1. Freifunk Rheinland e.V. Routing-Days

Learn to build the Internet

#routingdays

Wer sind wir?

Das Freifunk Rheinland e.V. AS 201701 Admin Team

V. l. n. r.:

- ➔ Lars
- ➔ ThomasDOTwtf
- ➔ takt
- ➔ Barbarossa

Lars

- ➔ Lars Bußmann - freifunk@kill-you.net
- ➔ Unix/Linux Systemadministrator (arvato systems GmbH)
- ➔ AS_PATH: 201701, Freifunk Möhne

Thomas

- ➔ Thomas Drewermann - thomas@drewermann.org
- ➔ Cloud Architekt (ahd hellweg data GmbH & Co. KG)
- ➔ Autonomes System: AS197965
- ➔ AS_PATH: 197965, 201701
- ➔ Twitter: @thomasDOTwtf

takt

- Oliver Herms - takt@takt.guru
- Google Network Engineer (Global Prod. Ops, Tbit/s Scale, takt@google.com)
- Autonomes System: AS199714
- AS_PATH: 15169, 201701, 8560, 199714, 12581, 20773, 8422
- Twitter: @taktv6

Max

- ➔ Maximilian Wilhelm - max@rfc7511.org
- ➔ Senior Systems Engineer (DevOps, Linux, Storage, Netzwerk, Uni Paderborn)
- ➔ AS_PATH: 201701, Freifunk Hochstift, UPB
- ➔ Twitter: @BarbarossaTM

Frühsport / Zum Wachwerden

- Skillcheck / Selbsteinschätzung
- Gruppenbildung basierend aus Wissenständen
  - jeweils 4 Teilnehmer
  - jeweils 4 VMs
  - jeder kann sich um ein System kümmern (z.B.)
  - jeder hat Root-Access auf alle Systeme
- Umsetzen in die Gruppen
- Gruppen aufgeteilt nach Reihen, von vorne linke nach hinten rechts, 0 bis n
- Be excellent to each other

Programm

- Basics
- Statisches Routing
- Bird
- OSPF
- BGP
- Policy Based Routing
- NAT
- Debugging

Grundsätzliches

- Straffes Programm
- Bei Fragen oder Unklarheiten
  - melden
  - Teammitglieder / Nachbarn fragen
-

Disclaimer: Follow the white penguin.

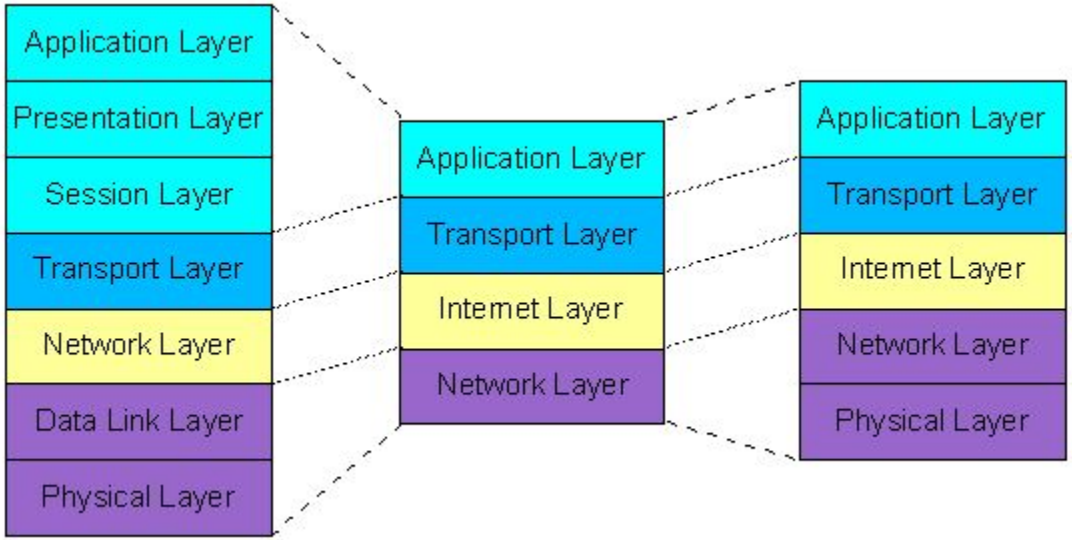
- IP / Routing-Grundlagen sind plattformunabhängig
- Wir arbeiten hier mit Linux
  - Es funktioniert
  - Wenn nicht, kann man es fixen
  - Alle Features und Protokolle frei verfügbar
  - Volle Flexibilität
- Auf Cisco, Juniper, etc. unterscheiden sich nur die Befehle (und der Preis)

Session 1: IP Grundlagen

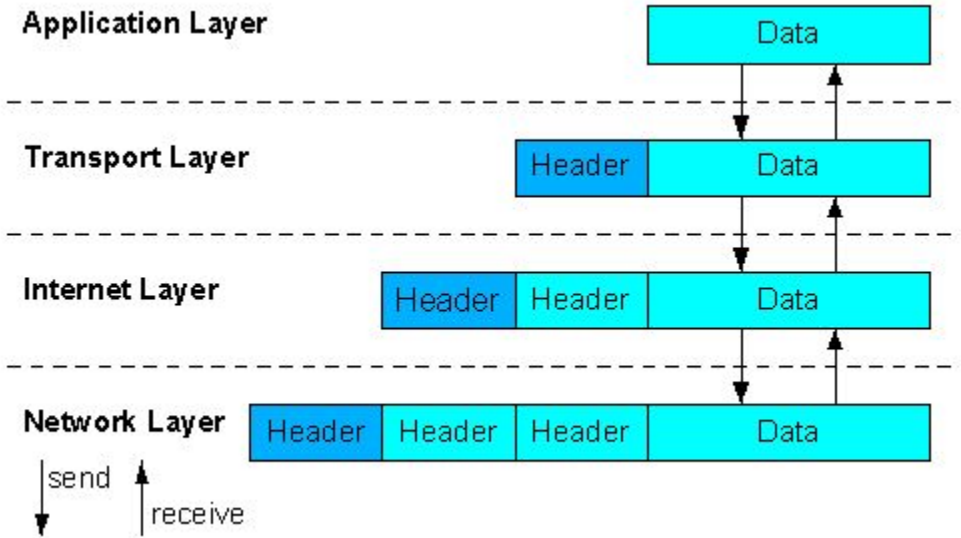
Agenda

- ➔ Schichtenmodell(e)
- ➔ Switching
- ➔ IPv4 / IPv6
- ➔ IP Header
- ➔ Subnetting
- ➔ Historischer Exkurs Netzklassen
- ➔ Routing
- ➔ Literaturhinweise

Schichtenmodelle - ISO/OSI, TCP/IP & Hybrid



Kapselung von Datenpaketen



Quelle: Hunt C.: TCP/IP Network Administration. O'Reilly & Assoc., Sebastopol, CA, 1995

Switching

- Layer2 wirft Frames von rechts nach links
- Forward, Replicate or Drop
- Skaliert nicht
- Layer2 ist tot.
- <Schweigeminute/>

Legacy IP, IPv6 and the state of the InterNAT

- IPv4 ist alt, alle und macht allein nicht mehr glücklich
- IPv6
  - wurde 1995 erstmals definiert
  - 1998 einmal überholt (RFC2460)
  - Bringt für unsere Betrachtungen heute kaum Unterschiede, da weitgehend analog
- Der Einfachheit ***heute und hier*** ausschliesslich IPv4
- Neue Installationen oder Updates bestehender Infrastrukturen ***immer*** Dual-Stack bauen!

IP Header

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

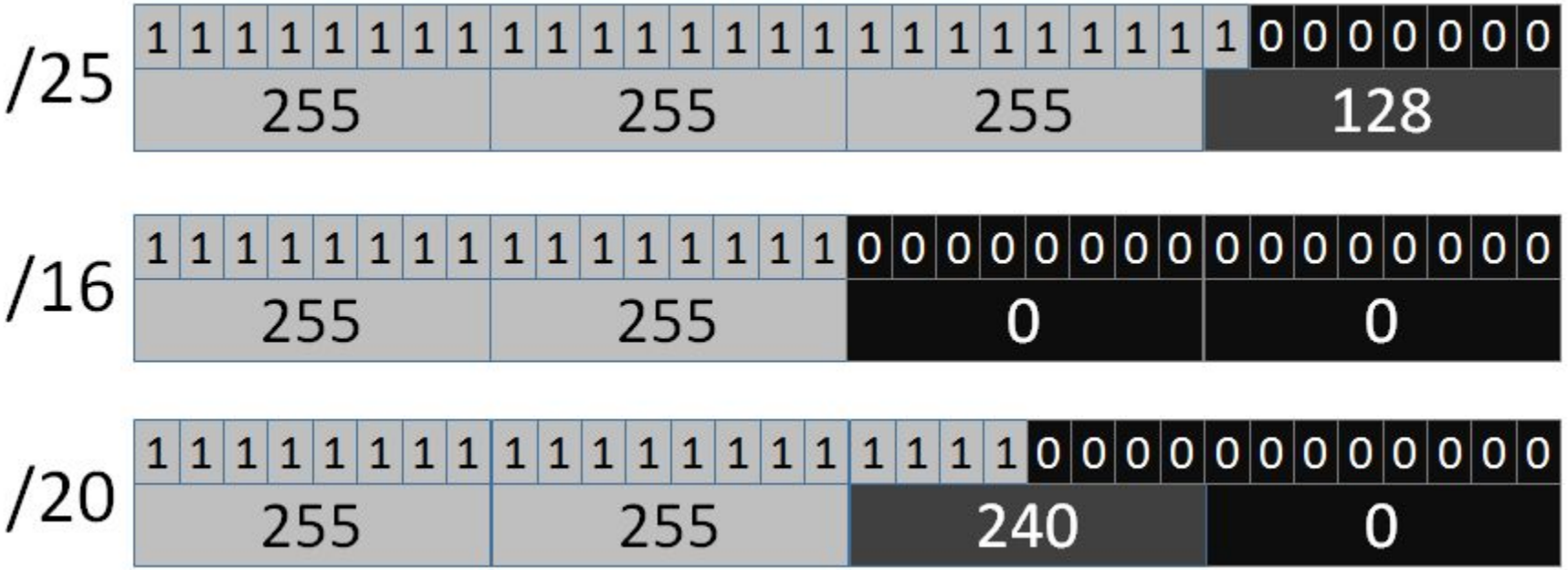
Quelle: https://en.wikipedia.org/wiki/IPv4#Header

Historischer Exkurs: Netzklassen

- Netzklassen sind seit 1993 (RFC1519) deprecated!!1!!elf
- Es lebe Classless InterDomain Routing (CIDR) bzw. die Variable Length Subnet Mask (VLSM)
- Die korrekte und vollständige Definition zur historischen Einordnung!

