Robert Sander <r.sander@heinlein-support.de>

Multi-Homing mit Redundanz

oder

Wie funktioniert das Internet eigentlich?

Wer sind wir?

- wir bieten seit über 20 Jahren Wissen und Erfahrung rund um Linux-Server und E-Mails
- IT-Consulting und 24/7 Linux-Support
- Eigener Betrieb eines ISPs seit 1992
- mailbox.org seit 2 Jahren

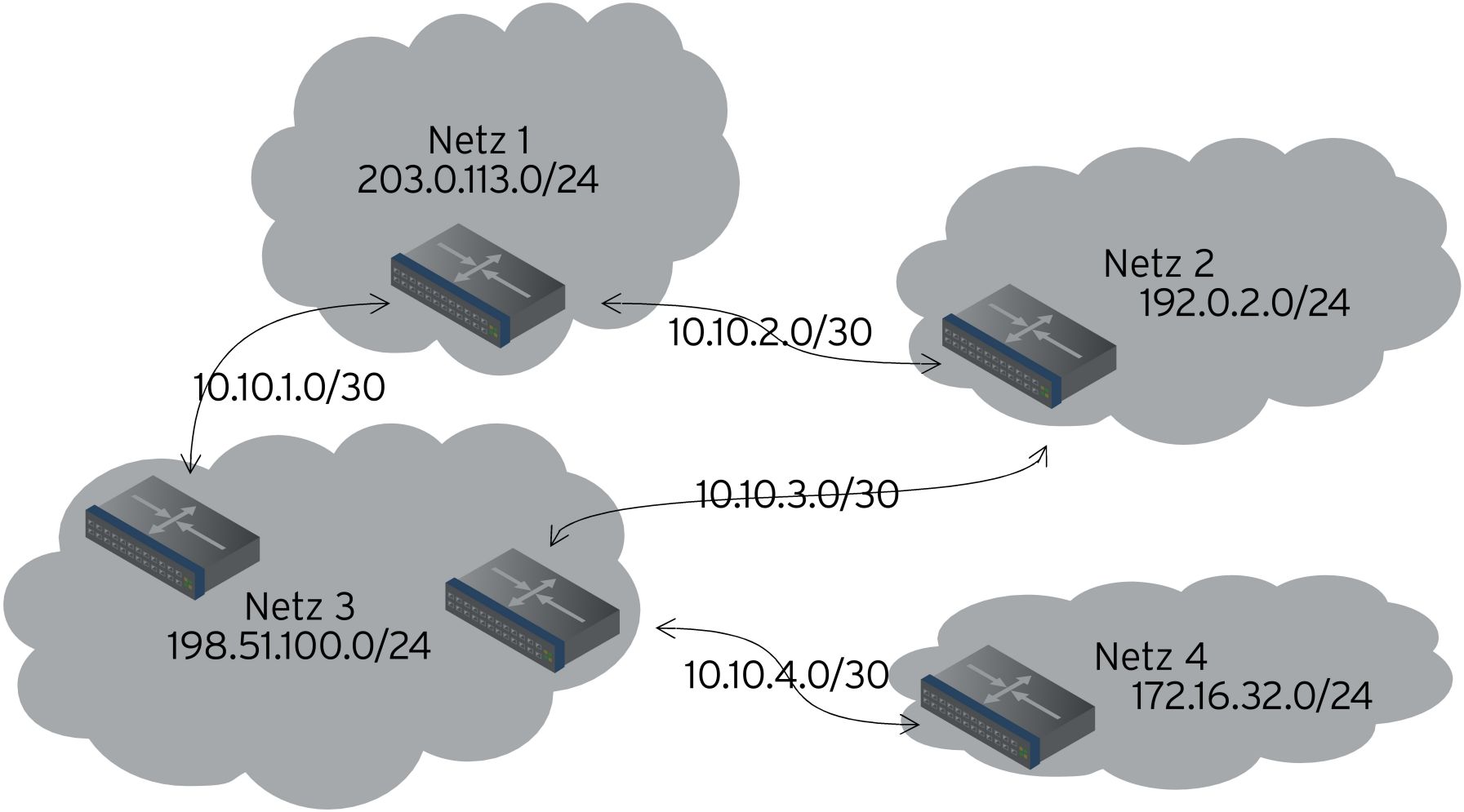
Inhalt

- Wie funktioniert das Internet?
- Wie funktioniert ein AS?
- Wie funktioniert das mit Linux?
- Wie funktioniert die Redundanz?

Robert Sander <r.sander@heinlein-support.de>

Teil 1:
Wie funktioniert das Internet?

Netz aus Netzen



Router

- Netzwerkgerät mit mindestens 2 Schnittstellen
  - intern (LAN)
  - extern (WAN)
  - Verbindungen zu anderen Routern
- Router-ID
  - IPv4-Adresse
  - Auf Loopback-Interface
- Interface-Adressen

Autonomous System

- Sammlung von gemeinsam verwalteter IP-Netze
- „Autonome Systeme sind untereinander verbunden und bilden so das Internet.“
- Eindeutig identifiziert über AS-Nummer (32 bit)
- In Europa: RIPE http://ripe.net/
- Voraussetzung für eigene AS-Nummer
  - Verbindung zu zwei anderen AS über BGP

RIPE NCC
Réseaux IP Européens Network Coord. Centre

- Objekte in der RIPE-Datenbank

- aut-num: AS-Nummer
  - Mit Import und Export (BGP-Partner)
- inetnum / inet6num: zugeteilter IP-Addressbereich
  - Wem gehören die IP-Adressen?
- route / route6:
  - Welches AS darf den IP-Prefix annoncieren?
  - IP-Prefix ggfs kleiner als inetnum / inet6num
- domain:
  - Welche Nameserver sind für die PTR-Records zuständig?

Robert Sander <r.sander@heinlein-support.de>

Border Gateway Protocol

- Exterior Gateway Protocol / Routing Protokoll
- Router tauschen IP-Routen untereinander aus
- Router A sagt Router B
  - Du erreichst a.b.c.0/24 über mich
- Router B sagt Router A
  - Du erreichst y.z.0.0/16 und u.v.w.0/24 über mich
- Wird die Defaultroute übergeben, dann Provider → Kunde
- Empfänger kann entscheiden, ob er einzelne Routen akzeptiert

