Freifunk Hochstift Infrastruktur

Freifunk Tag 2017 #fftag17

Maximilian Wilhelm (@BarbarossaTM)
Freifunk Hochstift (@ffhochstift,
@ffho_noc)

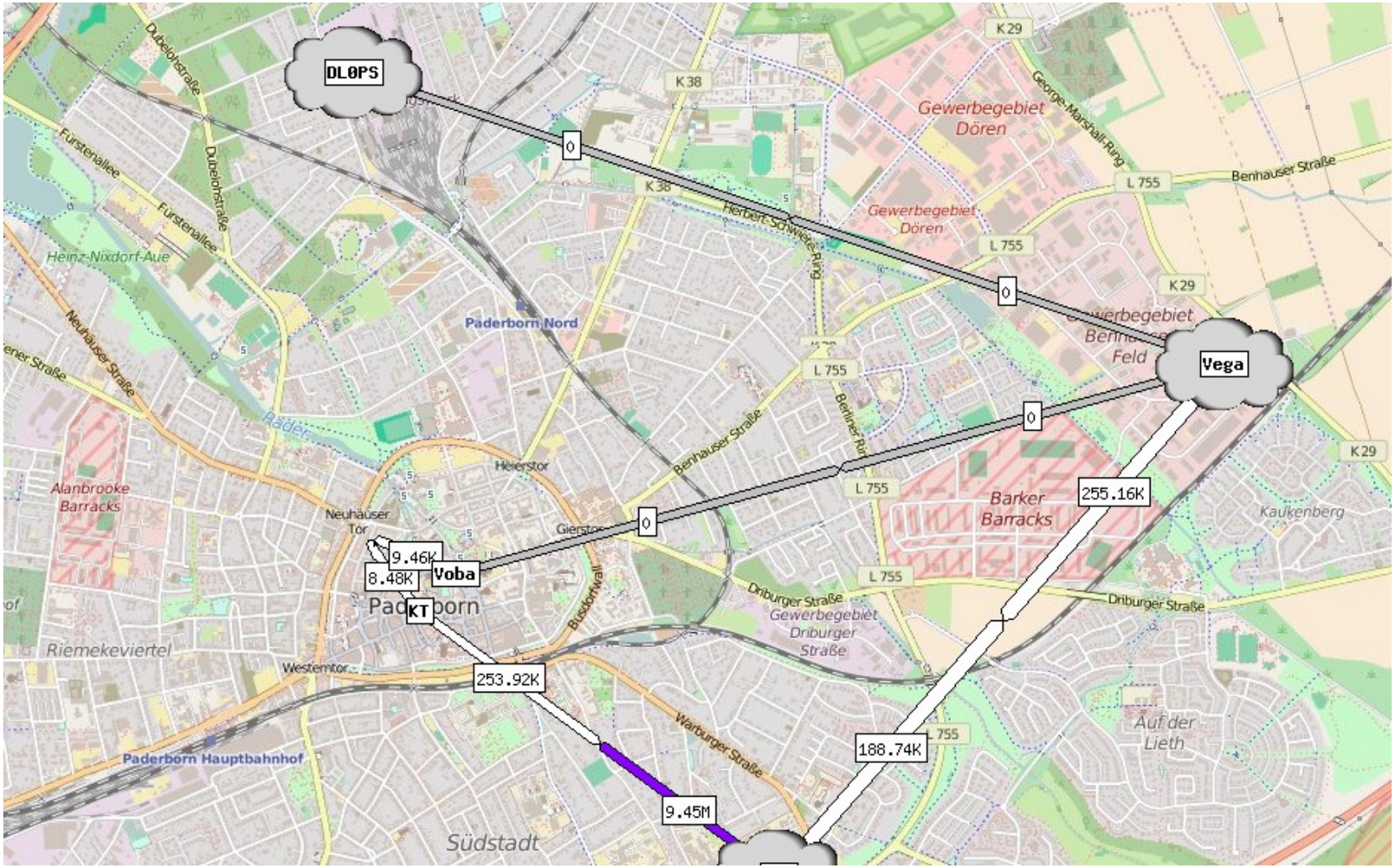
Background: Freifunk Hochstift

- Gegründet als Freifunk Paderborn (12/2013)
- Schnell stark gewachsen dank viel Unterstützung
  - Bäcker
  - Werbegemeinschaft Paderborn
  - Land NRW
  - Seit 2016 auch Stadt + Kreis Paderborn
- Expandiert zum Freifunk Hochstift (09/2015)
- ~900 Knoten, ≥1.500 Clients