Klasse	Binary Prefix	IP Space	Default Netmask
A	0...	0.0.0.0 - 127.255.255.255	/8
B	10..	128.0.0.0 - 191.255.255.255	/16
C	11..	192.0.0.0 - 223.255.255.255	/24
D	111.	224.0.0.0 - 239.255.255.255	
E	1111	240.0.0.0 - 255.255.255.255	

Subnetzmasken / Subnetting



Prefixes to know / Private stuff

- Loopback
  - 127.0.0.0/8
- RFC 1918
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192168.0.0/16
- APIPIA / Link local
  - 169.254.0.0/16
- RFC 6598
  - 100.64.0.0/10

Routing

- Weg für IP Pakete von a nach b bestimmen
- Wegbestimmung jeweils aus lokaler Sicht auf das Netzwerk
- Informationen dazu in Routing-Tabelle

Prefix	Weg / Next Hop
192.168.42.0/24	eth0
0.0.0.0/0	via 192.168.42.1

- Technically: Pakete von a nach b weiterleiten = Forwarding
- Jeder Router, der ein Paket weiterleitet, dekrementiert TTL um 1

Netzwerkkonfiguration unter Linux

- R.I.P. ifconfig
- R.I.P. route
- R.I.P. arp
- R.I.P. vlan
- R.I.P. brctl
- R.I.P. tunctl
- <Schweigeminute/>
- Lang lebe iproute2

iproute - Schweizer Netzwerker Taschenmesser

- Funktionen sauber getrennt in Unterbefehle
- Erlaubt Abkürzung der Befehle für Tippfaule :)
- ip link
  - Layer2 Konfguration
  - Vlans
  - Birdges
- ip addr
  - Layer3 Konfiguration / IP-Adressen
- ip route
  - Routing
- ip neigh
  - ARP / ND

ip link

```
ip link set { DEVICE | dev DEVICE } [ { up | down } ]
    [ promisc { on | off } ]
    [ name NEWNAME ]
    [ address LLADDR ]
    [ mtu MTU ]
    [ master DEVICE ]
    [ nomaster ]
    [...]

ip link show [ DEVICE ]
```

ip addr

```
Usage: ip addr {add|change|replace} IFADDR dev STRING [ LIFETIME ] [...]

    ip addr del IFADDR dev STRING

    ip addr {show|flush} [ dev STRING ] [ scope SCOPE-ID ]
                         [ to PREFIX ] [ FLAG-LIST ] [ label PATTERN ]

IFADDR := PREFIX | ADDR peer PREFIX
        [ broadcast ADDR ] [ anycast ADDR ]
        [ label STRING ] [ scope SCOPE-ID ]

SCOPE-ID := [ host | link | global | NUMBER ]

[...]
```

ip route

Usage: ip route { list | flush } SELECTOR

ip route { add | del | change | append | replace } ROUTE
SELECTOR := [root PREFIX] [match PREFIX] [exact PREFIX]
[table TABLE_ID] [proto RTPROTO] [type TYPE] [scope SCOPE]

ROUTE := NODE_SPEC [INFO_SPEC]

NODE_SPEC := [TYPE] PREFIX [tos TOS]
[table TABLE_ID] [proto RTPROTO] [metric METRIC]
INFO_SPEC := NH OPTIONS FLAGS [nexthop NH]...
NH := [via ADDRESS] [dev STRING] [weight NUMBER] NHFLAGS

ip neigh

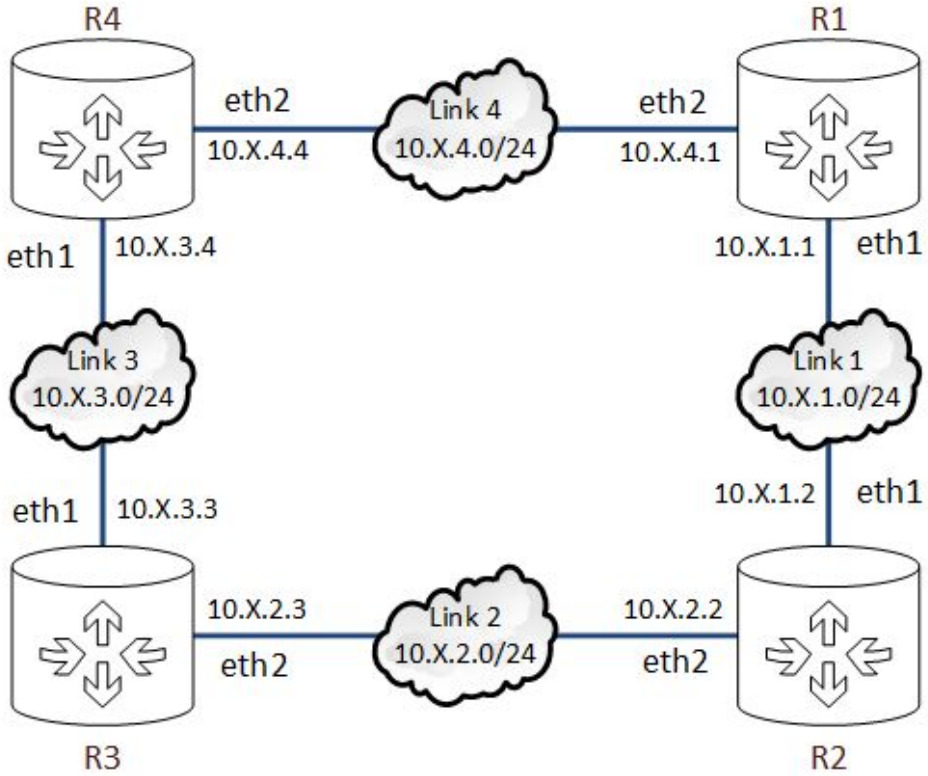
```
Usage: ip neigh { add | del | change | replace } { ADDR [
lladdr LLADDR ]

       [ nud { permanent | noarp | stale | reachable } ]
       | proxy ADDR } [ dev DEV ]
   ip neigh {show|flush} [ to PREFIX ] [ dev DEV ] [ nud
STATE ]
```

Lab 1: Static Routing

Agenda

- ➔ Netzplan
- ➔ Reachability Test
- ➔ Debugging :)
- ➔ Transfer / Loopback Reachability
- ➔ Routen reboot-fest machen (manuell)
- ➔ Statische Routen mit Bird

Netzplan


R4
R1
eth2
10.X.4.4
Link 4
10.X.4.0/24
eth2
10.X.4.1
eth1
10.X.3.4
10.X.1.1
eth1
Link 3
10.X.3.0/24
Link 1
10.X.1.0/24
eth1
10.X.3.3
10.X.1.2
eth1
10.X.2.3
eth2
Link 2
10.X.2.0/24
10.X.2.2
eth2
R3
R2

Please kindly do the needful

- http://tea.systems/workbooks
- Systeme erreichbar über:
- router-X-Y.r.tea.systems
- IPv6 only!11!elf
- sudo verwenden!
- `User: user`
- `Pass: user`
- Loopback IPs: 10.Gruppe.0.RouterNr

ARP

- Funktioniert IMMER **NUR** in lokal angeschlossenen Segmenten
- ARP-Requests und Antworten werden NICHT weitergeleitet

Routen reboot-fest machen

I. Start-up-Skript bauen bzw. Routen in /etc/rc.local eintragen
   -> Doofe Idee, da Routen verloren gehen, wenn Interface(s) um-/dekonfiguriert werden

II. Cron-Job bauen, der Routen alle $n Minuten einträgt, falls nicht vorhanden
   -> Minimal weniger doofe Idee, löst ggf. Probleme von I. (mit Delay)
   Garantiert trotzdem nicht, dass Routen immer da sind

III. Routen in Netzwerk-Konfigurationsdatei eintragen
   -> Gute Idee, da bei jeder Um-/Rekonfiguration wieder eingetragen :-)

Netzwerkconfig unter Debian

/etc/network/interfaces

```
iface eth1 inet static
    address 10.0.1.1
    netmask 24
    #
    post-up ip route add 10.0.0.2/32 via 10.0.1.2
    post-up ip route add 10.0.0.3/32 via 10.0.1.2
```

Erkurs: Bird Internet Routing Dameon

- Erlaubt Verwalten von Routen durch eigenen Daemon
- Unterstützt dynamische Routingprotokolle (u.a. OSPF, BGP)
- Grundlage für alle Labs
- Für IPv4/IPv6 jeweils ein Daemon: ‘bird’ bzw. ‘bird6’
- Konfiguration über /etc/bird/*
- Verwaltung mit ‘birdc’ bzw. ‘birdc6’
- Homepage: http://bird.network.cz/
- Debian APT Repository: http://bird.network.cz/?download&tdir=debian/
- Version >= 1.5.0 nehmen (OSPF-rewrite hat die Welt besser gemacht :))

/etc/bird/bird.conf

```
# IP Adresse des Loopback-Interfaces (eindeutiger Identifier für den
Router)
router id 10.0.0.1;

protocol kernel {
#       learn;            # Learn all alien routes from the kernel
#       persist;          # Don't remove routes on bird shutdown
        scan time 20;     # Scan kernel routing table every 20 seconds
#       import none;      # Default is import all
        export all;       # Default is export none
#       kernel table 5; # Kernel table to synchronize with (df: main)
}
```

/etc/bird/conf.conf (Ctd.)

```
# This pseudo-protocol watches all interface up/down events.
protocol device {
    scan time 10;           # Scan interfaces every 10 seconds
}

protocol static {
    route 10.0.0.2/32 via 10.0.1.2;
    [...]
}
```

=> http://tea.systems/bird.conf

import / export von Routen

- Routen kommen in Bird “in einem Protokoll an”
- **`import`** regelt Übernahme der Protokoll-Routen in Bird-Routingtabelle
- **`export`** regelt Übergabe von lokalen Routen an Protokoll
- Parameter
  - **`all`**
  - **`none`**
  - Filter
- Defaults:
  - **`import all`**
  - **`export none`**

birdc

- Interaktive CLI zur Steuerung des BIRD Daemons
  - Erlaubt Abkürzungen und Tab-Completion :)
  - Interaktive Hilfe mit ‘?’
  - Scriptbar

// Interaktiv

```
birdc> show route
```

// Mit cmdline Paramtern

```
# birdc “show route 0.0.0.0/0” | grep -c 'via 100.64'
```

birdc - Befehle für Protokolle

```
# Config neu laden
configure

# Konfigurierte Protokolle samt Status anzeigen
show protocols

# Protokoll (de)aktivieren
enable <protocol>
disable <protocol>
```

birdc - Befehle für Routen

```
# Routen anzeigen
show route [all] [primary]
show route for <prefix> [all]

# Prefixe eines Protokolls anzeigen lassen
show route protocol <protocol>

# Prefixe zählen lassen
show route [...] count
```

Pause

Session 2: OSPF - Open Shortest Path First

Agenda

- ➔ Was bedeutet dynamisches Routing?
- ➔ Verwendung
- ➔ Grundlagen
- ➔ Hello Protocol / Adjacencies
- ➔ Link-State Datenbank
- ➔ Shortest Path Tree
- ➔ Exkurs: Multi Area OSPF
- ➔ Literaturhinweise

OSPF: Was bedeutet dynamisches Routing?