- IPv4 + IPv6

Robert Sander <r.sander@heinlein-support.de>

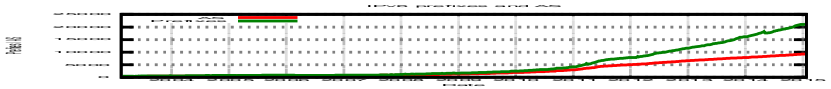
BGP Updates

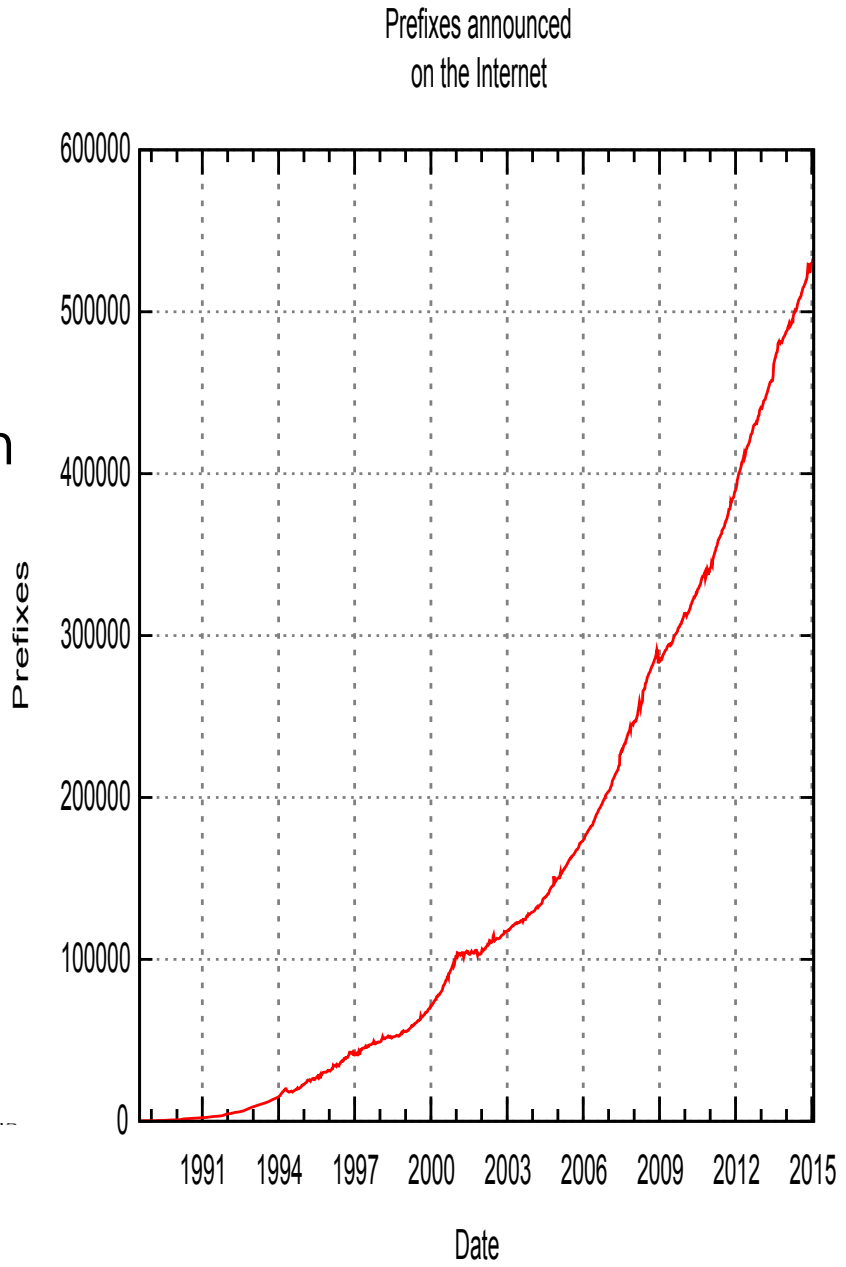
- Announcement oder Withdrawal von Routen
- AS Path
  - über welche AS ist der IP-Prefix erreichbar
- IGP-Metrik
  - wie teuer ist es bei mir dorthin
- Multi-Exit Discriminator (MED)
  - Priorität bei parallelen Peerings zwischen zwei AS
- Local Preference
  - bevorzugt AS-Pfade mit höherem Wert
- Next Hop
  - IP-Adresse des Next-Hop-Routers zum annoncierten Prefix

BGP Pfadauswahl

- Anhand der verschiedenen Prioritäten
- Zusätzlich durch Filter beeinflussbar

- Full Routing Table
  - IPv4 ~600.000 Einträge
  - IPv6 ~27.000 Einträge
  - http://www.cidr-report.org/





Teil 2:
Wie funktioniert ein AS?

Internes Routing

- Interior Gateway Protocol
  - iBGP
  - OSPF
  - IS-IS
- Statisches Routing
  - ip route add 203.0.113.0/24 via 10.10.1.2

Open Shortest Path First
OSPF

- Grundlage: Dijkstra-Algorithmus
- Graphentheorie
- Zwischen Knoten liegen Kanten, die Kosten haben.
- Je kleiner die Kosten, desto kürzer der Weg (Shortest Path)
- Für verschiedene Wege unterschiedliche Kosten
  - „Billigster“ Weg wird ausgewählt

OSPF

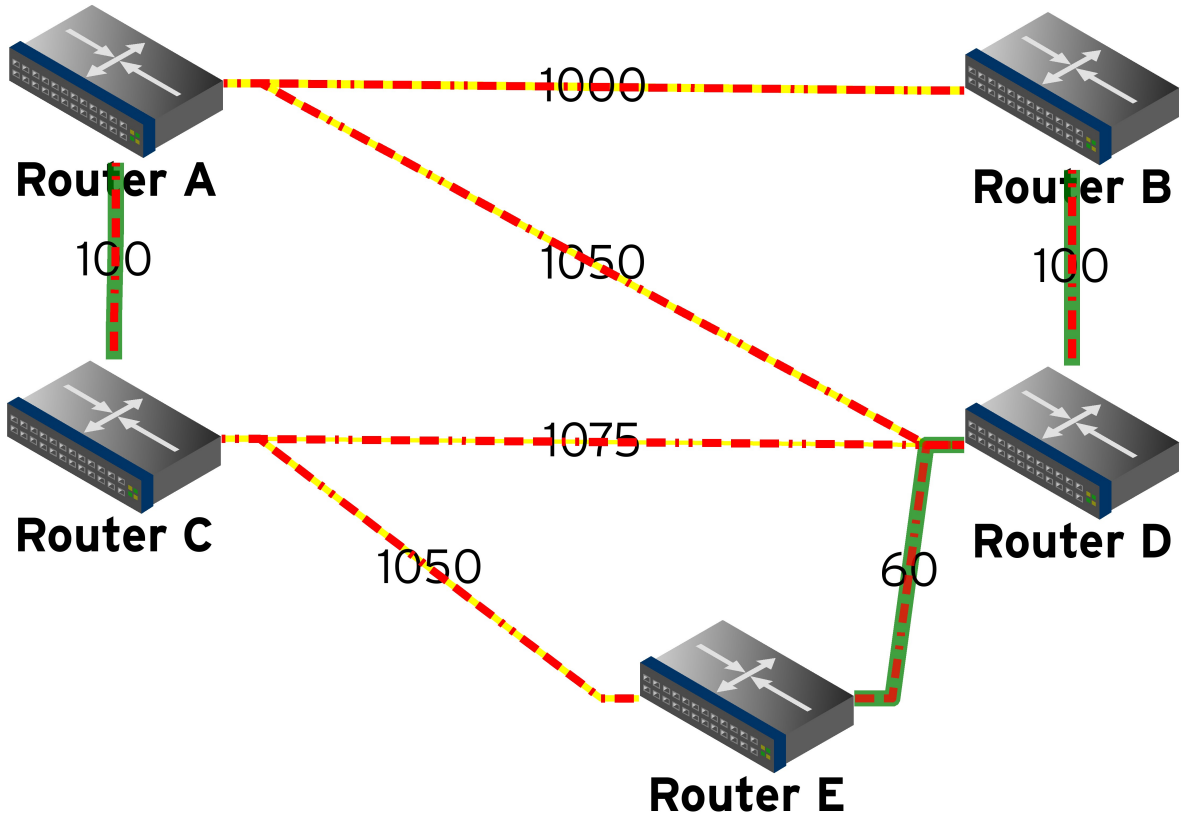
- Keine Schleifen im Routing
- Überwachung der Nachbarn durch Hello-Protokoll
  - Wer ist mein Nachbar? Keepalive + Wahl von DR und BDR
- Hierarchische Struktur mit Areas
  - Immer eine Area 0 (Backbone)
  - Andere Areas über einen Router mit der Area 0 verbunden
    - Area Border Router (Designated und Backup)
    - Kann Routen zusammenfassen und LSAs filtern
- Link State Advertisement (LSA)
  - verschiedene Typen (Router, Network, Summary, AS-External)
- Designated Router in Broadcast-Netzen