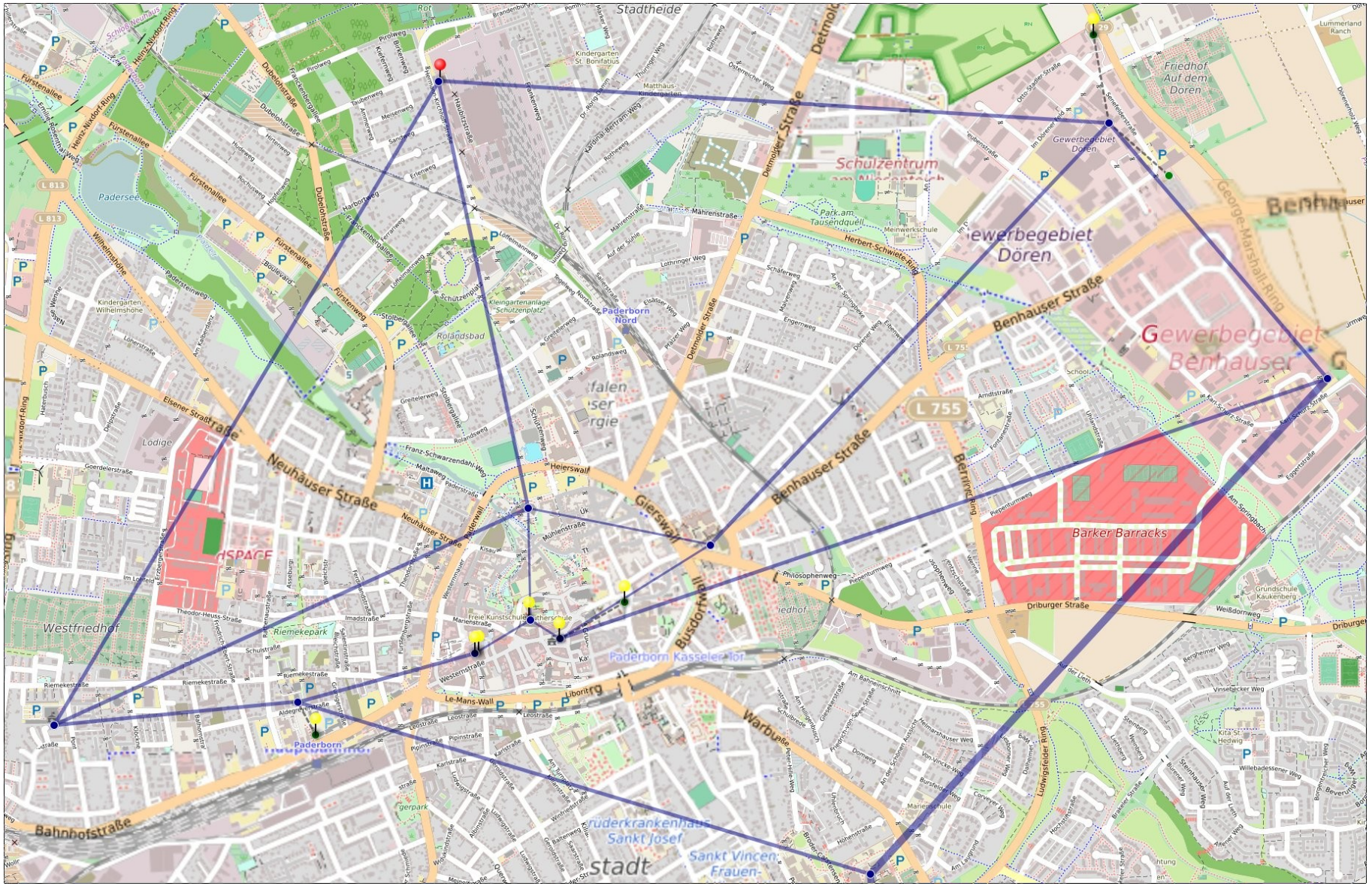
Backend-Infrastruktur

- Mittlerweile über 60 Systeme
  - Debian Linux auf Blech oder in VMs
  - An immer mehr POPs im Hochstift
  - Server + VMs verteilt in .de
  - Angebunden per RiFu-Backbone oder VPNs
- Monitoring per LibreNMS + Icinga2 (upcoming)

Wireless Backbone (alt)



Wireless Backbone (Planung)



~~BATMAN~~ Layer2 skaliert nicht

- Ein BATMAN Mesh mit 1k Knoten dreht nicht
- Ergo kleinere “Sites” bauen
  - Legacy + 10 neue (City, PB N, O, S, W, ...)
- Aber:
  - Ein Vlan pro Site und RF-Strecke?
  - Plus Vlan für IP?

Erkenntnisse

- $Linux-Appliance als BATMAN Hop (APU2)
- BCP38 hilft
- Point-To-Point Links (VPNs) bauen
- #routingrocks
- Teuer nicht unbedingt besser
- L2-Kopplung per Vlans anstrengend

- Infrastruktur-Manufaktur skaliert nicht
- Bitte kaufen Sie Automatisierung

Automatisierungsstufen

- Gateway #1 Handgeklöppelt
- Gateway #2 per setup.sh Basiskonfiguriert
- Gateway #3-6 per setup.sh initial Vollkonfiguriert
- Aber Changes?
  - Manuell überall anders
- Skaliert nicht

Salt Stack

- Continuous Management gewollt
  - Pakete installieren
  - Configs anpassen
  - Dienste/Units starten
  - Netzwerk konfigurieren
  - Zertifikate verteilen
  - ...
- 1. Ansatz hat ~1 Jahr gehalten
- Mittlerweile fast einmal alles neu gebaut

Automatisierung

States

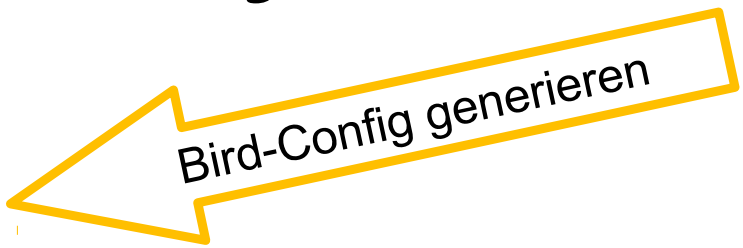
- Repräsentiert Status den $etwas haben soll
- YAML-Format
- Sammlung von Definitionen für
  - (De)Installierte Pakete
  - (De)aktivierte Services
  - Dateiinhalte
  - User
  - …
- Abhängigkeiten modellierbar

Pillar

- Strukturierter Key-Value-Store
- YAML-Format
- Erlaubt Filterung wer was lesen darf
- Daten in Templates auslesbar
- Prädestiniert für
  - Keys
  - Host-spezifische Konfigurationen

Pillar Example

```
bbr-vega.in.ffho.net:
   id: 198
   sysLocation: Vega
   roles:
      - router
      - batman
      - bbr
   sites:
      - pad-cty
```



Pillar Example contd.

```
ifaces:
    bond0:
      bond-slaves: "eth0 eth1 eth2"

    vlan1002:
      desc: "<-> gw04"
      vlan-raw-device: bond0
      prefixes:
        - 10.132.253.58/31
        - 2a03:2260:2342:fe1c::1/126
      batman_connect_sites: pad-cty
[...]                                                    #
```

The SDN part

#SDN

Disclaimer: Schriftart auf besonderen Wunsch von AbraXXL

SDN für Linux (Debian)?