- ➔ Routing Tabelle wird automatisch gebildet
- ➔ Setzt Informationsaustausch zwischen Routern voraus
- ➔ Routing Tabelle passt sich automatisch an die Umstände an

Zwei Varianten:

1. Distance Vector Protokoll (z. B. RIP)
   “Subnet X ist über Y mit Metric Z zu erreichen”
   => Mehr oder weniger bedeutungslos
2. Link-State Protokoll (z. B. OSPF, IS-IS)
   Datenbank enthält vollständiges Abbild des Netzwerks
   “Router A ist mit B und C mit Metric X verbunden”

OSPF: Verwendung

OSPF ist ein Interior Gateway Protocol (IGP)
=> Verwendung innerhalb eines Netzes

Einsatzbereiche:

- ➔ Enterprise
- ➔ Datacenter
- ➔ Internet Service Provider

Zwischen Netzen wird das Border Gateway Protocol verwendet.
(Dazu mehr am Sonntag)

OSPF: Grundlagen

1. Hello Protocol macht Nachbarn miteinander bekannt
2. Nachbarn tauschen Link-State Datenbanken aus
3. Router berechnet kürzesten Weg zu allen anderen Routern
4. Berechnete Routen werden installiert

OSPF Pakete werden über IP übertragen (Protocol 89)

OSPF: Hello Protocol

OSPF: Hello Protocol

Hello Protocol stellt sicher dass benachbarte Router sich kennen und bidirektionale Kommunikation möglich ist!

Hello Paket:

- ➔ Wird an Nachbarn (PtP) oder AllSPFRouters 224.0.0.5 (Broadcast) gesendet
- ➔ Hello Interval = 10s
- ➔ Dead Interval = Hello Interval * 4 = 40s
- ➔ Enthält Liste bekannter Nachbarn

OSPF: Voraussetzungen für Adjacencies

- ➔ IP Adressen an beiden Enden müssen im selben Subnetz liegen
- ➔ Konfiguration der Authentifizierung muss übereinstimmen
  - ◆ Keine
  - ◆ Klartext
  - ◆ MD5 (empfohlen)
- ➔ Konfigurierte Area muss identisch sein
- ➔ Hello und Dead Intervals müssen identisch sein
- ➔ MTU muss übereinstimmen[1]

[1] Keine zwingende Notwendigkeit. Datenbankaustausch scheitert aber erfahrungsgemäß bei Abweichung.

OSPF: Adjacency Zustände

Down - Startzustand

Init - Router hat Hello empfangen. Nachbar hat ihn aber nicht gelistet

2-Way - Router hat Hello empfangen. Nachbar hat ihn gelistet

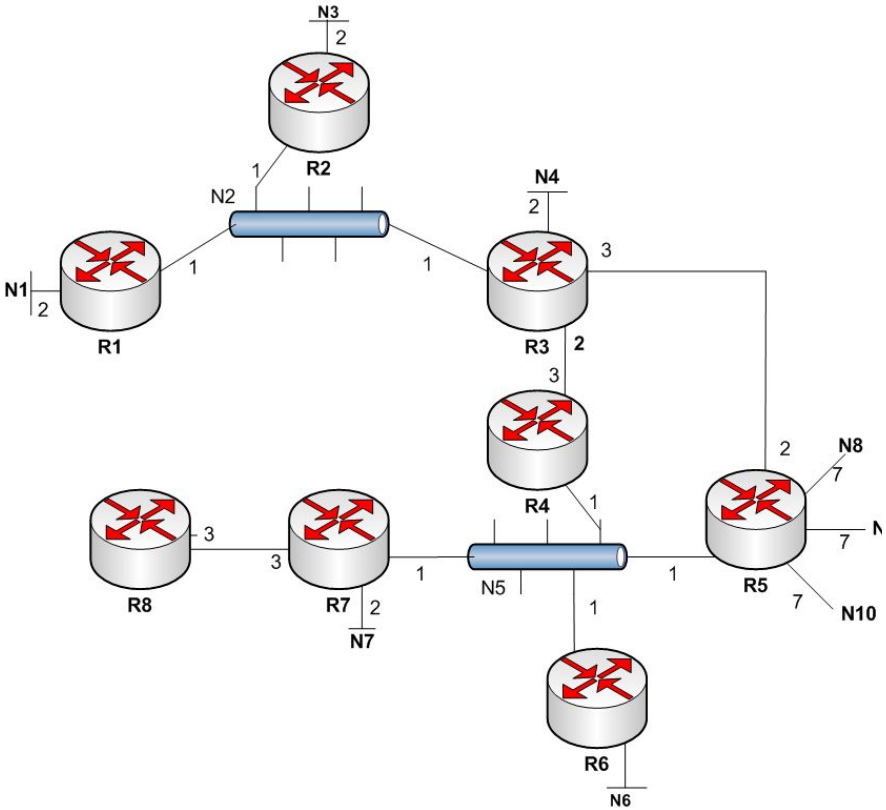
ExStart - Routers verhandeln wer Master sein wird für die Dauer

Exchange - Router tauschen Beschreibungen der Link-State Datenbank aus.

Loading - Router tauschen Link-State Datenbank aus

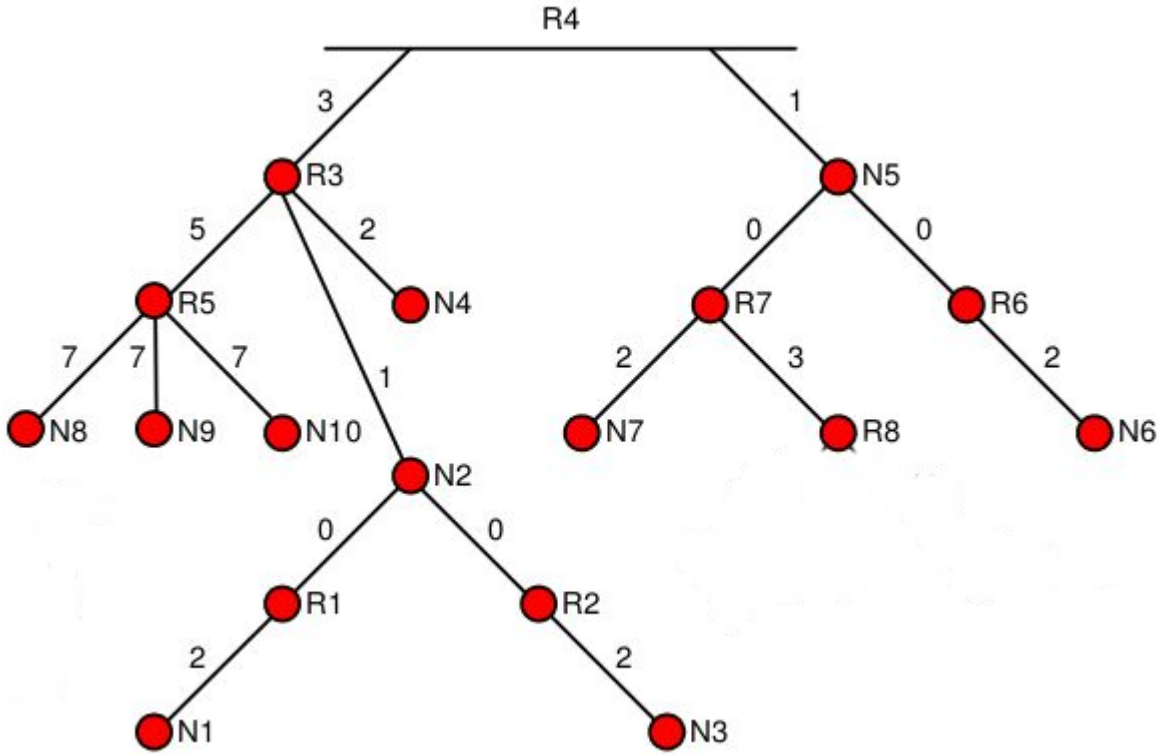
Full - Alle notwendigen Informationen sind ausgetauscht. Datenbank vollständig.