OSPF

- Schnelle Propagierung von neuen Routen
- Dead-Time erkennt tote Nachbarn
  - Routen werden auf den „zweitbilligsten“ Link umgelegt
- Ausfallsicheres Routing möglich
  - Aber immer mit Dead-Time Unterbrechung

- IPv4 + IPv6
  - OSPFv2
  - OSPFv3

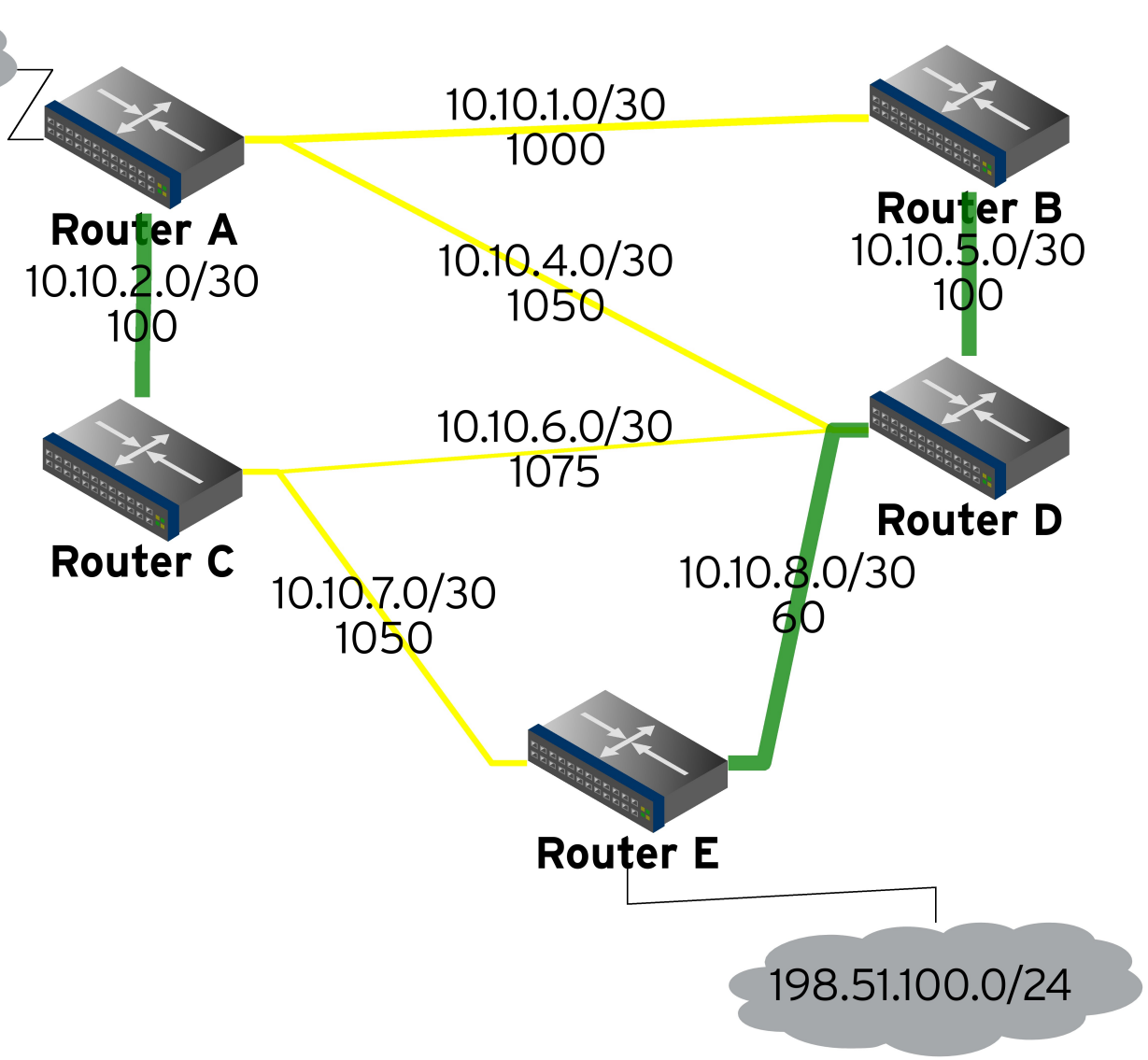
OSPF Kostenbeispiel



- A - B - D: 1100
- D - C - A: 1175
- E - C - A: 1150
- A - D - E: 1110
- A - B - D - E: 1160

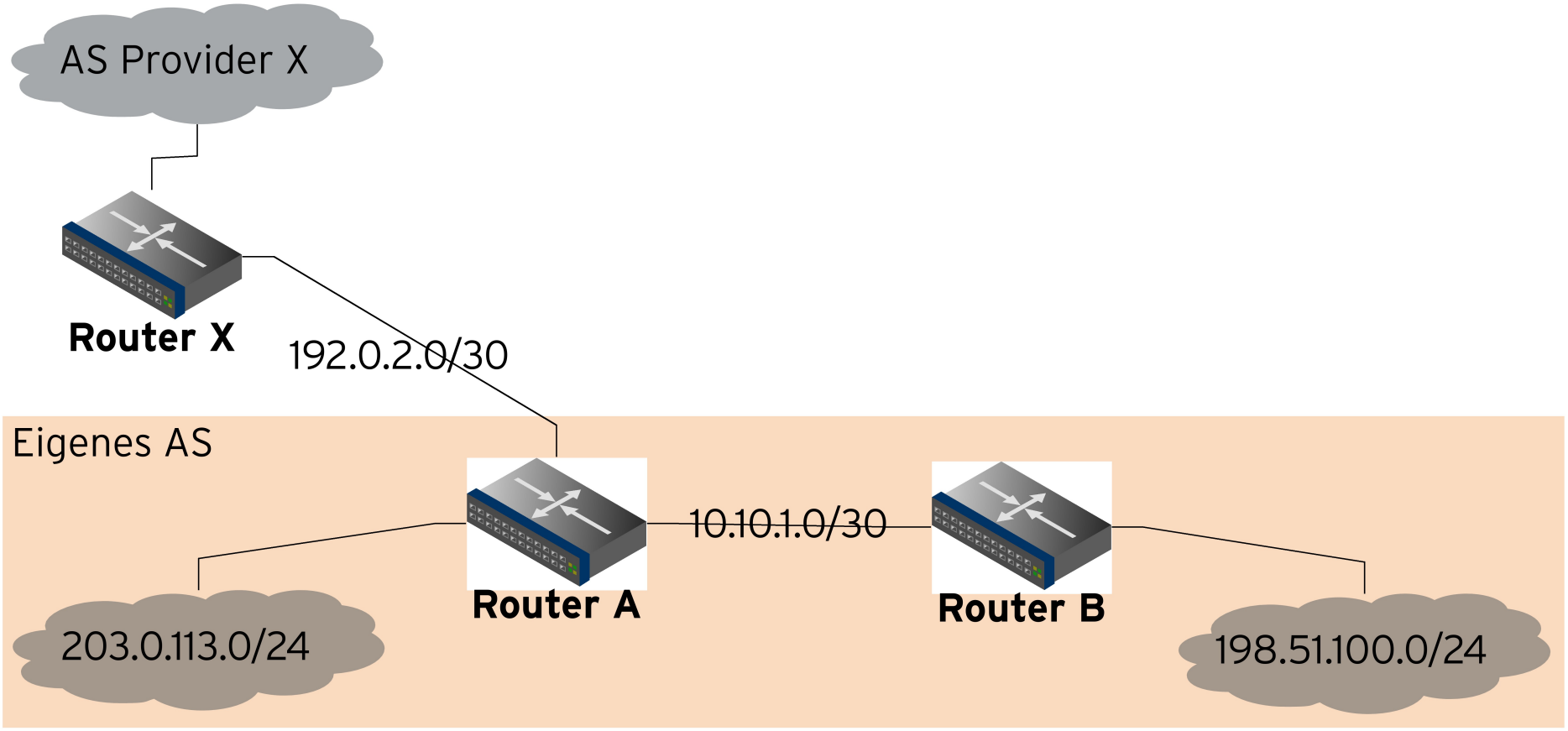
OSPF

- Router A
  - 203.0.113.0/24 dev X
  - 198.51.100.0/24 via 10.10.4.2
- Router E
  - 203.0.113.0/24 via 10.10.8.1
  - 198.51.100.0/24 dev X
- Router C
  - 203.0.113.0/24 via 10.10.2.1
  - 198.51.100.0/24 via 10.10.7.2
- Router D
  - 203.0.113.0/24 via 10.10.4.1
  - 198.51.100.0/24 via 10.10.8.2
- Router B
  - 203.0.113.0/24 via 10.10.1.1
  - 198.51.100.0/24 via 10.10.5.2



Teil 3:
Wie funktioniert das mit Linux?

Beispielsetup



Packet Forwarding

- echo 1 > /proc/sys/net/ipv4/ip_forward
- echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
- oder in /etc/sysctl.conf
  - net.ipv4.ip_forward=1
  - net.ipv6.conf.all.forwarding=1
- Redirects ausschalten
  - net.ipv4.conf.all.accept_redirects = 0