- Klassisches ifupdown nur mäßig automatisierbar
- /etc/network/interfaces generieren einfach
- Aber wie neu laden?
  - »service networking restart« disruptive
  - Kein Tool für “neu laden” vorhanden
  - Untrivial zu bauen
- CumulusNetworks Ifupdown2
  - Rewrite von ifupdown in Python
  - https://github.com/CumulusNetworks/ifupdown2

ifupdown2

- Leider keine Featureparität mit ifupdown
- Kann von Hause aus
  - Dependency Resolution
  - `ifreload`
  - VLAN-Aware-Bridges
  - VRFs
  - VXLAN
- Kann (noch) nicht:
  - ppp

Ifupdown2 Patches

- Dank Python leicht erweiterbar
- Upstream kommunikativ
- Kann mittlerweile
  - B.A.T.M.A.N. Interfaces
  - Tunnel (GRE, SIT, IPIP)
- Offene Pull-Requests für
  - Filter für Bridge-Interfaces
  - Phys-dev für VXLAN
  - Pointopoint-Bugfix

Automatisierung mit SaltStack

- Node-Informationen in »pillar«
  - Strukturierter Key-Value-Store in YAML-Syntax
- Eigene Python Module für “SDN” und mehr
- Daraus generiert:
  - /etc/network/interfaces
  - Bird-Config (OSPF, iBGP, eBGP)
  - OpenVPN
  - DHCPd

`https://github.com/FreifunkHochstift/ffho-salt-public`

Netzwerk-Setup

- Alle Systeme per DualStack verbunden
  - Dynamisches Routing mit Bird
  - OSPF
    - Loopback-Reachability
    - Propagation von Mgmt-Netzen der POPs
  - iBGP mit 3 Core-Routern (RRs)
    - Default-Route (kommt per eBGP)
    - Announcement von Client-Netzen
    - Announcement von (Anycasted) Service-IPs
    - Traffic Engineering

`https://github.com/FreifunkHochstift/ffho-salt-public/tree/master/bird`

Recap

- ✔ Erledigt
  - ✔ Automatisierung
  - ✔ Geroutete Infrastruktur
- Next up
  - Ausfallsichere Services
  - Elegantes und sicheres Routing
  - B.A.T.M.A.N.-Overlays

Exkurs: Anycast

- Idee: Service-Prefix mehrfach announcen
- Client verbindet immer zu nahem Server
  - Nähe aus Sicht des Routings!
- Realisierung: anycast-healthchecker + bird
  - Service-Prefix nur announcen, wenn Dienst OK
  - Obacht: Flow-Based ECMP nutzen (ab Kernel 4.4)
- Fällt ein Server aus, “Umleitung” zum nächsten

```
https://github.com/unixsurfer/anycast_healthchecker
https://github.com/FreifunkHochstift/ffho-salt-
public/blob/master/bird/ff-policy.conf
```

Exkurs: Traffic Engineering

- Steuerung von Traffic-Flows
  - Ingress-Traffic steuern
  - Kürzeste Weg zu Batman-Gateway-Prefixen
  - → Ost-West-Traffic vermeiden
- Lösung:
  - More-Specific Routen von/zu gewünschtem Ziel
  - Mit spezieller BGP-Community gekennzeichnet

VRFs

- Virtual Routing and Forwarding
- Unabhängige Routinginstanzen
  - Layer3 Separation
  - Strikte Trennung von Netzen
  - Überlappende Prefixe möglich
- L3-VPNs
  - Üblicherweise im Kombination mit MPLS
- “VRF ohne MPLS” → VRF-lite
  - Hier: VRF-lite

VRFs vs. Policy Routing

- Alt bekannt: Policy-Routing (seit Kernel 2.2)
- Fussschusspotential
  - Rules für v4 und v6 vorhanden?
  - Rules für alle Interfaces vorhanden?
  - Rules für alle Source-Prefixe vorhanden?
  - Pipe-Protokoll in Bird
  - Management-Katastrophe

VRFs vs. Network Namespaces

- Seit Kernel 2.6.24++
- NetNS haben eigene
  - Routingtabellen
  - Routing Policies
  - Netfilter Regeln
- Device Layer separation
- Prozesse müssen in NetNS laufen
- Uncharmant im Freifunk Umfeld
  - “Zuviel des Guten”

VRFs unter Linux

- Separierung für Layer3 Kommunikation
- VRF-Interface als Master für “echte” Interfaces
  - Legt Routing-Table für VRF fest
- Ab Kernel 4.[345] (nehmt >= 4.9)

`https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/networking/vrf.txt`

`https://cumulusnetworks.com/blog/vrf-for-linux/`

`https://de.slideshare.net/CumulusNetworks/operationalizing-vrf-in-the-data-center`

VRFs unter Linux

```
ip link add VRF_DEVICE type vrf
            table ID

ip link set dev DEVICE
            master VRF_DEVICE
```

VRFs mit ifupdown2

```
auto eth0
iface eth0
        address 185.46.137.163/25
        address 2a00:13c8:1000:2::163/64
        gateway 185.46.137.129
        gateway 2a00:13c8:1000:2::1
        vrf vrf_external


auto vrf_external
iface vrf_external
       vrf-table 1023
```