OSPF: Link-State Datenbank

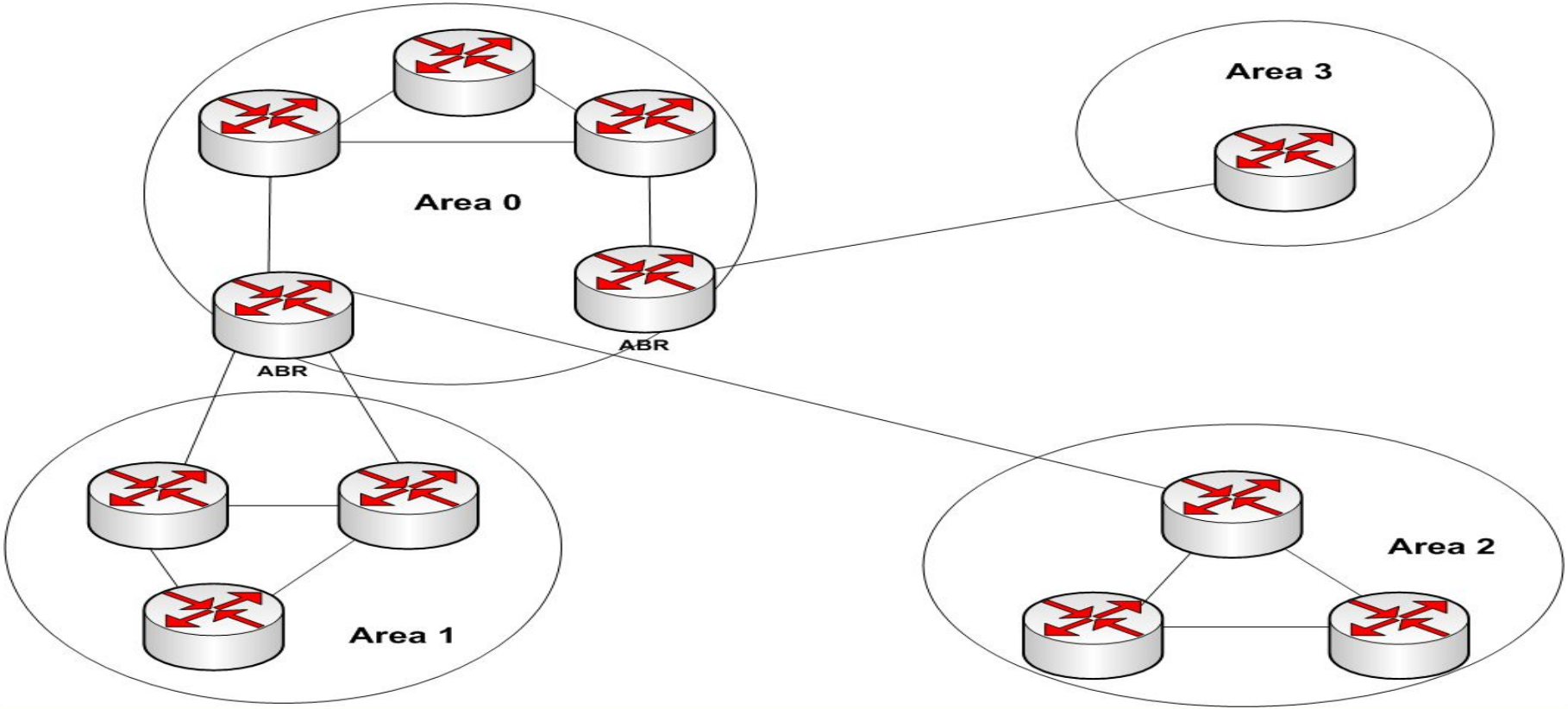


	R1	R2	R3	R4	R5	R6	R7	R8	N2	N5
R1	-	-	-	-	-	-	-	-	0	-
R2	-	-	-	-	-	-	-	-	0	-
R3	-	-	-	3	2	-	-	-	0	-
R4	-	-	2	-	-	-	-	-	-	0
R5	-	-	3	-	-	-	-	-	-	-
R6	-	-	-	-	-	-	-	-	-	0
R7	-	-	-	-	-	-	-	3	-	0
R8	-	-	-	-	-	-	3	-	-	-
N1	2	-	-	-	-	-	-	-	-	-
N2	1	1	1	-	-	-	-	-	-	-
N3	-	2		-	-	-	-	-	-	-
N4	-	-	2	-	-	-	-	-	-	-
N5	-	-	-	1	1	1	1	-	-	-
N6	-	-	-	-	-	2	-	-	-	-
N7	-	-	-	-	-	-	2	-	-	-
N8	-	-	-	-	7	-	-	-	-	-
N9	-	-	-	-	7	-	-	-	-	-
N10	-	-	-	-	7	-	-	-	-	-

OSPF: R4 Shortest Path Tree



Exkurs: OSPF Multi Area



OSPF: Multi Area

- ➔ Eine Link-State Datenbank je Area
- ➔ Alle Areas müssen mit Area 0 (Backbone) verbunden sein
- ➔ Virtuelle Links durch dritte Areas möglich
- ➔ Router mit Interfaces in mehreren Areas = Area Border Router (ABR)
- ➔ Ermöglicht das Filtern und Zusammenfassen von Routen

OSPF: Literatur

RFC2328: OSPF version 2 - John T. Moy, 1998

OSPF - Anatomy of an Internet Routing Protocol - John T. Moy, 1998

Routing in the Internet - Christian Huitema, 1999

Lab 2: OSPF mit BIRD https://goo.gl/69gEXT

Nützliche Kommandos:

- birdc - Startet BIRD shell
  - `show ospf interface`
  - `show ospf neighbors`
  - `show ospf state`
  - `show route protocol` *`IGP`*
- `ping -I <SRC_IP> -c 4 -i 0.5 <DST_IP>`
- `traceroute -n -s <SRC_IP> <DST_IP>`
- `mtr -n -a <SRC_IP> <DST_IP>`
- `ip link set down dev ethX`

OSPF mit BIRD

config example und Erklärbär auf der Tonspur?

Beering

Frühstück

Routing im Internet

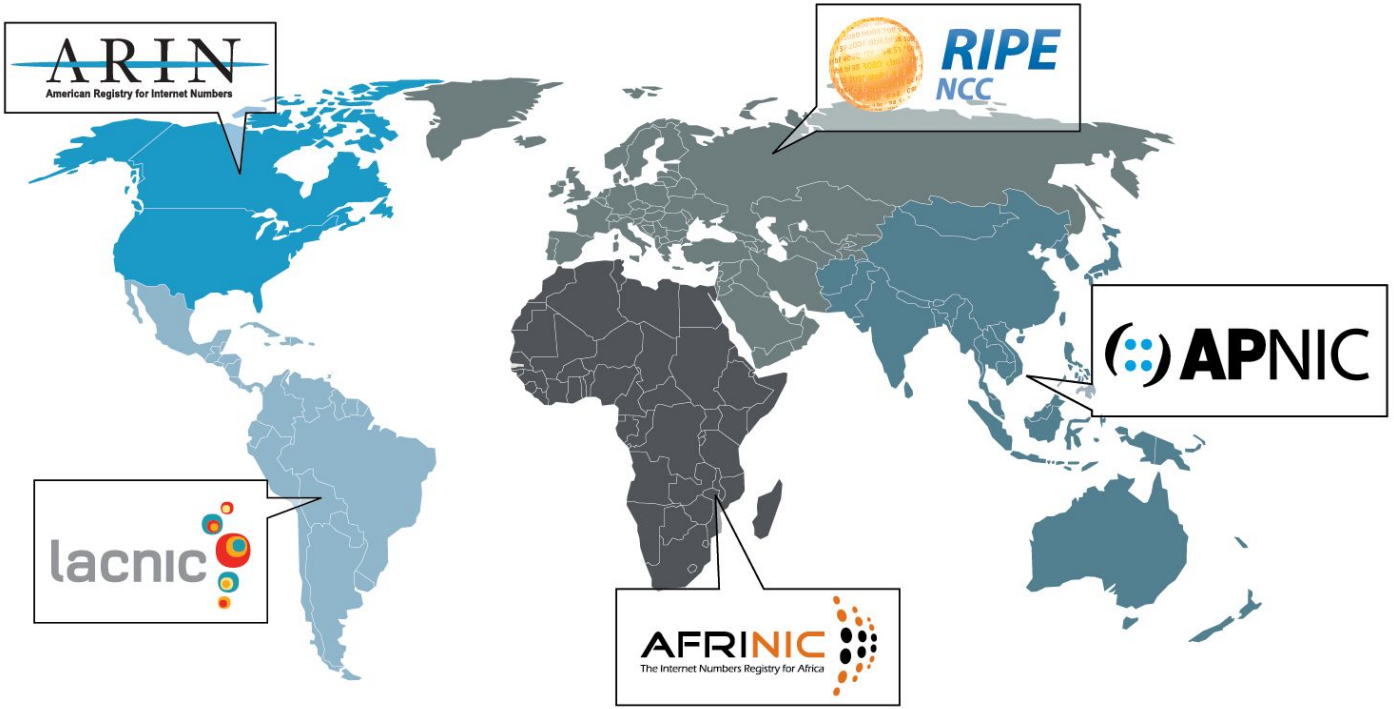
- Adressvergabe
- Autonome Systeme
- Transit
- Peering
- Internet Exchanges
- Politics

Thomas

Das Internet

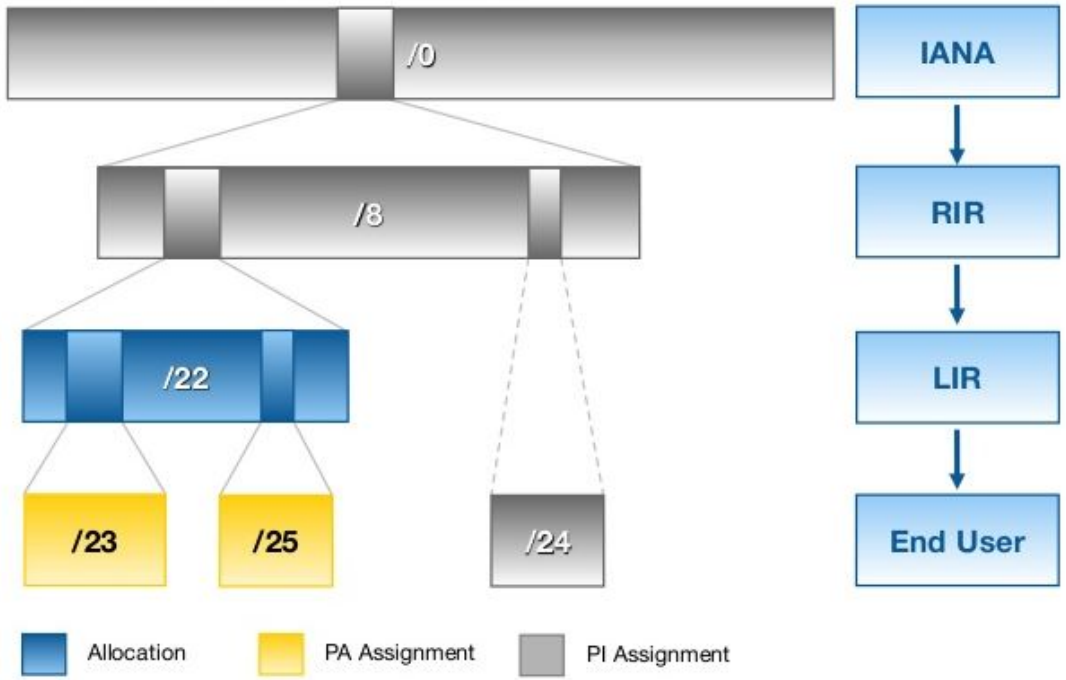
Guten Morgen Yoga-Session

Adressvergabe Weltweit



Quelle: xakep.ru

Vergabehierarchie IP-Space



Quelle: Obtaining IPv4 - Andrew de la Haye- RIPE Regional Meeting 2014

Home Scale Internet

- Einzelner ISP
- Lokales Netzwerk
  192.168.56.0 / 24
- Default Router
- Einfache Welt für den Router

