anderen erreichen kann, hätte man auch ein Tunnel GRE bauen können. Man würde den anderen Standort als Ziel eintragen. GRE würde dann noch die ganzen Pakete umhüllen und durch das Netzwerk schieben. Ja so war es früher.

Wir wollen ja unser MPLS Netzwerk benutzen. Die Idee ist nun im MPLS ein MPLS zufahren. Das bedeutet das ein Paket bereits gelabelt ist, der Ingress Router dieses Paket noch einmal Labelt. In dem Backbone hat man dann also zwei Labels. Eins für den Backbone und eins für das VPN. Wie ich bereits geschrieben hab, ist es dem Core sowieso egal was er weiterschiebt.

Das bedeutet das Netzwerk muss bereits MPLS können, gut das kann es bereits.

Das bedeutet das der Ingress Router ein VPN Paket für andere Standorte entsprechend labeln muss, und zwar mit einem VPN Label. Genau ... Und der Ingress Router in diesem Fall ist unser VRF. Die Hardware ist quasi ein Double-Ingress-Router. Einmal wenn ein Paket am VRF ankommt, und dann noch einmal wenn es auf dem VRF in das Globale Netzwerk geht.

Im globalen Routing kennt durch IS-IS jeder jeden. Das MPLS im globalen Routing hängt also am IS-IS. Im MPLS ist es zwingend erforderlich das die PE Router jeden kennen.

Aber in unserem VPN ist das zurzeit nicht so ... Hier kennt sich nur das lokale Netzwerk. Standort A kennt Standort B nicht. Für MPLS ist das aber wichtig, weil es die Labels zu allen Zielen aufbauen muss. Zwischen den VRF auf entfernten Routern kann man halt kein LDP (Label Distribution Protokoll) aufbauen. Aber dummerweise genau das müssen wir tun, Router A muss die Labels von Router B

kennen, nur wie soll er die bekommen, da beide ja vom globalen Routing abgeschottet sind? Und genau da kommt BGP ins Spiel.

Mit BGP können wir Labels verwalten und tauschen, und BGP braucht keine physikalisch angeschlossene Nachbarn (wie IS-IS/LDP) damit es funktioniert. BGP ist perfekt. Die Überlegung ist nun, wenn jeder Router seine VPN Routen in eine zentrale BGP Instanz wirft, in allen Standorten, dann kennt am Schluss jeder jeden. Und wenn jeder jeden kennt, hat man so eine Verteilung der Labels zustande gebracht. Ist das gleiche wie LDP, nur langsamer.

Und genau das werden wir tun. Wir werden im Netzwerk ein sogenannten globalen Route Reflektor aufbauen. Das ist nichts anderes als ein BGP Prozess der Routen sammelt und sie allen zur Verfügung stellt. Dann werden wir auf allen PE Routen alle Routen aus einem VRF an diesen Route-Reflektor weiterreichen. In der Konsequenz das dann alle VPN beteiligten Router automatisch über alle Routen im Netzwerk informiert wird.

Es ist zwar jetzt nicht so super einfach, aber im Prinzip ist es genauso.

Lasst uns erst einmal ein Route Reflektor bauen und die Verbindung herstellen, dann wird vieles klarer.

6.2 BGP Peer Groups

Wir bräuchten keine Peering Groups. Aber ich möchte gerne daraufhin deuten, dass man grundsätzlich Peering Groups erstellen sollte. Sie machen einem nicht nur die Arbeit leichter, nein, sie sparen tatsächlich auch Router Ressourcen. Gerade in einem Netzwerk mit mehreren Nachbarn die alle das Gleiche tun, sollte man unbedingt zwingend Peering Groups erstellen.