VRF-Konzept

- Haupt-VRF mit internem Freifunk-Netz
  - OSPF + iBGP
  - Routen zu allen internen Hosts und Diensten
  - Debugging mit Standardtools
- Ggf. “external” VRF für Interfaces mit public IPs
  - GRE-Tunnel zu AS201701
  - OpenVPN-Verbindungen
  - Fastd-Einwahl
  - Eigene public facing services

Inter-VRF-Kommunikation

- Nur in Ausnahmefällen erforderlich
- Erfordert vEth-Paar
  - Quasi virtuelles Netzwerkkabel
- Ein Ende in Haupt-VRF, eins in VRF “external”
- Bird spricht BGP mit sich selbst
  - Exportiert aggregierte Prefix(e)
  - Importiert Public IP(s)
- Public IP(s)s intern redistributiert

Veth unter Linux

```
ip link add VETH_END1 type veth
            peer name VETH_END2
```

```
# ip l
[...]
24: VETH_END2@VETH_END1:   [...]
25: VETH_END1@VETH_END2:   [...]
```

OpenVPN vs. VRFs

- Viele OpenVPN Tunnel im Einsatz
- OpenVPN muss über VRF “external” reden
- Dafür brauchts einen kleinen Patch

```
setsockopt (sd, SOL_SOCKET,
SO_BINDTODEVICE, dev, strlen(dev)
+1);
```

- https://github.com/OpenVPN/openvpn/pull/65

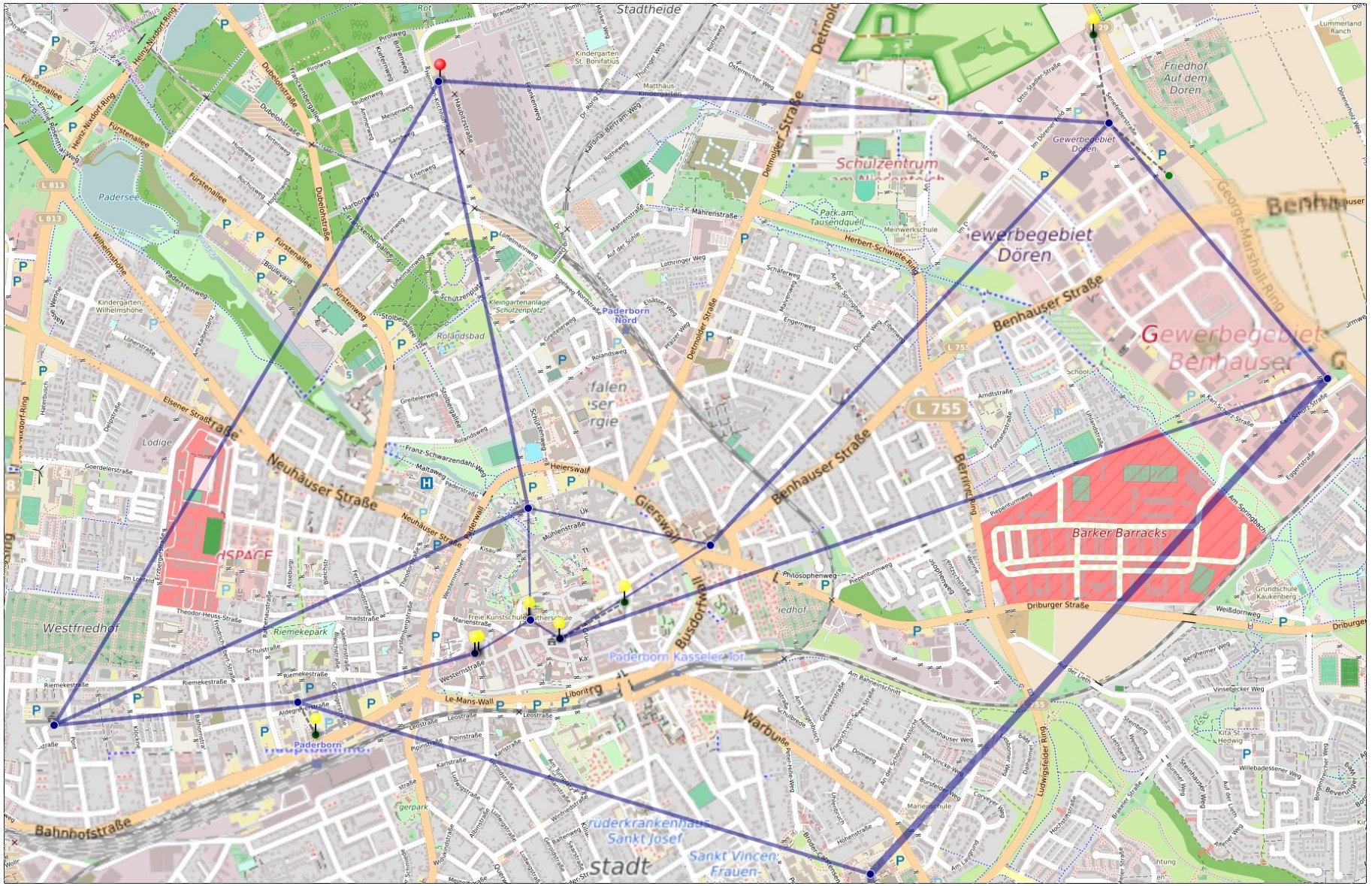
Upcoming: VRF support für pppd

- An einem Standort DSL-Uplink vorhanden
- ppp0 sollte in VRF “external”
- Mit post-up scripts geht's leider nicht direkt
  - Mit 3 Skripten und at geht's
  - Ist aber nicht schön
- pppd braucht wohl auch einen Patch :-)