Enterprise Scale Network

185.66.192.0/22

AS201701

gtt

TeliaSonera
International Carrier

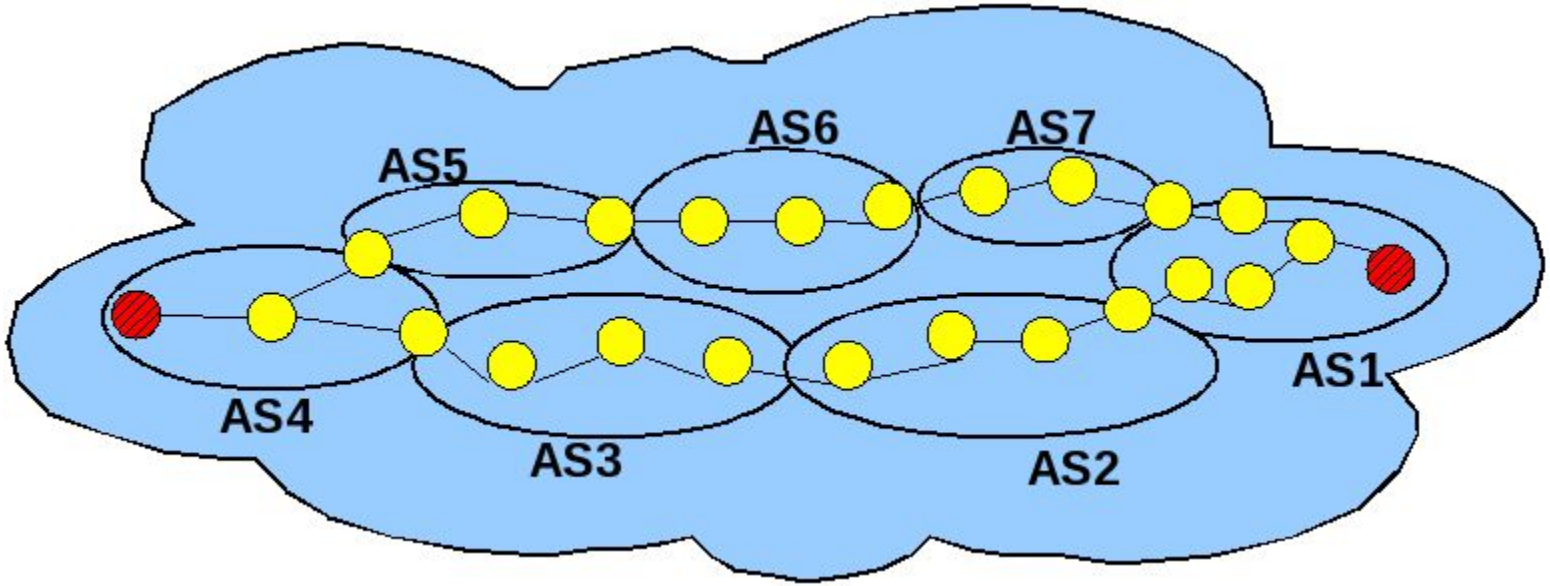
Level(3)

at&t

cogent
COMMUNICATIONS

Optical Internet

Default Free Zone - Autonome Systeme



Peering

Facebook AS32934

69.171.255.0/24	ethX
173.252.96.0/19	ethX
74.119.76.0/22	ethX

Apple AS714

17.253.216.0/23	ethZ
17.248.128.0/17	ethX
17.252.32.0/20	ethY

Kable Deutschland AS31334

146.52.192.0/18	ethZ
95.91.128.0/17	ethY
188.193.0.0/17	ethX

Peering

Facebook AS32934

69.171.255.0/24	ethX
173.252.96.0/19	ethX
74.119.76.0/22	ethX
17.248.128.0/17	AS714
17.252.32.0/20	AS714
17.249.0.0/16	AS714

Apple AS714

17.253.216.0/23	ethZ
17.248.128.0/17	ethX
17.252.32.0/20	ethY
173.252.96.0/19	AS32934
74.119.76.0/22	AS32934
204.15.20.0/22	AS32934

Kable Deutschland AS31334

146.52.192.0/18	ethZ
95.91.128.0/17	ethY
188.193.0.0/17	ethX

Peering

Facebook AS32934

69.171.255.0/24	ethX
173.252.96.0/19	ethX
74.119.76.0/22	ethX
17.248.128.0/17	AS714
17.252.32.0/20	AS714
17.249.0.0/16	AS714

Apple AS714

17.253.216.0/23	ethZ
17.248.128.0/17	ethX
17.252.32.0/20	ethY
173.252.96.0/19	AS32934
74.119.76.0/22	AS32934
204.15.20.0/22	AS32934
146.52.192.0/18	AS31334
95.91.128.0/17	AS31334
188.193.0.0/17	AS31334

Kable Deutschland AS31334

146.52.192.0/18	ethZ
95.91.128.0/17	ethY
188.193.0.0/17	ethX
17.253.216.0/23	AS714
17.248.128.0/17	AS714
17.252.32.0/20	AS714

Paid-Peering Politics

Session 3: BGP (extern)

Agenda

- ➔ Verwendung
- ➔ Zustandsautomat
- ➔ Attribute
- ➔ Routen Auswahl
- ➔ Filter, Adjacency RIB-IN, Adjacency RIB-OUT
- ➔ Literaturhinweise

BGP: Verwendung

BGP ist ein Exterior Gateway Protocol (EGP)
=> Verwendung zwischen Netzwerken

Einsatzbereiche:

- ➔ Enterprise
- ➔ Datacenter
- ➔ Internet Service Provider

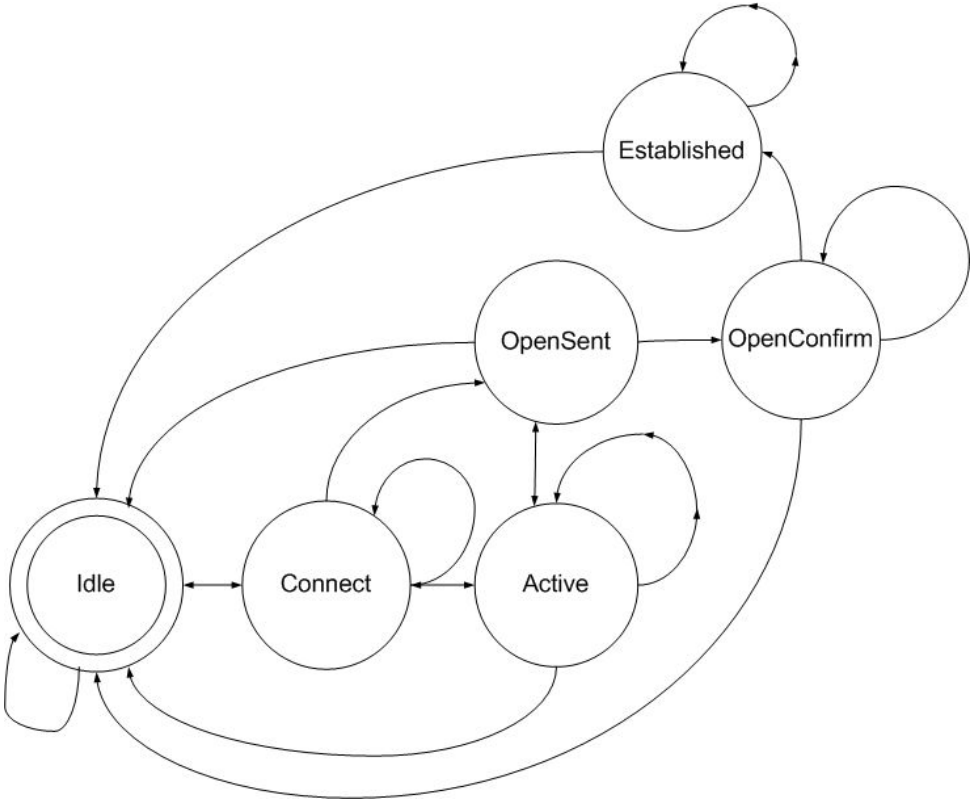
BGP: Grundlagen

1. Hello Protocol macht Nachbarn miteinander bekannt
2. Nachbarn tauschen Network Layer Reachability Information (NLRI) aus
3. Router selektiert beste Route
4. Selektierte Routen werden installiert

BGP verwendet TCP Port 179.

BGP: Zustandsautomat

- ➔ Idle: Startzustand
- ➔ Connect: Router wartet auf Verbindungsaufbau von Gegenseite
- ➔ Active: Router versucht Verbindung zu initiieren
- ➔ OpenSent: TCP Verbindung etabliert, OpenSent Nachricht versendet
- ➔ OpenConfirm: OpenSent Nachricht empfangen
- ➔ Established: Keep Alive empfangen Verbindung vollständig aufgebaut



BGP: Attribute

Jede Route bringt diverse Attribute mit:

- ➔ Next-Hop
- ➔ Local Preference
- ➔ AS Path
- ➔ Origin
- ➔ Multi Exit Discriminator (MED, optional)
- ➔ Communities (optional)

BGP: Next Hop

Jede Route muss irgendwohin zeigen…

- ➔ IPv4 Adresse des nächsten Routers
- ➔ Next Hop kann benachbarter Router sein
- ➔ Next Hop kann entfernter Router sein
  - ◆ Route zu entferntem Router z. B. per OSPF (Mehr dazu am Sonntag)
- ➔ Next Hop Adresse wird in Routing Tabelle nachgeschlagen

Ist der Next Hop nicht in der eigenen Routing Tabelle vorhanden, so kann die Route nicht verwendet werden.

BGP: Local Preference

- ➔ 32 bit unsigned int
- ➔ Wertebereich: 0 - 4294967295
- ➔ Standard: 100
- ➔ Gilt innerhalb eines AS
- ➔ Hoehere Werte = besser

Wird verwendet um Transit, Peering und Kunden zu unterscheiden.

Freifunk Rheinland e.V. AS 201701 Backbone:
500: Transit
1.000: Peering
10.000: Community

BGP: AS Path

- ➔ Liste aller Autonomen Systeme die eine Route durchwandert hat:
  - 49009 12731 39138 199714
  - Freifunk Hamburg, IPHH, rrbone, takt
- ➔ Kurze Pfade = besser
- ➔ Path prepending: 8881 3209 15924 15924 15924 15924 15924 15924 8386
- ➔ Dient auch der Vermeidung von Routing Loops
  - Router akzeptiert keinen Pfad der sein eigenes AS enthält!

BGP: Route Origin

I: IGP (1)

E: EGP (2)

?: Incomplete (2)

Kleiner = Besser

IGP > EGP > Incomplete

BGP: Multi Exit Discriminator (MED)

- ➔ 32 bit unsigned int
- ➔ Wertebereich: 0 - 4294967295
- ➔ Standard: 0
- ➔ Optional
- ➔ Nicht transitiv (wird nicht über Nachbar AS hinaus verbreitet)
- ➔ Kleinere Werte = besser

Wird verwendet um zu steuern, auf welchem Peering ein Nachbar Traffic senden soll.