  - net.ipv6.conf.all.accept_redirects = 0
- Keine Redirects senden
  - net.ipv4.conf.all.send_redirects = 0
  - ip6tables -A OUTPUT -p ipv6-icmp -m icmp6 --icmpv6-type 137 -j DROP
  - führt ggfs. zu ineffizientem Routing

Routing Software

- Quagga (Fork von Zebra)
  - verschiedene Daemonen für Protokolle BGP, OSPFv2, OSPFv3, RIP, Kernel
  - vtysh ähnlich zu IOS von Cisco
- XORP
  - aktuelles Release von Januar 2012...
- OpenBGPD
  - BGP und OSPF, Teil von OpenBSD
- BIRD
  - Entwickelt in Prag von nic.cz
  - an vielen europäischen IXPs genutzt (DE-CIX, AMS-IX, LINX, ECIX)
  - Linux, FreeBSD, NetBSD und OpenBSD
  - `http://bird.network.cz/`

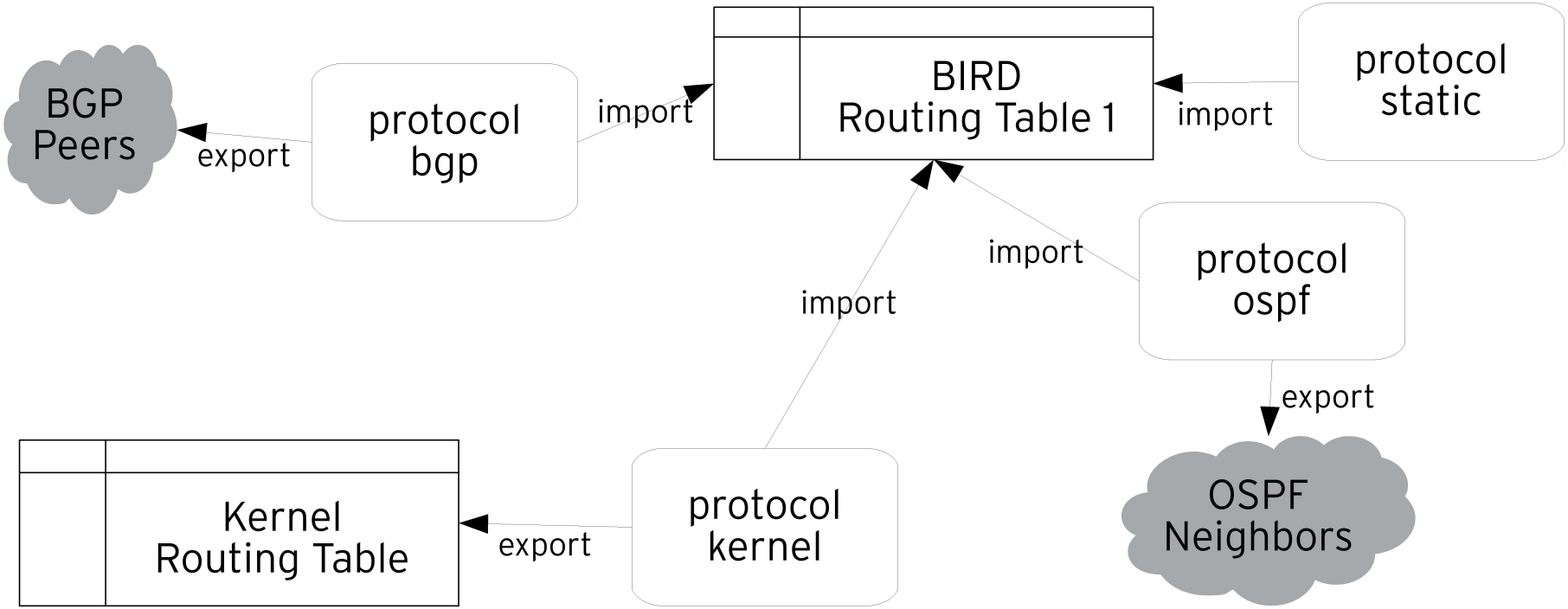
Robert Sander <r.sander@heinlein-support.de>

BIRD
BIRD Internet Routing Daemon

- BGP, OSPFv2, OSPFv3, RIP, BFD, statische Routen, Kernel
- Mehrere Routing-Tabellen
- IPv4 und IPv6 in zwei separaten Prozessen
- Gemeinsame Konfiguration möglich
  - Ausnahme: IP-Adressen...
- Mächtige Filterfunktionen
- birdc / birdc6 als CLI

BIRD
Konzept

- An einer BIRD-internen Routing-Table hängen mehrere Protokolle
- Import und Export Filter