Recap

- ✔ Erledigt
  - ✔ Geroutete Infrastruktur
  - ✔ Automatisierung
  - ✔ Ausfallsichere Services
  - ✔ Elegantes und sicheres Routing
- Next up
  - B.A.T.M.A.N.-Overlays

Wireless Backbone (Planung)



Wege aus der VLAN-Hölle

- B.A.T.M.A.N. braucht Layer2-Verbindung
- IP-Backbone vorhanden
- Layer2-Overlay wäre praktisch!
  - MPLS unter Linux bisher nur tw. Vorhanden
  - ➔ VXLAN

VXLAN

- “Ethernet over UDP”
  - Oder: “Poor mans approach to MPLS”
- Designed als Layer2-Overlay in Datacentern
  - Multi-tenant Overlay über IP-Fabric
  - 24Bit VNI => 16M Instanzen möglich
  - Unicast/Multicast Kommunikation
  - Endpunkt = VTEP (VXLAN Tunnel End Point)
- RFC7348

VXLAN unter Linux

```
ip link add DEVICE type vxlan id ID
[ dev PHYS_DEV ]
[ { group | remote } IPADDR ]
[ local { IPADDR | any } ]
[ … ]
```

VXLAN mit ifupdown2

```
auto vx_v1002_padcty
iface vx_v1002_padcty
        vxlan-id          655617
        vxlan-physdev     vlan1002
        vxlan-svcnodeip 225.10.2.1
        #
        hwaddress f2:00:c1:01:10:02
        mtu 1560
```

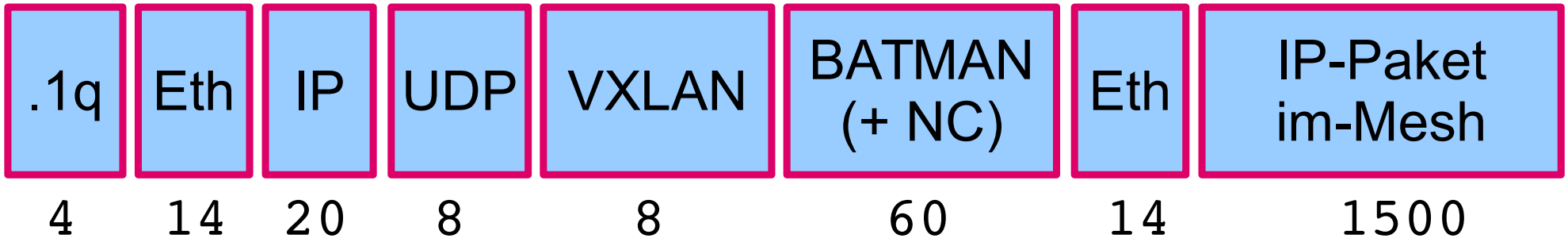
#

IPoBATMANoVXLANoVLANoRF

- Wait, what?

MTU

- »Increasing MTUs for fun and profit«
  - Mesh-Netz (BATMAN): 1500
  - BATMAN-Underlay (VXLAN): 1560
  - VXLAN-Underlay (VLAN): 1610
  - On-Wire: 1628



https://github.com/FreifunkHochstift/ffho-salt-public/blob/master/_modules/ffho_net.py#L77