Viele Netze überschreiben den MED beim Empfang einer Route:
Freifunk Rheinland e.V. AS201701, Google Inc. AS15169

BGP: Communities

- ➔ BGP Communities sind numerische Attribute an Routen
  - 64 bit
  - 32 bit ASN, 32 bit Wert
  - 201701:1234
- ➔ Well known Communities
  - no-export
  - no-advertise
- ➔ Bedeutung individuell je AS, z. B.
  - 201701:193 = Route in Düsseldorf gelernt
  - 201701:194 = Route in Frankfurt gelernt
  - 201701:195 = Route in Berlin gelernt
  - 201701:200 = Globale Route

BGP: Routenauswahl (nur eBGP)

1. Next Hop erreichbar?
2. > LOCAL_PREF
3. < AS_PATH
4. < ORIGIN
5. Falls mehrere Routen mit selbem Nachbarn: < MED
6. < Router ID
7. < Nachbar IP Adresse

BGP: Route Filter

- ➔ Nicht jede Route soll (unverändert) akzeptiert werden.
  - MED überschreiben
  - LOCAL PREF setzen
- ➔ Nicht jede Route soll (unverändert) gesendet werden.
  - MED setzen
  - Path prepending

Lösung: Route Filter

BGP: Receive filter

Dinge die man nicht empfangen und akzeptieren möchte:

- ➔ Den eigenen IP Adressbereich
- ➔ RFC1918 (192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8)
- ➔ RFC6598 (100.64.0.0/10)
- ➔ Multicast (224.0.0.0/3)
- ➔ Reserved (240.0.0.0/4)
- ➔ Default Route (0.0.0.0/0)
- ➔ Loopback (127.0.0.0/8)
- ➔ Documentation (192.0.0.0/24, 198.51.100.0/24, 203.0.113.0/24)

BGP: Announce filter

Dinge die man nicht senden sollte:

- ➔ Alles was man nicht empfangen mag, mit Ausnahme des eigenen Adressbereichs
- ➔ Peering/Transit:
  - ◆ Eigene Adressbereiche + Kunden Adressbereiche
- ➔ Kunden:
  - ◆ Default Route oder alles (“Full Table”)

BGP: Literatur

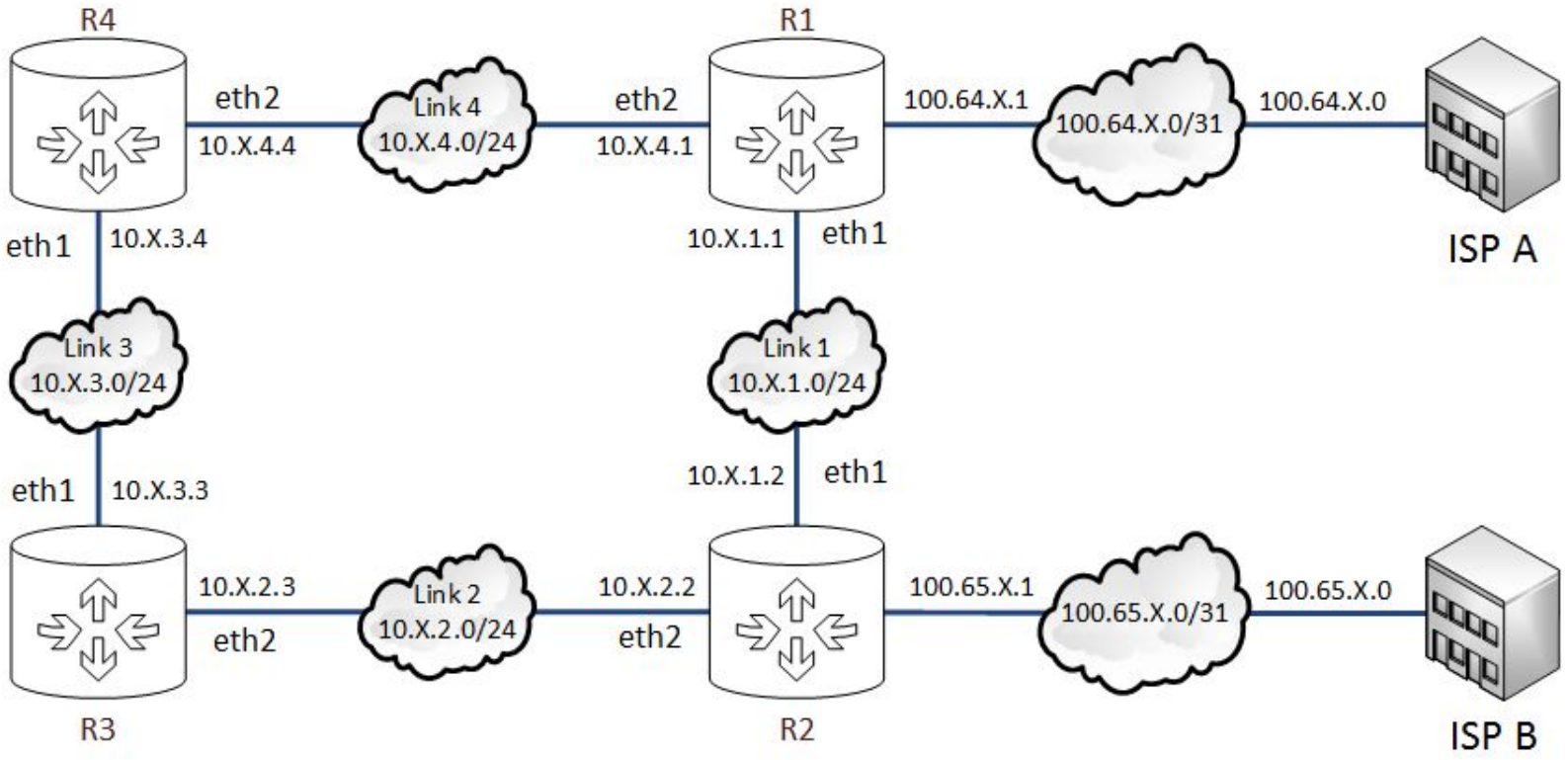
RFC4271: A Border Gateway Protocol 4 (BGP-4) - Rekhter, Li, Hares, 2006

BGP - Building Reliable Networks with the Border Gateway Protocol - I. van Beijnum, 2002

Routing in the Internet - Christian Huitema, 1999

BGP Design and Implementation - Randy Zhang & Micah Bartell, 2003

externes BGP LAB Setup



Lab 3: Externes BGP mit BIRD

Nützliche Kommandos:

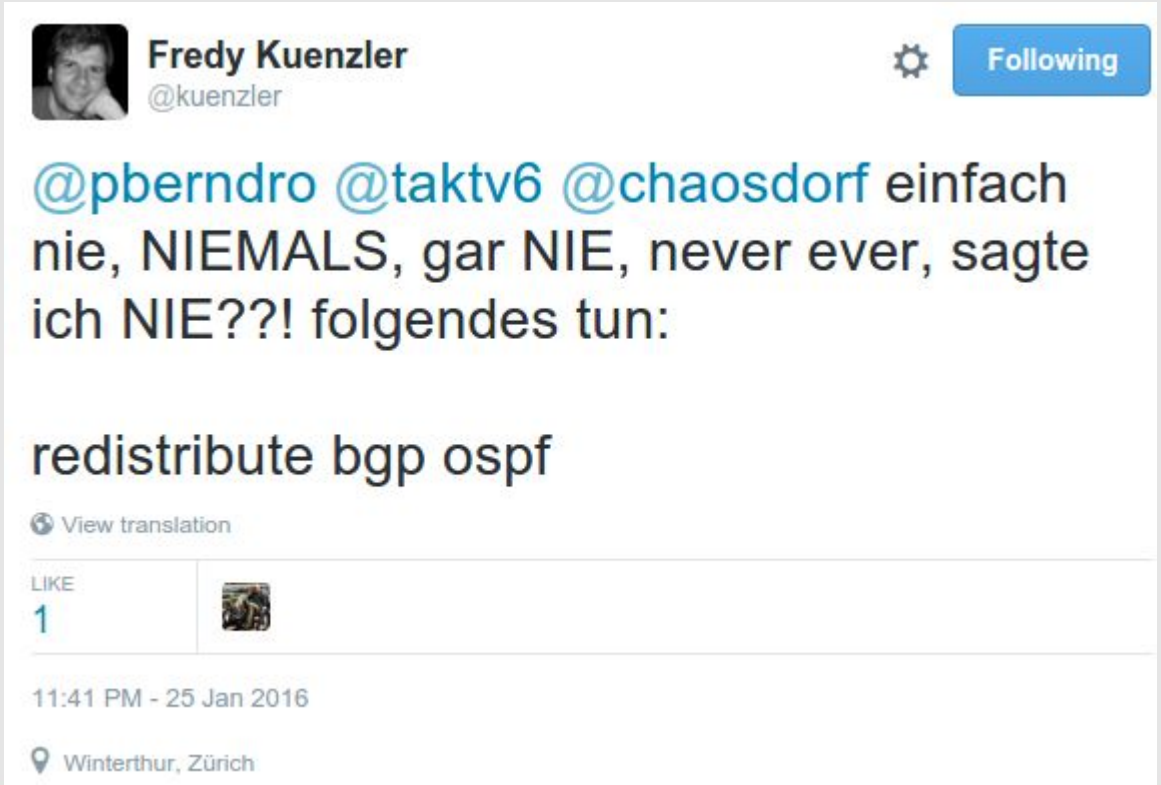
- ➔ birdc - Startet BIRD shell
  - `show protocols`
  - `show route protocol <PROTOCOL>`
  - `show route all protocol <PROTOCOL>`
  - `show route all for <IP>`
  - `show route export <PROTOCOL>`
  - `show protocols all <PROTOCOL>`
- ➔ `ping -I <SRC_IP> -c 4 -i 0.5 <DST_IP>`
- ➔ `traceroute -n -s <SRC_IP> <DST_IP>`
- ➔ `mtr -n -a <SRC_IP> <DST_IP>`
- ➔ `ip link set down dev ethX`
- ➔ `ip link set up dev ethX`

https://goo.gl/erh4FS

Pause

Session 4: Verbotene Dinge

BGP in OSPF redistribution



Disclaimer

- Ja, Fredy hat recht
- Ja, das sollte man gut überlegen
- Und nur tun, wenn man weiss was man tut
- Dann kann man sich aber ein zweites Routingprotokoll sparen :)
- Niemals, unter keinen Umständen, eine Full Table ins IGP übernemhen!
- BGP Attribute gehen verloren (AS_PATH, MED, local_pref, …)

Exkurs: OSPF External Routes

- OSPF External Type 1 (E1)
  - Pfadkosten werden wie gewohnt kalkuliert
  - OSPF-Kosten + Kosten des externen Links
- OSPF External Type 2 (E2)
  - Pfadkosten bestehen NUR aus den Kosten des externen Links
  - Kosten alles interne OSPF-Links werden NICHT addiert
- E1 Routen werden vor E2 Routen bevorzugt
- Default ist E2

Originate default

- Kommt per eBGP nur eine oder mehrere Defaultrouten, kann man die recht entspannt ins IGP weitergeben
- Kein internes BGP oder anderer interner Verteilungsmechanismus für eine Route nötig
- Vereinfacht internes Netzwerk
- Für “nur default-Route weitergeben” existiert z.B. bei Cisco separater Befehl

redistribute bgp -> ospf

- Kein iBGP full Mesh erforderlich
- Für eine kleine Zahl Routen abseits der Default-Route filtern!
- Import-Filter für OSPF bauen!!