Robert Sander <r.sander@heinlein-support.de>

BIRD
Konzept

- Mehrere Protokolle durch Namen unterscheidbar
  - BGP PeerX, PeerY, PeerZ...
  - OSPF Area0, AreaN, AreaM...
  - Kernel RT1, RT2, RT3...
- Preference pro Protokoll einstellbar
  - Höchste Preference ergibt aktive Route
  - Plus protokollspezifische Prioritäten
- Mehrere Routing-Tables möglich
  - ähnlich wie im Kernel auch
- Routing-Tables durch „Pipe“-Protokoll verbunden
  - Import/Export-Filter

Robert Sander <r.sander@heinlein-support.de>

BIRD
Filter

- Einfache Programmiersprache
  - benannte Filter
  - benannte Funktionen für Filter
  - Variablen
- Filter erhalten eine Route mit Attributen
  - Entscheidung basierend auf Attributen (z.B. IP-Prefix, Quelle der Route, BGP-Pfad)
  - Manipulation von Attributen (z.B. Source-Address)
- Funktionen fassen Code zusammen
  - Parameter
  - lokale Variablen
- Datentypen für IP-Prefixe
  - Prefix-Listen mit Super- oder Subnetzen

BIRD
Prefixe

- [1.0.0.0/8, 2.0.0.0/8+, 0.0.0.0/32-, 4.0.0.0/8{16,24}]
  - Genau 1.0.0.0/8
  - Alle Subnetze von 2.0.0.0/8
    - 2.1.1.0/24
    - 2.2.0.0/16
  - Alle Supernetze von 0.0.0.0/32
    - 0.0.0.0/24
    - 0.0.0.0/7
  - Alle Netze 4.X.Y.Z mit Netzmasken 16 - 24
    - 4.1.2.0/24
    - 4.4.0.0/17
    - 4.8.16.0/20

BIRD
Konfiguration Kernel

- `router id 192.0.2.11;`
-
```
protocol kernel {
    scan time 20;
    device routes off;
    learn on;
    import all;
    export filter {
        krt_prefsrc = 192.0.2.11;
        accept;
    };
};
```

BIRD
Konfiguration Static

- ```
  protocol static static_bgp {
      preference 10;
      route 198.51.100.0/24 unreachable;
      route 203.0.113.0/24 unreachable;
  }
  ```
- Damit die eigenen BGP Prefixe in der BIRD-Routing-Table sind
  ```

BIRD
Konfiguration BGP

```
→ protocol bgp peerX {
    import filter {
      if rt_import_all(25074) then accept;
      reject;
    };
    export filter {
      if rt_export() then accept;
      reject;
    };
    local as 199118;
    neighbor 192.0.2.1 as 25074;
    source address 192.0.2.2;
  }
```

BIRD
Konfiguration BGP

```
function rt_import_all(int asn) {
   if net_martian() || net_local() then return false;
   if bgp_path.first != asn then return false;
   if bgp_path.len > 64 then return false;
   if bgp_next_hop != from then return false;
   return true;
}
```

```
function net_martian() {
   return net ~ [ 172.16.0.0/12+, 192.168.0.0/16+, 10.0.0.0/8+, … ];
}
```

```
function net_local() {
   return net ~ [ 203.0.113.0/24+, 198.51.100.0/24+ ];
}
```

BIRD
Konfiguration BGP

```
function rt_export() {
   if net_bgp() then return true;
   return false;
}
```

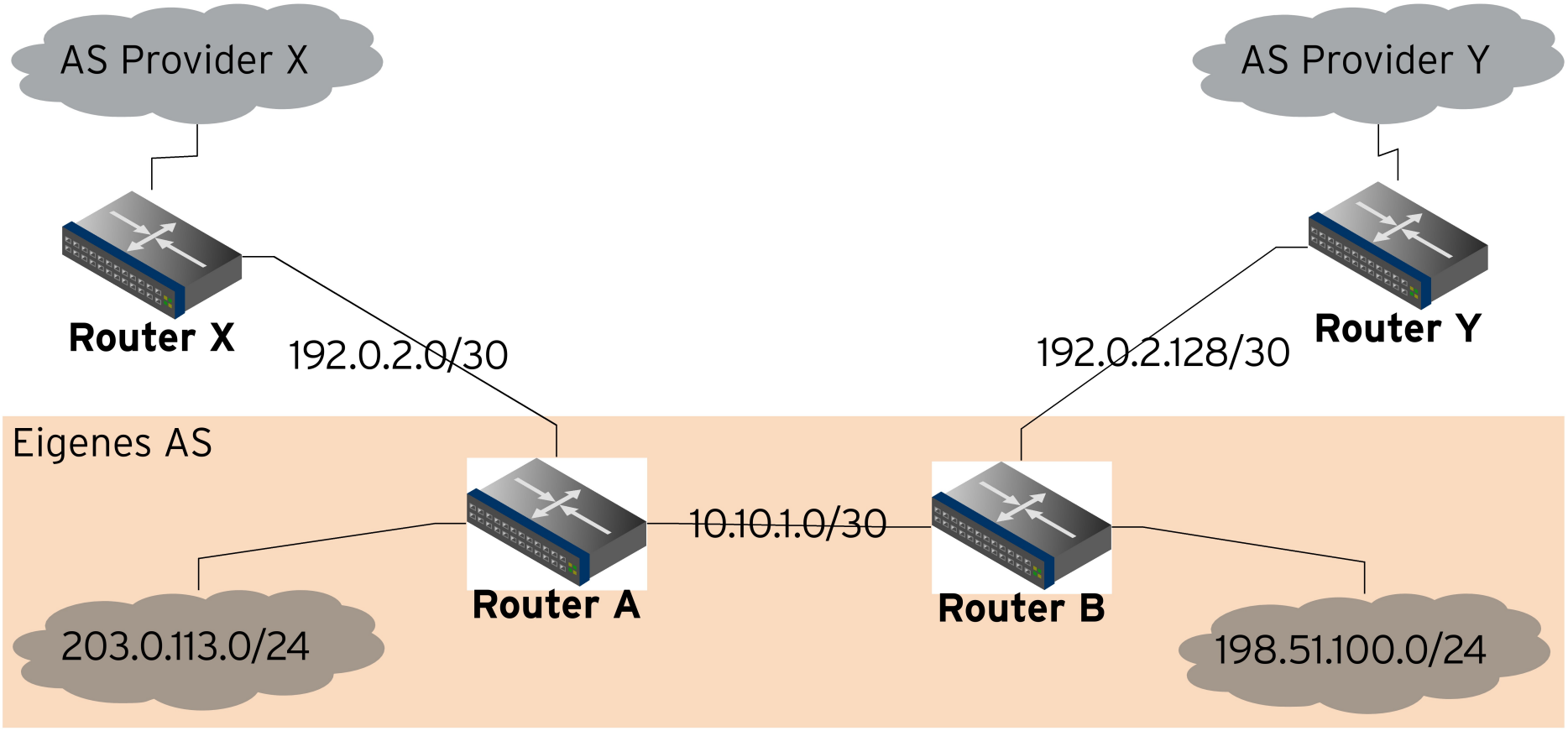
```
function net_bgp() {
   return net ~ [ 198.51.100.0/24, 203.0.113.0/24 ];
}
```

BIRD
Konfiguration OSPF