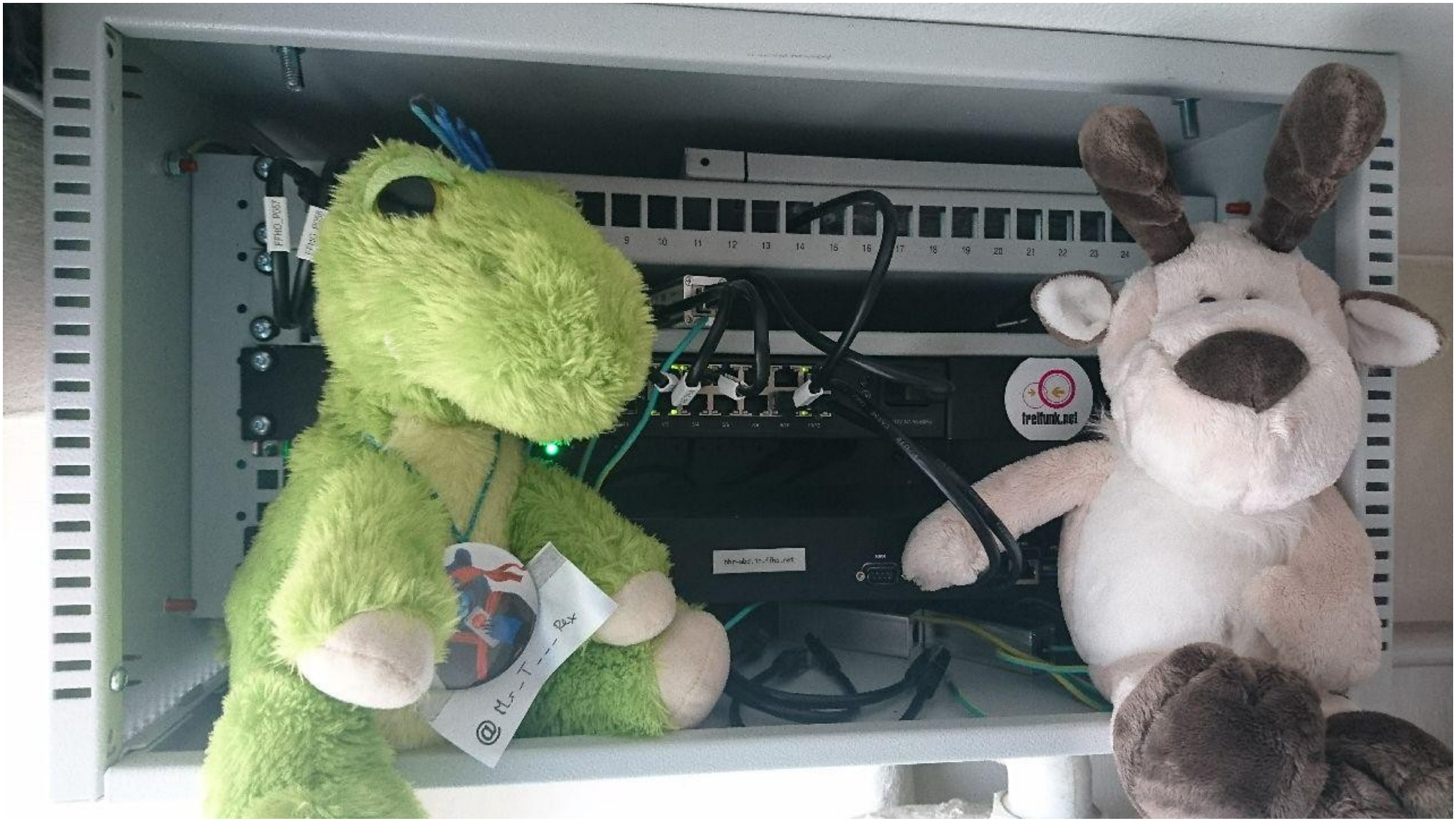
Recap

- Erledigt
  - Geroutete Infrastruktur
  - Automatisierung
  - Ausfallsichere Services
  - Elegantes und sicheres Routing
  - B.A.T.M.A.N.-Overlays

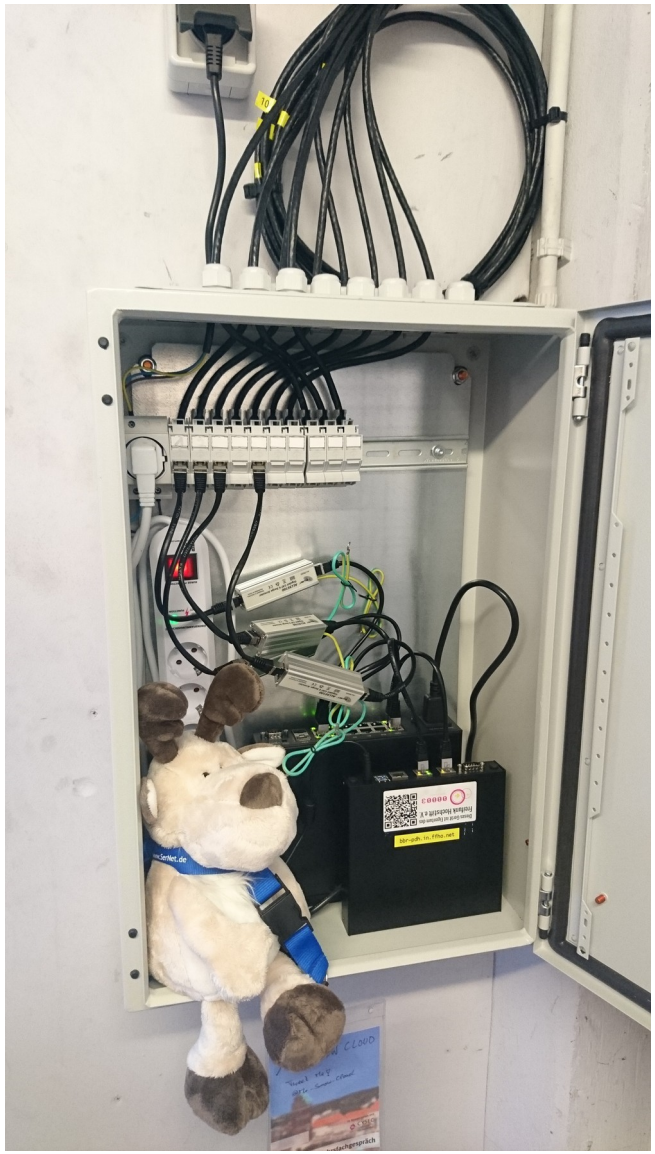
Hardware

- Zoo aus z.T. gesponsorter Hardware
  - Server, Switche, Richtfunk, ..
- Versuch der Homogenisierung
  - PCengines APU2
  - Netonix WISP Switches
  - Ubiquiti Networks
    - PowerBeam
    - LiteBeam
    - AC Mesh Pro