HANDLE. WITH. CARE.

LAB: Redistribute BGP -> OSPF

http://www.tea.systems/workbooks/

→ FFRLRDBGPOSPFRedistLAB.pdf

Pause

Session 5: BGP (intern)

Agenda

- ➔ Verwendung
- ➔ Grundlagen
- ➔ iBGP Full Mesh
- ➔ Was passiert ohne Full Mesh?
- ➔ Routenauswahl
- ➔ Route Reflection
- ➔ Confederations

IBGP: Verwendung

Internal Border Gateway Protocol
=> Dient der Verbreitung von eBGP Routen innerhalb eines Netzes

Einsatzbereiche:

- ➔ Enterprise
- ➔ Datacenter
- ➔ Internet Service Provider

IBGP: Grundlagen

- ➔ Sessions werden zwischen Loopback Adressen konfiguriert
  - ◆ Session bleibt erhalten so lange Zwischen Router A und B ein Weg existiert
  - ◆ Setzt funktionierendes IGP (OSPF) voraus!
- ➔ Eine per IBGP empfangene Route wird grundsätzlich nicht per IBGP weitergereicht!
  - ◆ IBGP Full Mesh erforderlich! (Gleich dazu mehr)
- ➔ Es wird kein Attribut verändert
  - ◆ Daher keine Loop Erkennung möglich (vgl. AS Path bei EBGP)

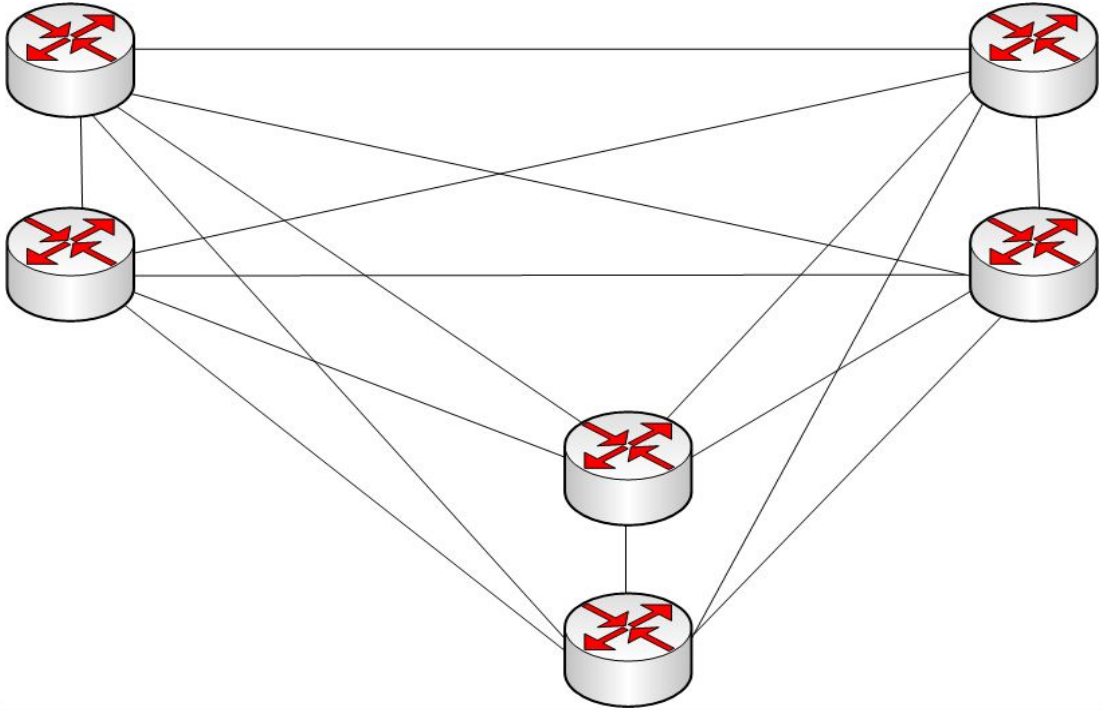
IBGP: Full Mesh

Grundregel: Jeder Router in einem AS muss mit jedem anderen Router im selben AS eine IBGP Session aufbauen!

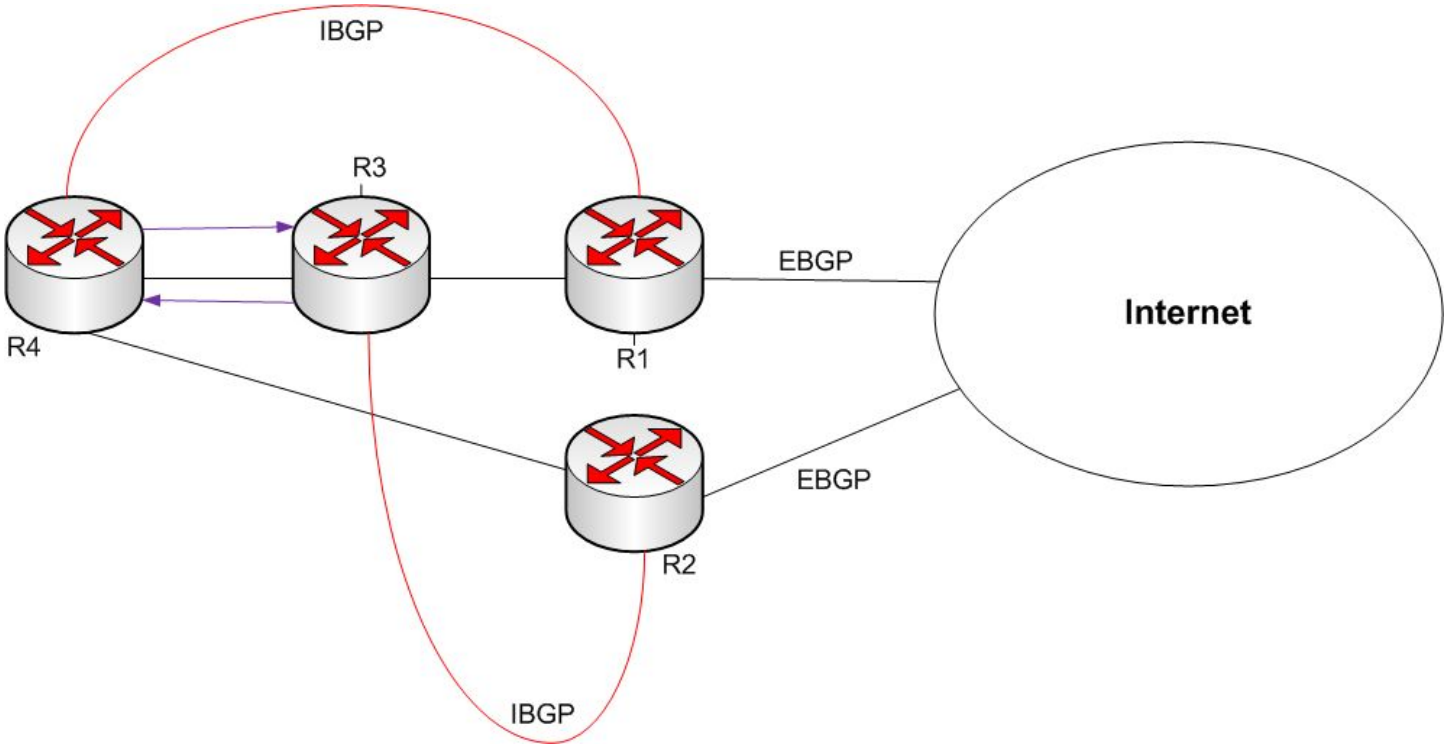
- ➔ N Router = N * (N - 1) BGP Sessions! O(n²)
- ➔ Skaliert nicht!
- ➔ Keine Regel ohne Ausnahme: Route Reflection (spaeter mehr)

IBGP: Full Mesh Beispiel

6 Router: 6 * (6 - 1) / 2 = 15 IBGP Sessions



BGP: Was passiert ohne Full Mesh?



BGP: Routenauswahl (vollständig)

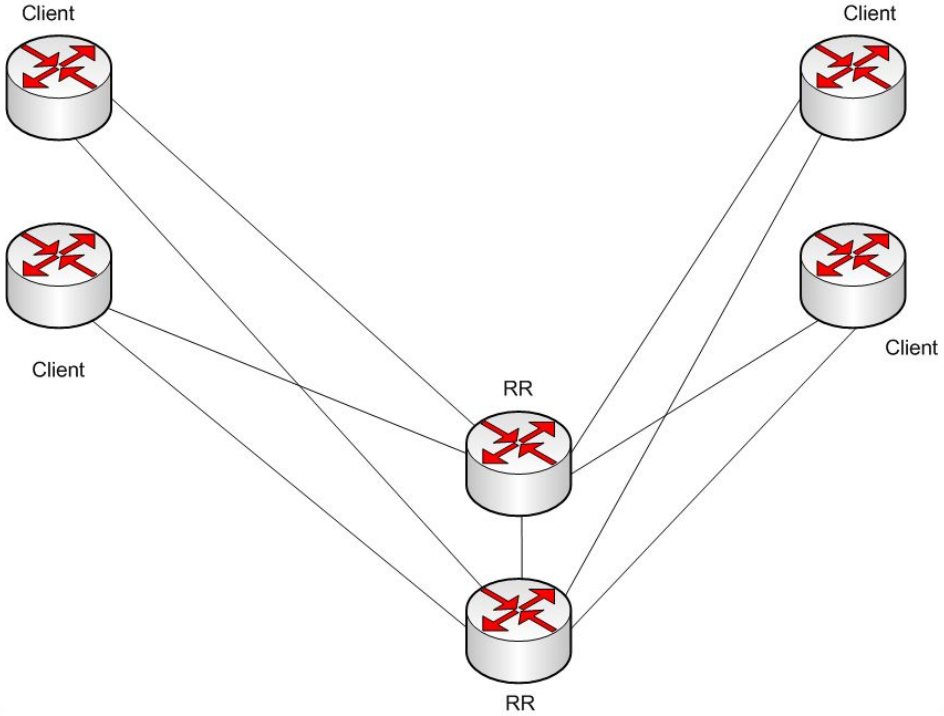
1. Next Hop erreichbar?
2. > LOCAL_PREF
3. < AS_PATH
4. < ORIGIN
5. Falls mehrere Routen mit selbem Nachbarn: < MED
6. EBGP > IBGP
7. < IGP (OSPF) Metric
8. < Cluster List (Nur bei Route Reflection)
9. < Router ID
10. < Nachbar IP Adresse