```
→ protocol ospf {
      import all;
      export filter {
         if source = RTS_BGP then {
            ospf_metric1 = 100;
         }
         accept;
      };
      area 0 {
         interface "eth2" {
            cost 100; hello 10; retransmit 5; dead 45;
         };
      };
  }
```

Teil 3:
Wie funktioniert die Redundanz?

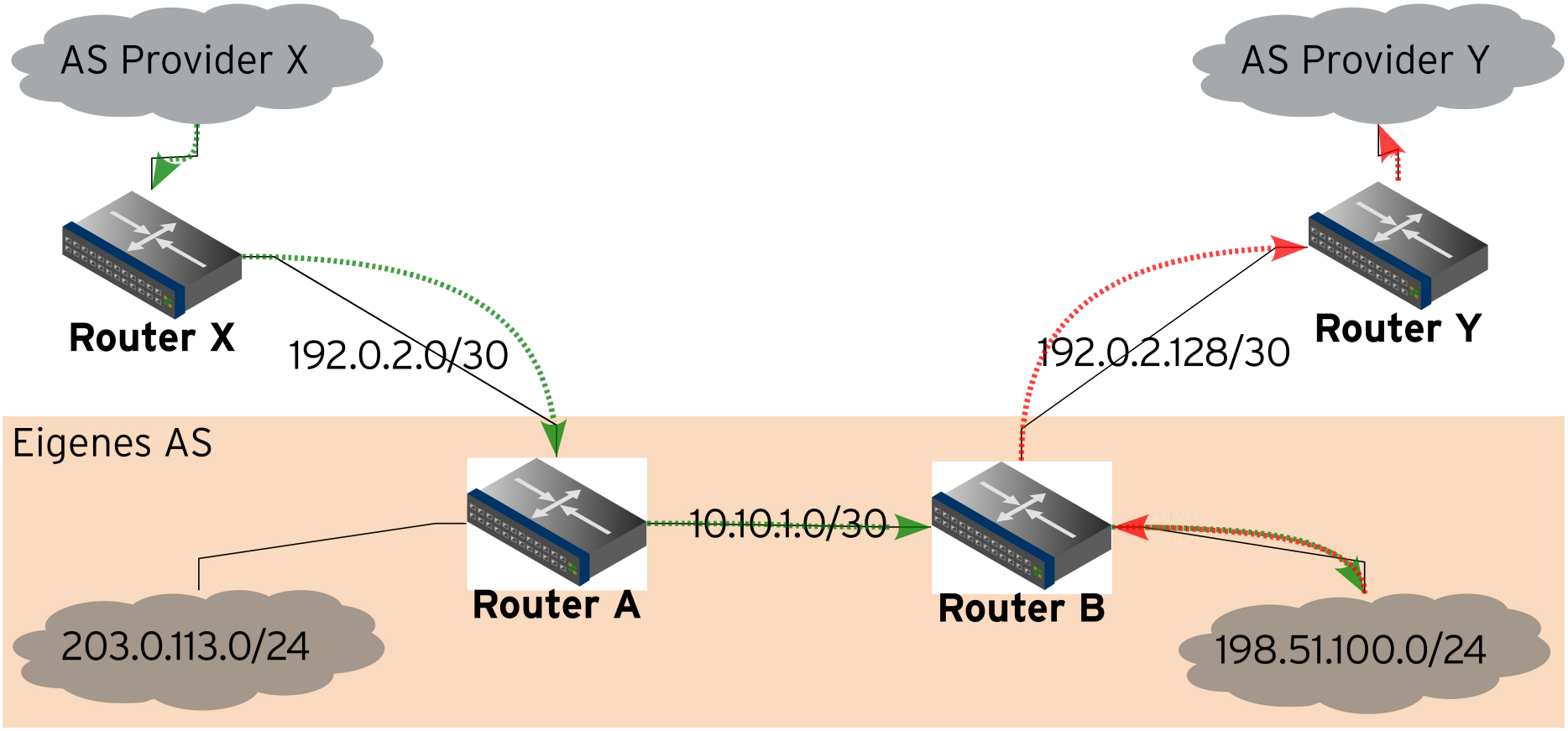
Redundanz durch weiteren Provider



BIRD
Konfiguration BGP