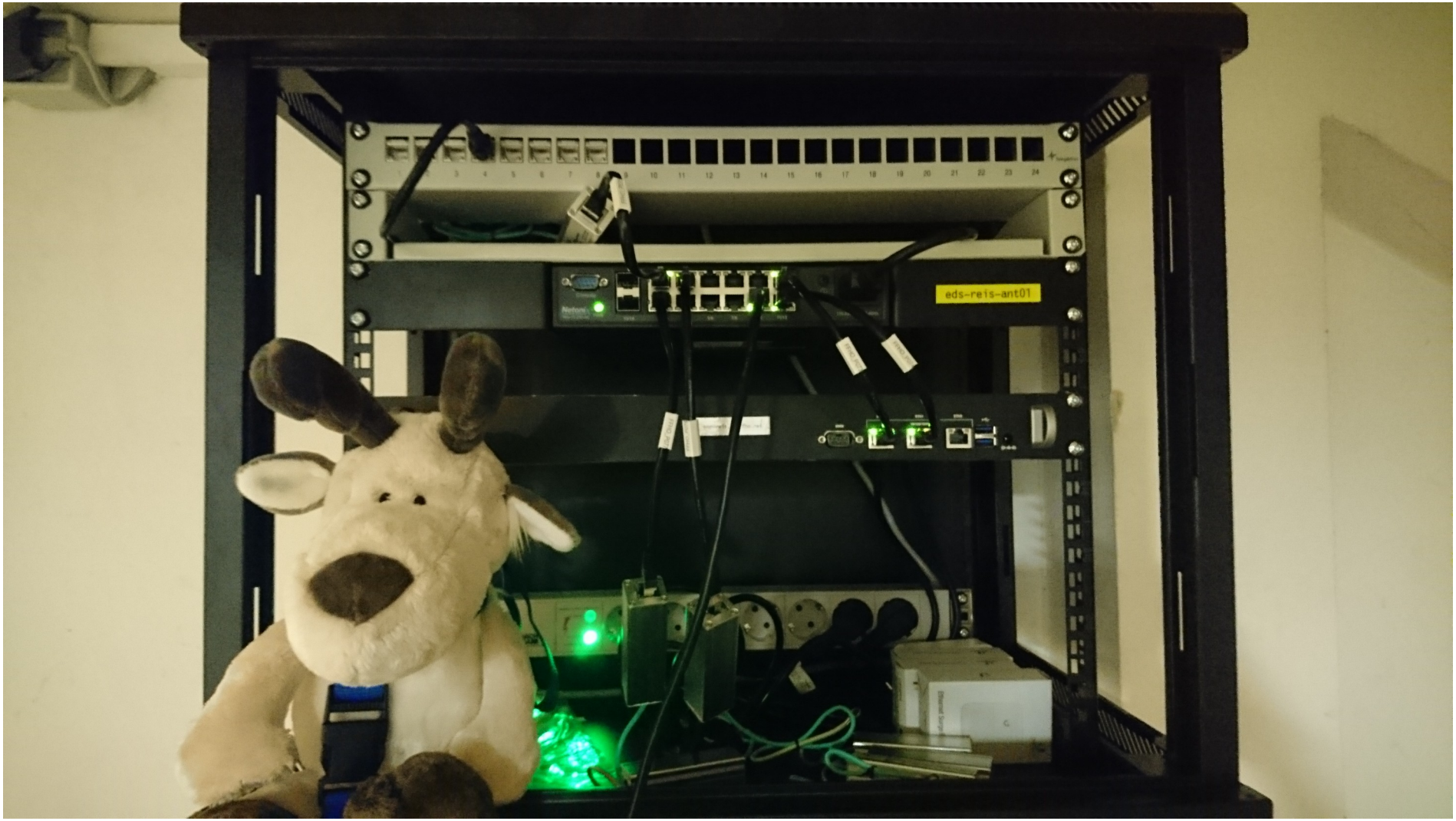
Abdinghof



Aufbau PaderHalle (gestern)



Reismann Gymnsaium (gestern)



Freifunkromantik

Lessons Learned

Lessons Learned

- Offload ist ein Thema voller Missverständnisse
  - Abschalten! (GS, GRO, GSO, TSO)
  - 4KB/s vs. 40MB/s
- BCP38 ausrollen
- Kernel >= 4.9 nehmen
  - Mit 4.6 / 4.7 IPv6-Routing in VRF subtil kaputt
  - Mit 4.8
    - Problem mit Bridges und B.A.T.M.A.N.
    - IPv6 und Fragmentation

Systemd + OpenVPN vs. ifup

- Einige OpenVPN Instanzen konfiguriert
- “up /etc/openvpn/ifup”
  - ifup “$1”
- Dank systemd starten Instanzen parallel
  - Einige ifup-Aufrufe parallel
  - Nahezu keine IPs mehr konfiguriert

➔ `flock –exclusive –wait 30`

Fragen?

Maximilian Wilhelm @BarbarossaTM

Freifunk Hochstift @ffhochstift + @ffho_noc

B.A.T.M.A.N. auf EdgeRoutern

- Wir haben kein Geld
- EdgeRouter sind billig
- Wäre cool, wenn die auch BATMAN sprächen..
- ➔ https://git.c3pb.de/freifunk-pb/edgerouter
- Nice try, aber
  - Nicht update-safe
  - Explodiert auf andere Modellen und Versionen
  - Performance eher so mittel, da kein offloading