Hier ist eine Liste der Gruppen die ich verwenden werde:

myClients

myClients wird auf den Route-Reflektoren verwendet und beinhaltet alle anderen Router die den RR als RR benutzen werden. Hier sind also alle PE Router aufgelistet. Alle Router die VPNs zur Verfügung stellen würden, könnten oder wollten

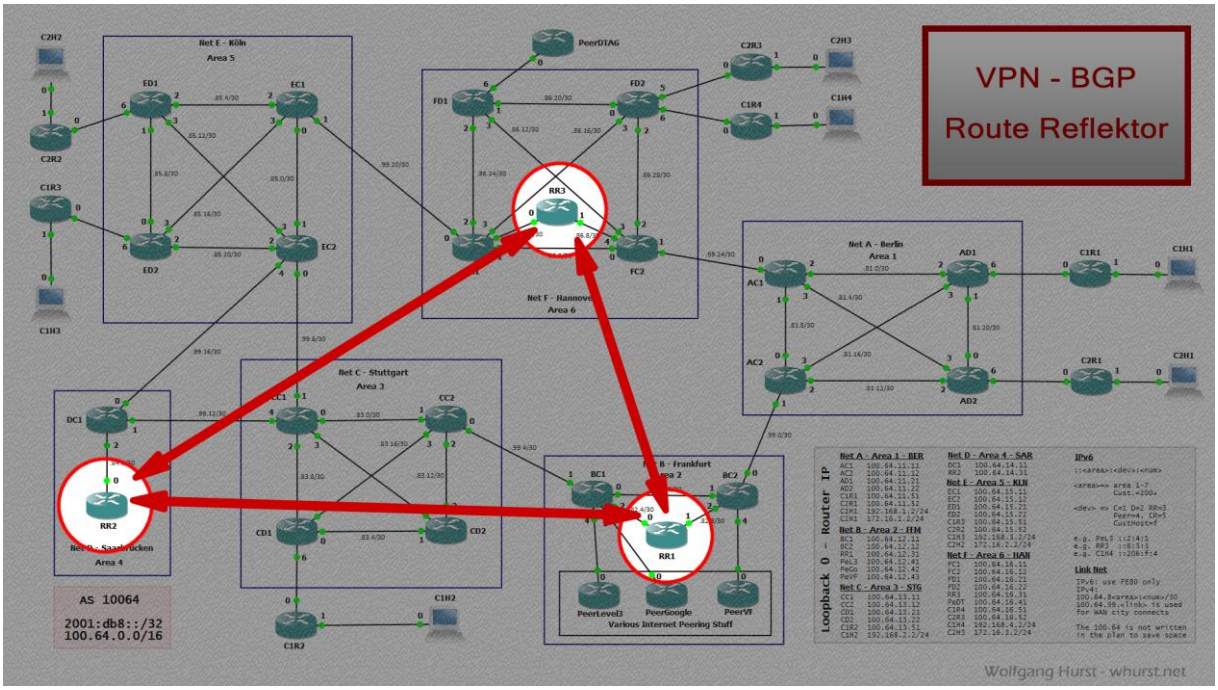
myRR

myRR wird auf den PE Routern verwendet und beinhaltet alle Route Reflektoren die er benutzen soll.

leRR

leRR benutzen die RR untereinander um sich gegenseitig zu verbinden.

6.3 BGP Route Reflektor



Wir werden nun den Route Reflektor einrichten. Dazu werden wir den RR so konfigurieren, dass jeder RR den anderen RR als Nachbarn hat, damit wird eine entsprechende Verteilung sichergestellt. Weiterhin konfigurieren wir die Clients (PE Router) damit sie den RR benutzen können.

Es ist zwingend erforderlich mehr als nur einen RR zu betreiben. Wenn wirklich mal einer ausfallen sollte, wäre wirklich alles Dunkel, daher habe ich im Netz drei RR aufgebaut.

Zuerst will ich das die RR untereinander sprechen können. Auf jedem RR muss man folgende Konfiguration anwenden, hier ein Beispiel vom RR3:

```
RR3#conf t
RR3(config)#router bgp 10064
RR3(config-router)#neighbor leRR4 peer-group
RR3(config-router)#neighbor leRR4 remote-as 10064
RR3(config-router)#neighbor leRR4 update-source Loopback0
RR3(config-router)#neighbor leRR4 soft-reconfiguration inbound
RR3(config-router)#address-family vpnv4
RR3(config-router-af)#neighbor leRR4 send-community extended
RR3(config-router-af)#neighbor 100.64.12.31 peer-group leRR4
RR3(config-router-af)#neighbor 100.64.14.31 peer-group leRR4
```

Wenn das soweit klappt, erstelle ich nun alle PE Router, hier ein Beispiel vom RR1:

```
RR3(config)#router bgp 10064
RR3(config-router)#neighbor myClients4 peer-group
RR3(config-router)#neighbor myClients4 remote-as 10064
RR3(config-router)#neighbor myClients4 update-source Loopback0
RR3(config-router)#neighbor myClients4 soft-reconfiguration inbound
RR3(config-router)#address-family vpnv4
RR3(config-router-af)#neighbor myClients4 route-reflector-client
RR3(config-router-af)#neighbor myClients4 send-community extended
RR3(config-router-af)#neighbor 100.64.11.21 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.11.22 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.13.21 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.13.22 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.15.21 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.15.22 peer-group myClients4
```

```
RR3(config-router-af)#neighbor 100.64.16.21 peer-group myClients4
RR3(config-router-af)#neighbor 100.64.16.22 peer-group myClients4
```

Man beachte das die Clients in die `address-family vpnv4` reinkommen, die Clients tauschen nur VPN Routen aus und keine globalen Routen. Dummerweise kann das Image was ich verwende noch kein vpnv6, das fehlt daher.

`update-source Loopback0` Damit geben wir an, dass der BGP Prozess mit der Loopback 0 kommunizieren soll, und nicht mit irgendeiner IP Adresse aus dem System. Das würde sonst zu Verwirrungen führen

`soft-reconfiguration inbound` Der Befehl hat ein Vorteil und ein Nachteil. Er legt im Speicher ein Spiegel aller Routen an und wendet neue Regeln auf den gesamten Spiegel an und alle Routen bekommen es mit

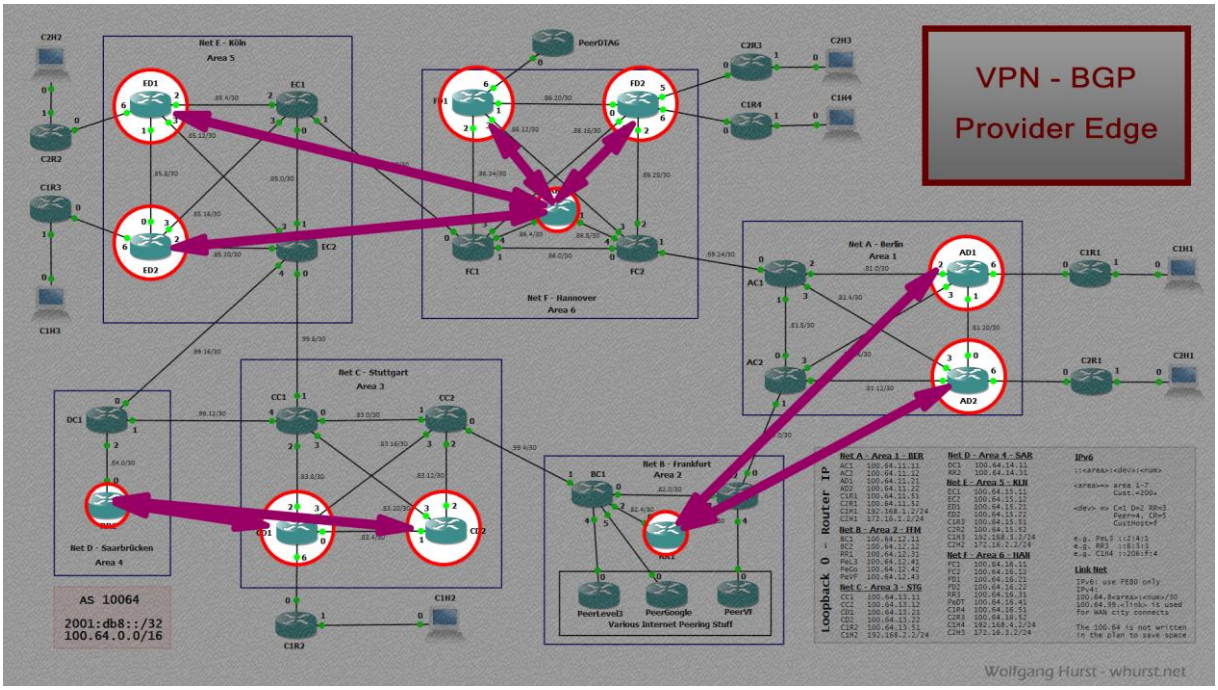
`send-community extended` Damit wird das senden und empfangen von Communities aktiviert. Die brauchen wir noch.

`route-reflector-client` Wird benötigt damit die Routen ausgetauscht werden. Fehlt der Befehl empfangen die Clients keine Routen von den anderen. Damit wird quasi das Teilen der Routen aktiviert

Insbesondere `soft-reconfiguration inbound` könnte bei sehr vielen Routen zu einem sehr großen Problem führen. Wenn man sich 100% Sicher ist, dass man keine Policy-Maps/Route-Maps oder nur selten mal welche ändert, könnte man auf den Befehl verzichten. Das Problem ist, das beim Ändern einer Policy oder einer anderen Route-Map, nur neue Routen entsprechend geändert werden. Die alten bleiben so drin, wie sie waren. Nur wenn mal alle Routen mittels `clear` löscht, wird eine neue oder geänderte Map beachtet. Das kann manchmal gut sein, manchmal nicht.

Ein Route Reflektor sollte keine Maps oder ACLs oder Policies haben. Er sollte nur stumpf alles was er hat ungefiltert weiterreichen. In unserem Test Netz aber habe ich es aktiviert, damit man schneller mal kurz eben was Testen kann, ohne ein Reset.

6.4 BGP Provider Edge



Nun gehen wir auf die PE Router und aktivieren dort das BGP. Wir sprechen dann mit dem Route-Reflektor. Ich werde dabei im Netzwerk eine Verteilung vornehmen. Die kann man später wunderbar zum Debuggen oder zum Testen nehmen. Je nach Größe des Netzwerkes, kann auch jeder jeden RR benutzen.

Area 1 und 2 benutzen den Route Reflektor 1 und als zweiten den RR2

Area 3 und 4 benutzen den RR2 und als zweiten den RR3

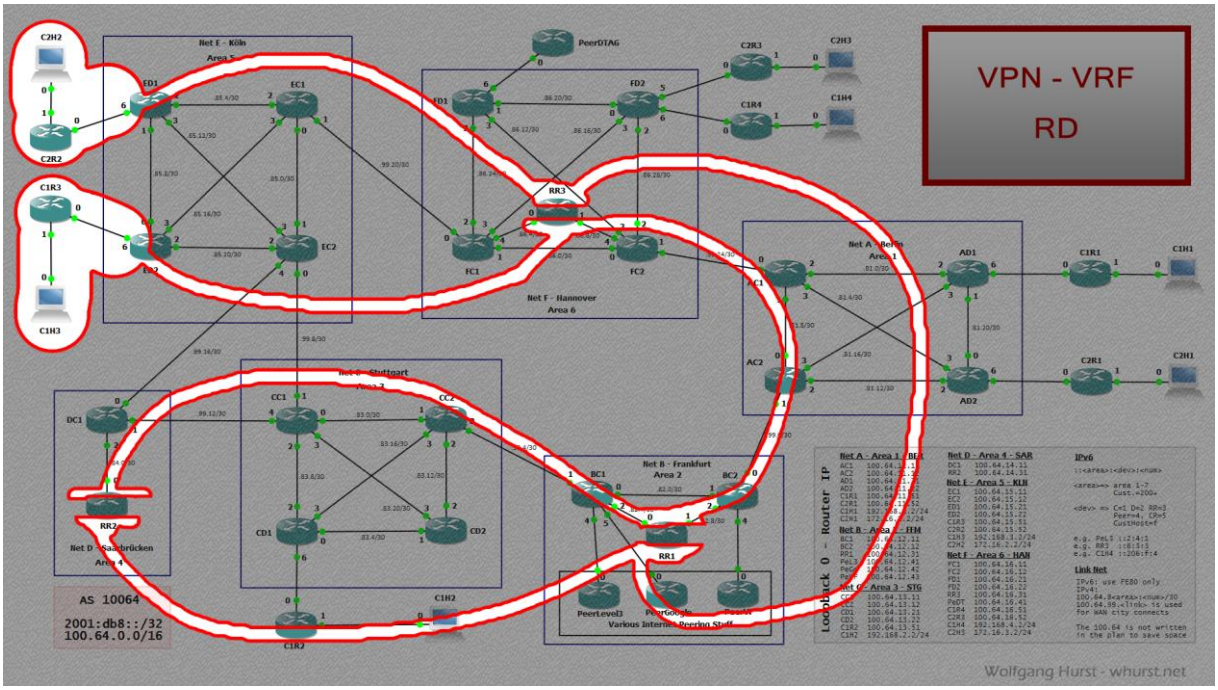
Area 5 und 6 benutzen den RR3 und als zweiten den RR1

Hier ein Beispiel:

```
AD1#conf t
AD1(config)#router bgp 10064
AD1(config-router-af)#neighbor myRR4 peer-group
AD1(config-router-af)#neighbor myRR4 remote-as 10064
AD1(config-router-af)#neighbor myRR4 update-source Loopback0
AD1(config-router-af)#neighbor myRR4 soft-reconfiguration inbound
AD1(config-router)#address-family vpnv4
AD1(config-router-af)#neighbor myRR4 send-community extended
AD1(config-router-af)#neighbor 100.64.12.31 peer-group myRR4
AD1(config-router-af)#neighbor 100.64.14.31 peer-group myRR4
```

Noch liegen ja keine Routen im BGP drin, von daher ist es auch noch nicht so spannend. Wichtig ist erst einmal das das BGP rennt und jeder fehlerfrei funktioniert. Auch hier brauchen wir kein globales BGP, sondern nur die für die VPNs

6.5 VRF RD



Der Route Reflektor hat ja nun alle Routen von allen Routern in der Routingtabelle. Das ist zwar ganz nett, hat aber ein Nachteil. Wenn man nun zwei Kunden hat, was ja durch aus passieren kann, gibt es ein Problem. Die Routen der VPNs würden sich jetzt vermischen. Man darf davon ausgehen das den Kunden das womöglich nicht gefällt.

Wir müssen also irgendwie diese Routen trennen. Man könnte jetzt für jedes VPN ein neuen RR aufbauen, aber das ist dann doch schon etwas Overkill. Nein wir können uns eines bereits vorhandenen Mechanismus bedienen. Wir nehmen ein RD.

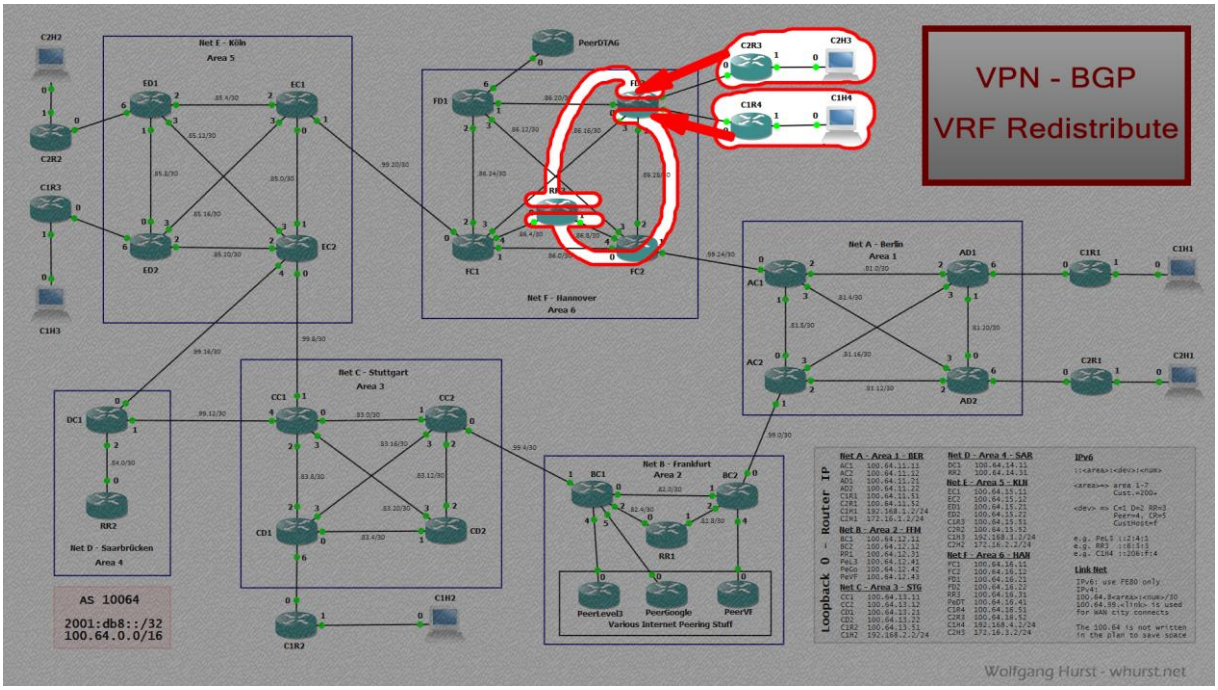
Mit einem RD kennzeichnen wir alle Routen innerhalb eines VRF. Wenn dann die Routen zum RR gesendet werden, wird immer der RD mittransportiert. Effektiv hat der RR dann quasi pro RD eine eigene Tabelle.

Mit einem RD wird nur die Routing Tabelle getrennt. Und genau diesen müssen wir im VRF angeben. Hinzukommt noch die explizite Anweisung die Routen aus dem RD zu importieren und exportieren. Achtung das passiert nicht automatisch.

Hier mal eine Beispiel Konfiguration vom AD1:

```
AD1#conf t
AD1(config)#ip vrf vpn01
AD1(config-vrf)#rd 1000:1
AD1(config-vrf)#route-target export 1000:1
AD1(config-vrf)#route-target import 1000:1
```

6.6 BGP VRF



So und jetzt kommt der letzte Akt. Wenn man diesen Schritt abgeschlossen hat, funktioniert das MPLS VPN. Weil das einzige was jetzt noch fehlt, eigentlich schon die ganze Zeit, ist eine Datenbank mit Netzwerken und Labels, die sich alle PE Router gegenseitig teilen können. Die Grundlage haben wir mit dem RR geschaffen. Und jetzt müssen wir nur noch die Routen aus dem VPN in das BGP werfen. Das wars.

Der Vorgang entspricht exakt dem, wie man es im globalen Routing machen würde. Man geht in die BGP Konfiguration und sagt einfach eine Redistribution. Nur so einfach geht das hier nicht ... Wir müssen es im VRF tun, weil sonst die Routen im globalen Routing wären.

Hier das Beispiel für den AD1:

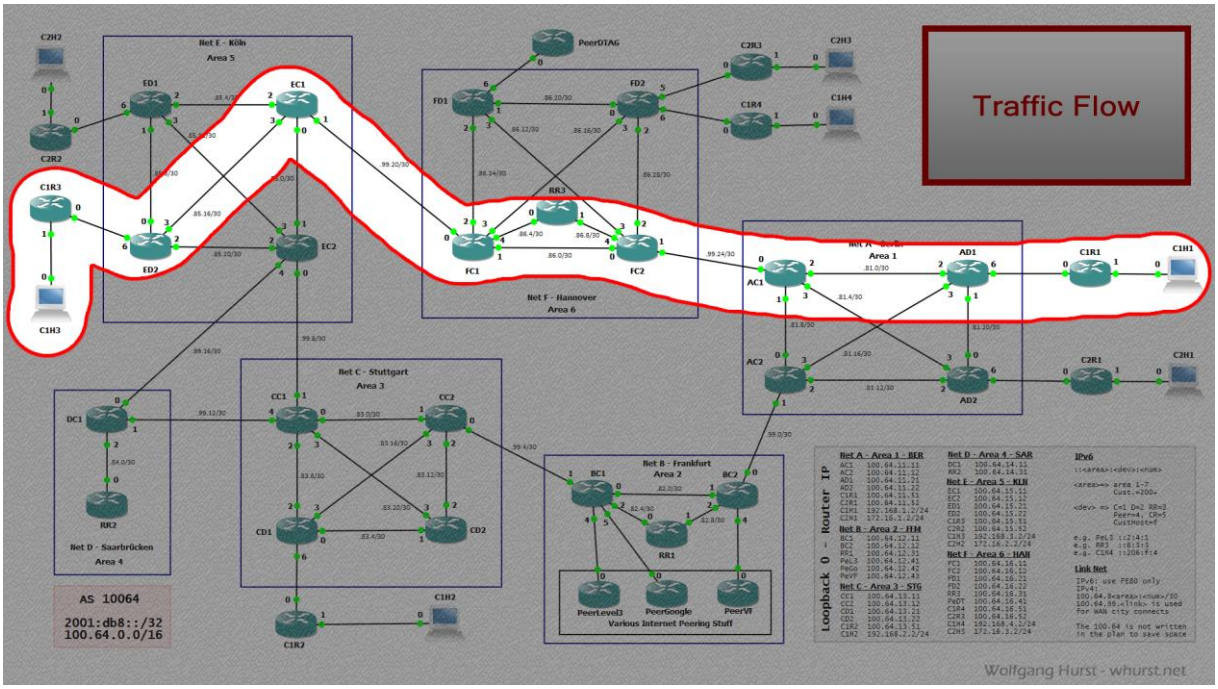
```
AD1#conf t
AD1(config)#router bgp 10064
AD1(config-router)#address-family ipv4 vrf vpn01
AD1(config-router-af)#redistribute connected
AD1(config-router-af)#redistribute static
```

Nun landet das Linknetz und auch unsere Statische Route in das Kundennetzwerk im BGP. Mit `show bgp vpnv4 unicast all` sieht man sie. Die internen Labels kann man mit `show ip bgp vpnv4 all labels` sich anschauen.

Wenn man nun auf dem PE im anderen Standort ebenfalls die Routen in das BGP wirft, dann hat man ein funktionsfähiges VPN über MPLS aufgebaut. Alle Hosts können alle anderen Hosts innerhalb des Netzwerkes über unser Netzwerk erreichen.

7 Zusammenfassung

7.1 Traffic Flow



Hier noch einmal eine Zusammenfassung die den Traffic Flow beschreibt. Am Beispiel eines PINGs von C1H1 auf C1H3

Wir pingen von C1H1 zum C1H3

1) C1H1 sucht in seiner Routingtabelle das Ziel und findet eine Default Route zum C1R1
2) C1R1 sucht das Ziel in seiner Routingtabelle und findet auch eine Default Route zum AD1
3) AD1 bekommt auf dem Interface 6 ein Paket geliefert
4) AD1 nimmt das Paket und er weiß er muss es in einem VRF Context werfen. Wer wirft es in das VPN01
5) AD1:VPN01 sucht das Ziel in der Routingtabelle, er filtert nach dem RD
6) AD1:VPN01 findet das Ziel in der BGP VPNv4 Routing Tabelle
7) AD1:VPN01 setzt das Label aus der BGP VPNv4 Routing Tabelle vor das Paket
8) AD1:VPN01 merkt sich die Ziel Router IP (ED2) aus den Globalen Routing
9) AD1:VPN01 übergibt das Paket in das Globale Routing, mit der Information des Ziel Routers
10) AD1 bekommt im globalen Routing das VPN Gelabelte Paket
11) AD1 sucht in der Label Forwarding Tabelle nach dem Ziel ED2
12) AD1 findet das Ziel und ein Label vom AC1
13) AD1 schreibt das Label vor das Paket und sendet es an AC1
14) AC1 bekommt ein Paket mit einem Label (es sind zwei - aber das juckt ihn nicht)

15) AC1 schaut in seine Label Forwarding Tabelle und findet das Label
16) AC1 schreibt das Label um und sendet es an FC2, weil es so in der LFT steht
17) FC2, FC1, EC1 tun das gleiche
18) ED2 bekommt ein Paket und merkt er selbst ist das Ziel
19) ED2 entfernt das Label (das erste)
20) ED2 sieht ein weiteres Label in diesem Paket und geht von einem VPN aus
21) ED2 schaut in seiner Label Forwarding Tabelle nach diesem Label
22) ED2 sieht es gehört dem VPN01
23) ED2 übergibt das Paket an den Virtuellen Router - ins VPN
24) ED2:VPN01 entfernt das Label (das letzte)
25) ED2:VPN01 schaut im Paket (IP) nun nach der Ziel-IP (C1H3)
26) ED2:VPN01 findet ein Statischen Routing Eintrag und stellt es C1R3 zu
27) C1R3 nimmt das Paket entgegen und sucht das Ziel in der Routingtabelle
28) C1R3 findet C1H3 am Interface 1 und stellt es zu
29) C1H3 bekommt den Ping

7.2 Ende

Sie sind am Ende des Dokumentes angekommen.

Alle Konfigurationen von allen Routern in allen Schritten findet man auf meiner Webseite. Dort kann man bei jedem Schritt die Konfiguration in den GNS laden und Experimente machen.

Vielen Dank für Ihre Aufmerksamkeit und fröhliches Labeln

Wolfgang Hurst

https://whurst.net