```
→ protocol bgp peerY {
    import filter {
      if rt_import_all(20647) then accept;
      reject;
    };
    export filter {
      if rt_export() then accept;
      reject;
    };
    local as 199118;
    neighbor 192.0.2.129 as 20647;
    source address 192.0.2.130;
  }
```

~~Policy Routing~~ ~~Warum?~~



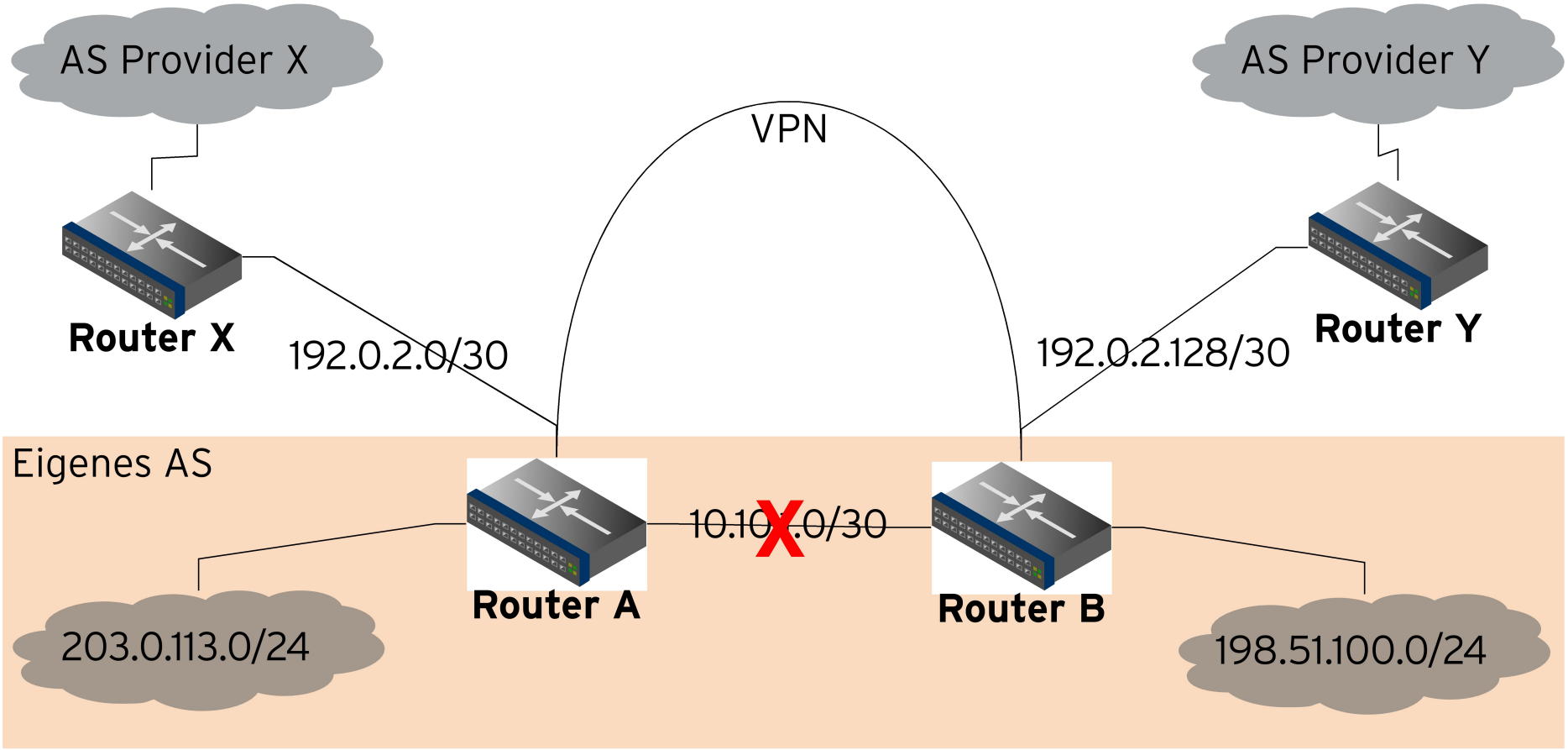
~~Policy Routing~~ ~~Wie?~~

~~Router B~~

- ~~iptables -t mangle -A PREROUTING ! -s 10.10.1.0/30 -i eth2 -j CONNMARK --set-xmark 2~~
- ~~iptables -t mangle -N RESTORE~~
- ~~iptables -t mangle -A PREROUTING -j RESTORE~~
- ~~iptables -t mangle -A RESTORE -i eth2 -j RETURN~~
- ~~iptables -t mangle -A RESTORE -m connmark ! --mark 0x0 -j CONNMARK --restore-mark~~
- ~~ip route add 10.10.1.0/30 src 10.10.1.2 dev eth2 table t2~~
- ~~ip route add default via 10.10.1.1 dev eth2 table t2~~
- ~~ip rule add fwmark 2 lookup t2~~

Robert Sander <r.sander@heinlein-support.de>

Redundanz für Crossconnect



Robert Sander <r.sander@heinlein-support.de>

VPN-Bridge

- Gegen Ausfall des Crossconnects
- OpenVPN bridged mode auf tap-Interface
- Interne VLANs werden durch den Tunnel weitergeleitet
- OSPF-Netze ebenso → kein OSPF-Ausfall
- Traffic läuft verschlüsselt über Provider X und Provider Y
- Crossconnect transparent ersetzt

Bridged OpenVPN Konfiguration

Router A

- `remote 192.0.2.130`
  `local 192.0.2.2`
  `dev tap0`
  `secret secret.key`

- `brctl addbr br0`
- `brctl addif br0 eth0`
- `brctl addif br0 tap0`

Router B

- `remote 192.0.2.2`
  `local 192.0.2.130`
  `dev tap0`
  `secret secret.key`

- `brctl addbr br0`
- `brctl addif br0 eth0`
- `brctl addif br0 tap0`

Keepalived VRRP

- implementiert Virtual Router Redundancy Protocol
- kann z.B. Service-IPs schwenken und Skripte ausführen
- eine VRRP-Instanz hat einen MASTER und mehrere BACKUPs
- MASTER sendet VRRP-Pakete
  - Multicast oder Unicast
  - werden diese nicht mehr empfangen, wird der höchstpriorisierte BACKUP MASTER
- schwenkt innerhalb von Sekundenbruchteil

- Immer aktuelle Version nutzen → aktive Entwicklung

Keepalived VPN-Service

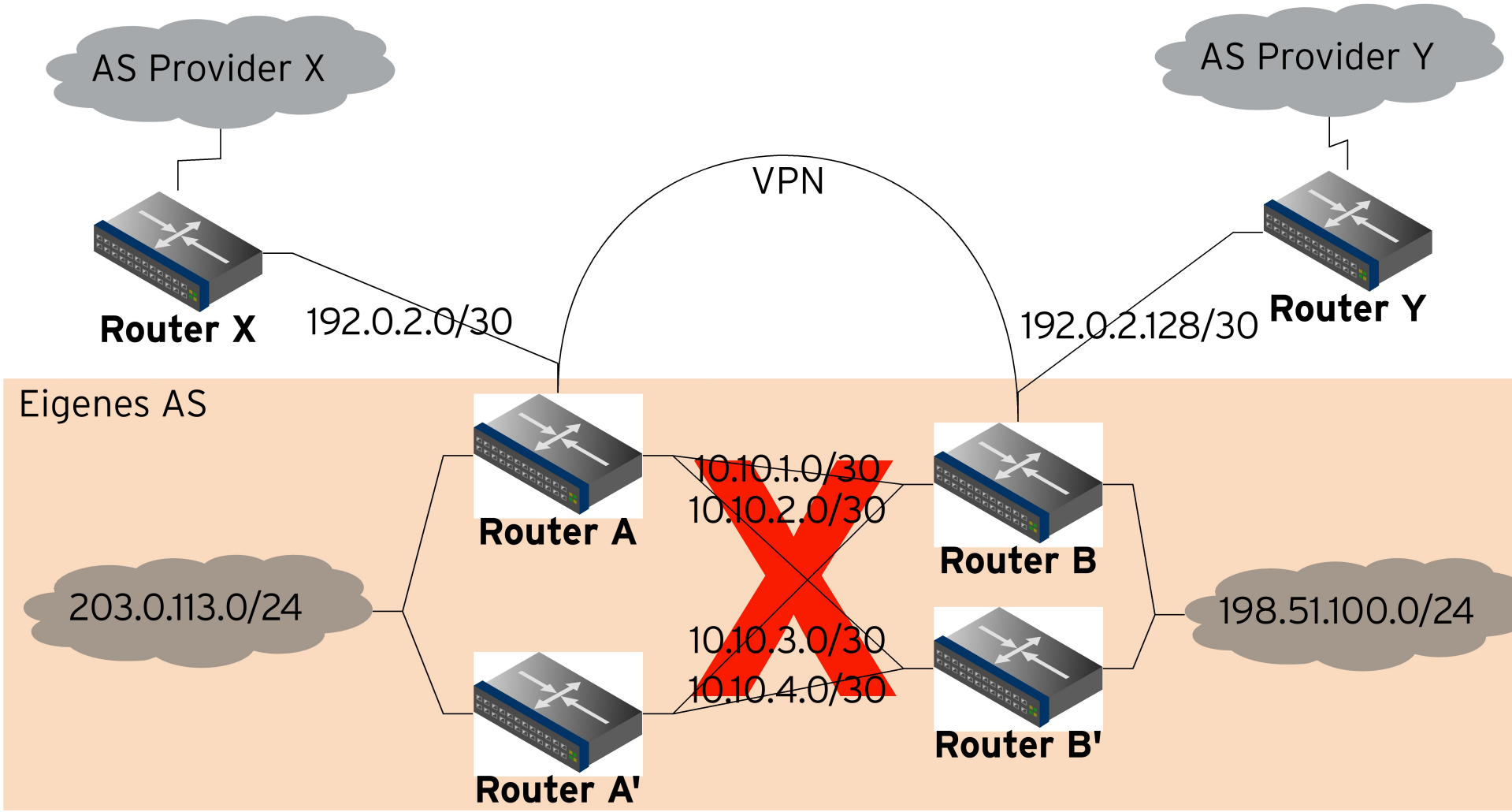
Router A