Heimnetz-Freifunk-Bridge / BCP38

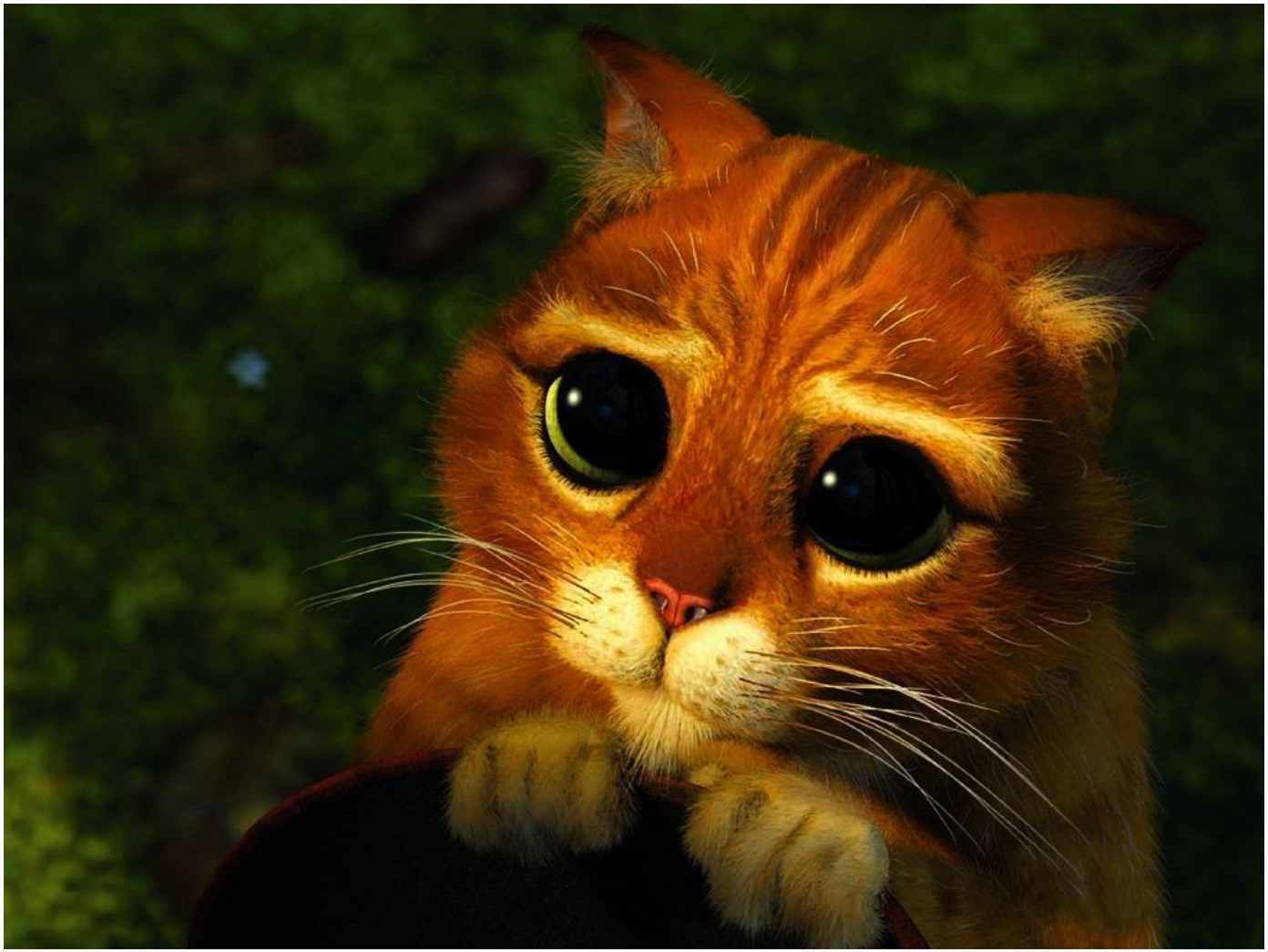
- Fritz!Box Prefix im Freifunk-Netz?
- Firmennetz im Freifunk-Netz?
- ➔ Prefixfilter für Freifunk-Prefix auf Knoten entfernt 30% Grundrauschen

https://github.com/FreifunkHochstift/ffho-packages/tree/master/ffho/ffho-ebtables-net-rules

Tinc

- “Viele Point-to-point Tunnel sind kompliziert.”
- “Ein großes Layer2-Netz ist easy. Da drüber kann man auch OSPF fahren.”
- “Mit zwei VPN-Servern haben wir auch gleich Redundanz.”
- Don't.

I can haz PoE switch for AF-5X?



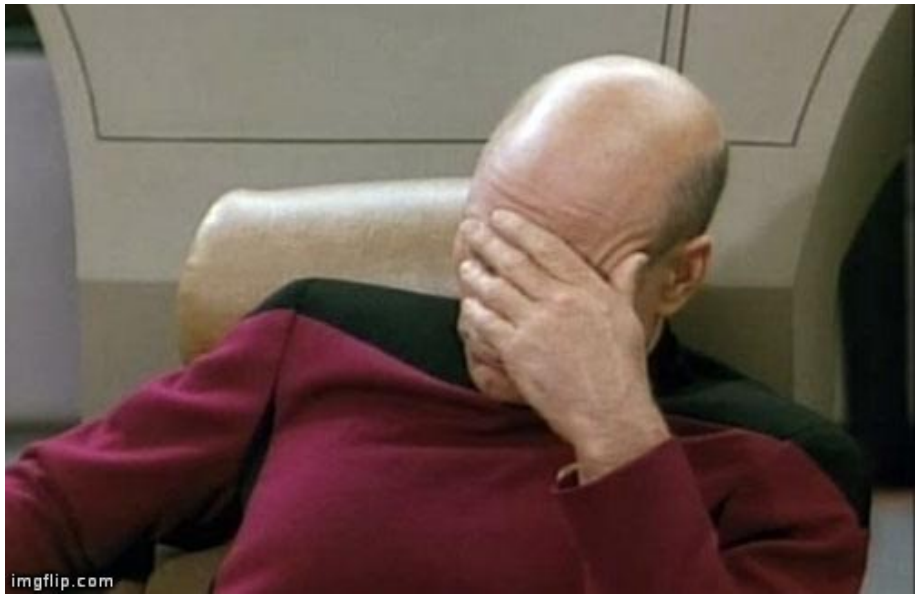
Nicht von UBNT. Netonix to the rescue!

AirFiber 5X

- Wetterradar (Beware of 5600MHz Band)
- Störungen in anderen Bändern (5,5 vs. 5,8GHz)
- Störungen auf parallel laufenden Kabeln?
- Weniger Durchsatz als PBE-5AC-500
- Montagsmodelle?

Bird 1.6.1 doom

- Bird 1.6.1 + Pipe-Protokoll = Segfault
- Unattended-upgrades ist 'ne tolle Sache
- Salt state pkg.latest auch
- Freifunk Hochstift war zweimal “aus”.
- Oops.



Offloading

- Der Unterschied zwischen 4KB/s und 40MB/s...

```
for iface in eth0 eth1; do
    for feature in sg gro gso tso
rxvlan txvlan; do
        ethtool --offload ${iface}
                  ${feature} off
        done
done
```