IBGP: Route Reflection

- ➔ Route Reflection ermöglicht die Full Mesh Regel auszusetzen
- ➔ Route Reflector reicht per IBGP empfangene Route per IBGP weiter
- ➔ Routing Loop wird über Cluster List verhindert (vgl. AS Path bei EBGP)
- ➔ Spezielle Konfiguration nur auf Route Reflector, nicht auf Client

BGP: Route Reflection Beispiel

4 + 2 Router: 2 (RR) * 4 (Client) + 1 = 9 IBGP Sessions



IBGP: Confederations

- ➔ Unterteile ein AS in mehrere Unter ASe
- ➔ Zwischen Unter ASen gelten EBGP Regeln (AS Path!)
- ➔ Innerhalb jedes Sub AS gelten die allgemeinen IBGP regeln
- ➔ Historisch: Probleme mit erweiterten BGP Anwendungen (L2VPN, L3VPN, VPLS)
  - ◆ Keine große Verbreitung in der Praxis

Lab 4: Internes BGP

Pause

Session 6: Policy Based Routing

Agenda

- ➔ Verwendung
- ➔ Möglichkeiten
- ➔ Fallstricke
- ➔ Konfiguration
  - ◆ Linux
  - ◆ Bird

Policy Based Routing

- Wir hatten gelernt: Routing-Entscheidung anhand von **Ziel**-IP
- Das stimmt grundsätzlich auch weiterhin :-)
- Reicht aber nicht immer aus
- Szenarien: Pakete/Traffic
  - von $prefix
  - über $interface kommend/abgehend
  - mit $Eigenschaft (think: per netfilter feststellbar)

  sollen einen anderen Weg gehen.

- Übliche Lösung: Mehrere Routing Tabellen + Policy

Policy Based Routing

- Praktisch: mehrere Routingtabellen (RIB/FIB)
- Policy entscheidet welche Tabelle genutzt wird
- Wenn Tabelle feststeht, reguläre Routingentscheidung
- Wenn keine passende Route gefunden wurde, wird Policy weiter abgearbeitet!
- Gängige Technik
- Wird von gängigen Routern sowie Linux unterstützt
- Linux unterstützt theoretisch 2^32 Tabellen (Linux <= v2.4 255)

Routing Tabellen unter Linux

```
max@pandora:~$ ip route help
Usage: ip route { list | flush } SELECTOR
       ip route { add | del | change | append | replace } ROUTE
[...]
       ip route get ADDRESS [ from ADDRESS iif STRING ]
                            [ oif STRING ] [ tos TOS ]
                            [ mark NUMBER ]
[...]
SELECTOR := [ root PREFIX ] [ match PREFIX ] [ exact PREFIX ]
            [ table TABLE_ID ] [ proto RTPROTO ] [...]
[...]
ROUTE := NODE_SPEC [ INFO_SPEC ]
NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ]
             [ table TABLE_ID ] [ proto RTPROTO ]
[...]
TABLE_ID := [ local | main | default | all | NUMBER ]
```

Policy Based Routing unter Linux

- Anzeige und Konfiguration der Routing-Policy-DB mit **`ip rule`**

```
max@pandora:~$ ip rule help
Usage: ip rule [ list | add | del | flush | save ] SELECTOR ACTION
       ip rule restore
SELECTOR := [ not ] [ from PREFIX ] [ to PREFIX ] [ tos TOS ] [ fwmark FWMARK[/MASK] ]
            [ iif STRING ] [ oif STRING ] [ pref NUMBER ]
ACTION := [ table TABLE_ID ]
          [ realms [SRCREALM/]DSTREALM ]
          [ goto NUMBER ]
          SUPPRESSOR
SUPPRESSOR := [ suppress_prefixlength NUMBER ]
              [ suppress_ifgroup DEVGROUP ]
TABLE_ID := [ local | main | default | NUMBER ]
```

Policy Based Routing unter Linux

- Default policy

```
max@pandora:~$ ip rule show
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

- Zahl am Zeilenanfang Priorität der Regel (kleiner = höher)
- Abarbeitung **aller Regeln** in absteigender Priorität bis Route-Match!
- Tabelle **local** enthält alle lokalen Prefixe und Broadcastadressen
- Tabelle **main** ist die “normale” Routing-Tabelle
- Handle with care! Policy kaputt -> Routing subtil kaputt

Policy Based Routing unter Linux - Fallstricke

```
max@pandora:~$ ip rule show
0:        from all lookup local
32766:    from all lookup main
32767:    from all lookup default

max@pandora:~$ ip route show table main
192.168.20.0/24 dev wlan0  proto kernel  scope link  src 192.168.20.42

max@pandora:~$ ip route show table default
default via 192.168.20.1 dev wlan0
```

- Augenscheinlich keine Default-Route (in Haupttabelle)
- Es greift aber die Default-Route in der danach ausgewerteten Tabelle!

Policy Based Routing unter Linux - Fallstricke

- Warum ist das gefährlich?

```
cr01:~# ip rule show
0:        from all lookup local
32764:    from all iif vlan1001 lookup 42
32764:    from 10.132.0.0/16 lookup 42
32766:    from all lookup main
32767:    from all lookup default

cr01:~# ip route show table 42 | head -n1
default via 100.64.0.84 dev gre_ffrl_dus_a proto bird
cr01:~# ip route show | head -n1
default via <lokales gw> dev eth0
```

- Fällt die Default-Route per BGP weg, geht sämtlicher Traffic lokal raus...

Doppelter Boden (zu Fuss)

- Statische Default-NullRoute mit hoher Metric
- Greift nur, wenn keine reguläre Default-Route vorhanden
- Generiert “Network unreachable” Antworten, wenn sie greift
- Falls gewünscht geht auch Blackholing (ist aber nicht schön)

```
ip route add unreachable 0.0.0.0/0 metric 200 table 42
```

Mehrere Routingtabellen in BIRD

- Bird kann intern mehrere Tabellen verwalten
- Jedes Protokoll kann einer Tabelle zugeordnet werden (Default: “master”)
- Jede Bird Tabelle kann mit einer Linux-Kernel-Tabelle synchronisiert werden
- Protocol Pipe erlaubt Routen-Austausch zwischen Bird-Tabellen

```
table freifunk;      # Tabelle erstellen (global)

protocol static {
    table freifunk;  # Tabelle “freifunk” benutzen
    route 0.0.0.0/0 unreachable;
}
```

Mehrere Routingtabellen in BIRD - Befehle

```
# Routen anzeigen
show route [all] [primary] [table <name>]
show route for <prefix> [all] [table <name>]

cr01:~# birdc show protocols
name         proto    table    state  since       info
device1      Device   master   up     2015-11-16
kernel1      Kernel   master   up     2015-11-16
kernel2      Kernel   freifunk up     2015-11-16
local_ffpb   Static   freifunk up     2015-11-16
p_ff_main    Pipe     master   up     2015-11-16  => freifunk
```

Mehrere Routingtabellen in BIRD - Pipes

- Synchronisation zwischen Tabellen
- Beispiel
  - OSPF / BGP / whatever füllt Tabelle “freifunk”
  - Tabelle “freifunk” wird in Kernel table 42 sychronisiert
  - Policy-Routing schickt sämtlichen Freifunk-Traffic in Tabelle 42
- Debugging auf Maschine fast unmöglich, da Routen nicht in main-Table
  - Linux Tools kann man keine Routing-Tabellen angeben :-(
- Lösung: Alle Routen bis auf Default-Route in main-Table synchronisieren
  - In Bird “freifunk” in “master” table pipe’n mit entsprechendem Filter
  - “master” Table wird üblicherweise eh mit Kernel table “main” synchronisiert
  - => Debugging leicht möglich :-)

Mehrere Routingtabellen in BIRD - Pipe Beispiel

```
# Sync (import) anything but the default route from freifunk to master
table.
# Sync (export) nothing from main kernel table to freifunk.
protocol pipe p_ff_main {
    table master;
    peer table freifunk;

    import filter {
        if is_default () then
           reject;
        accept;
    };
    export none;
};
```

Doppelter Boden mit Bird

- Bird erlaubt bei statischen Routen leider keine Metric :-(
- Workaround: Eigenes Protokoll mit niedriger Preference

```
protocol static doppelter_boden {
    preference 1;

    route 0.0.0.0/0 unreachable;
}
```

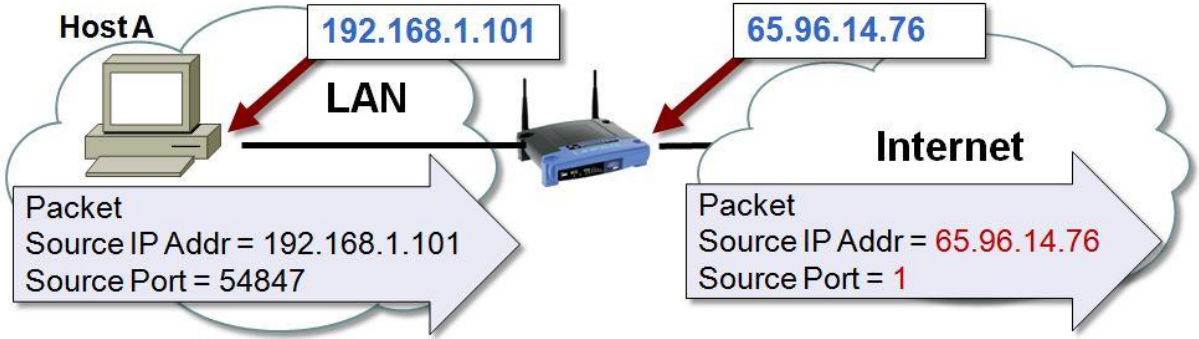
Further Reading

- man ip-route
- man ip-rule
- http://bird.network.cz/?get_doc&f=bird-2.html

Lab 5 - Policy Based Routing

Session 7: NATs are good

- Network Adress (Port) Translation



NAT Translation Table

	Local IP Address	Source Port #		Internet IP Address	Source Port #
process X, Host A →	192.168.1.101	54,847	=	65.96.14.76	1
Host B →	192.168.1.103	24,123	=	65.96.14.76	2
process Y, Host A →	192.168.1.101	42,156	=	65.96.14.76	3
Host C →	192.168.1.102	33,543	=	65.96.14.76	4

Quelle: microchip.wikidot.com

IPv6 is unNATural

- NAT für IPv6 mittlerweile auch definiert
- Genug IP-Space für alle(s) da!
- **Don’t do NAT for IPv6!**

Generelle Literaturempfehlung