```
→ vrrp_instance vpnA {
      virtual_router_id 201
      state BACKUP
      priority 50
      interface eth2
      notify_master …
      notify_backup …
  }
→ vrrp_instance vpnB {
      virtual_router_id 202
      state MASTER
      priority 100
      interface eth2
  }
```

Router B

```
→ vrrp_instance vpnA {
      virtual_router_id 201
      state MASTER
      priority 100
      interface eth2
  }
→ vrrp_instance vpnB {
      virtual_router_id 202
      state BACKUP
      priority 50
      interface eth2
      notify_master …
      notify_backup …
  }
```

Beispielsetup
Redundanz für Router



Keepalived

Router A

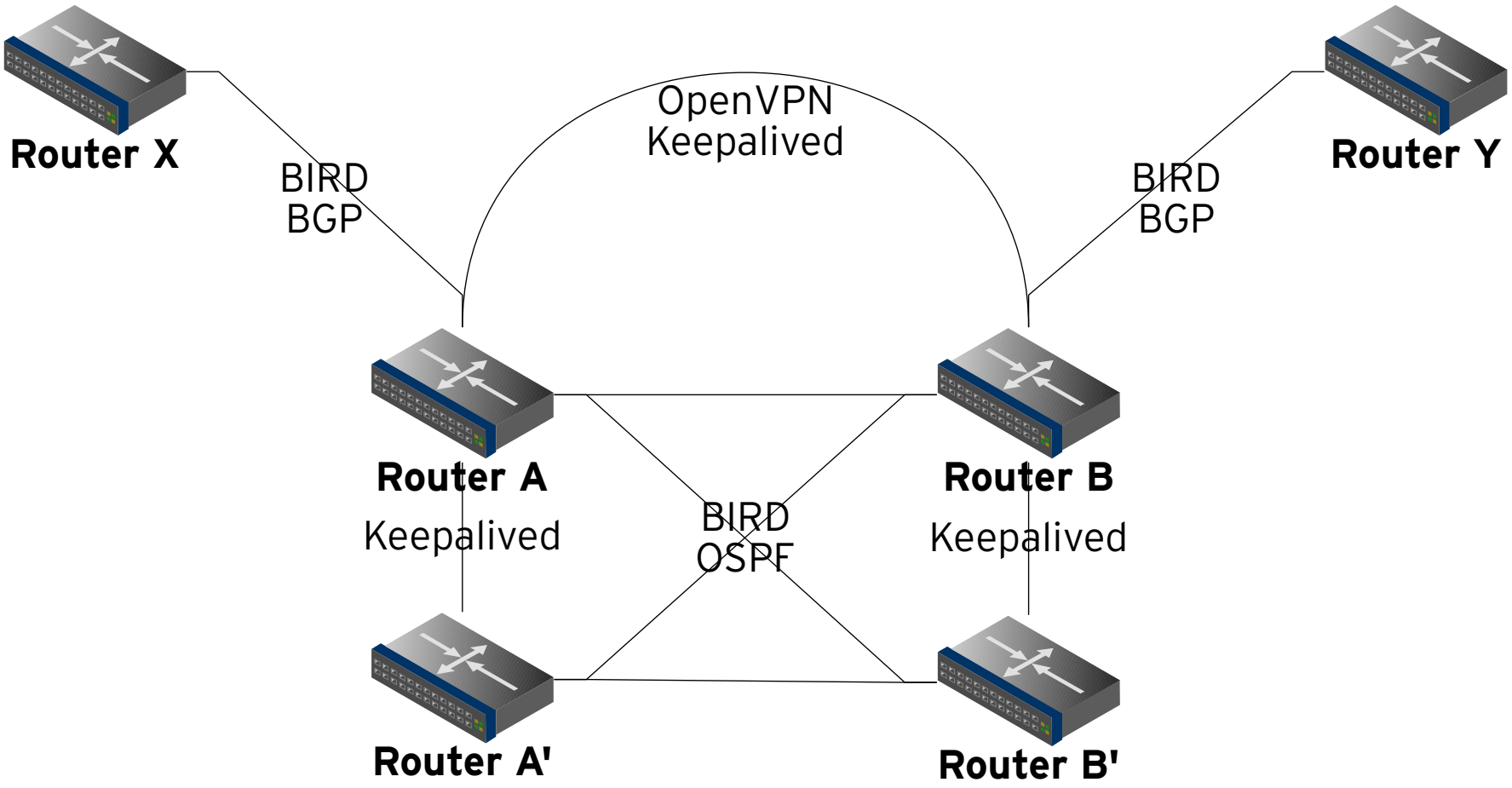
```
vrrp_instance vlan2 {
    virtual_router_id 2
    state MASTER
    priority 100
    interface br0.2
    virtual_ipaddress {
        203.0.113.1/24
    }
}
```

Router A'

```
vrrp_instance vlan2 {
    virtual_router_id 2
    state BACKUP
    priority 75
    interface br0.2
    virtual_ipaddress {
        203.0.113.1/24
    }
}
```

- Interface Setup auf beiden Seiten
  - Interface muß UP sein
  - mit privater IP

Fazit


Router X
BIRD
BGP
OpenVPN
Keepalived
BIRD
BGP
Router Y
Router A
Keepalived
BIRD
OSPF
Router B
Keepalived
Router A'
Router B'

Robert Sander <r.sander@heinlein-support.de>

Fazit

- Open-Source-Komponenten für Router vorhanden
  - BIRD
  - Keepalived
  - OpenVPN
  - bridge-utils
  - iproute2
- Setup nicht unkomplex
- viele Einzelteile zu orchestrieren
  - Router mit teilweise individueller Konfiguration
- Macht Spaß

- Natürlich und gerne stehe ich Ihnen jederzeit mit Rat und Tat zur Verfügung und freue mich auf neue Kontakte.
  - Robert Sander
  - Mail: r.sander@heinlein-support.de
  - Telefon: 030/40 50 51 - 43
- Wenn's brennt:
  - Heinlein Support 24/7 Notfall-Hotline: 030/40 505 - 110

Soweit, so gut.

Gleich sind Sie am Zug:
Fragen und Diskussionen!

Wir suchen:
Admins, Consultants, Trainer!

Wir bieten:
Spannende Projekte, Kundenlob, eigenständige Arbeit, keine Überstunden, Teamarbeit

...und natürlich: Linux, Linux, Linux...

http://www.heinlein-support.de/jobs

Robert Sander <r.sander@heinlein-support.de>

Heinlein Support hilft bei allen Fragen rund um Linux-Server

HEINLEIN AKADEMIE

Von Profis für Profis: Wir vermitteln die oberen 10% Wissen: geballtes Wissen und umfangreiche Praxiserfahrung.

HEINLEIN HOSTING

Individuelles Business-Hosting mit perfekter Maintenance durch unsere Profis. Sicherheit und Verfügbarkeit stehen an erster Stelle.

HEINLEIN CONSULTING

Das Backup für Ihre Linux-Administration: LPIC-2-Profis lösen im CompetenceCall Notfälle, auch in SLAs mit 24/7-Verfügbarkeit.

HEINLEIN ELEMENTS

Hard- und Software-Appliances und speziell für den Serverbetrieb konzipierte Software rund ums Thema eMail.

Referenzen

- Grafiken BGP-Statistiken
  - By Mro (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html) or CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons