Die Technik hinter Freifunk Franken

ICMP7

Tim Niemeyer <tim@freifunk-franken.de>

https://github.com/FreifunkFranken/vortrag-icmp7.git
5e8e4fa

09.08.2014

Mithilfe

Danke

- Christian Berger <casandro@casandro.dyndns.org>
- Johannes Stefan <email@johannesstefan.de>

Inhalt

Einleitung

- Freifunk Franken ist lokaler Ableger der Freifunk-Bewegung (freifunk.net)
- Nicht-kommerzielle Initiative für freie Funknetzwerke
  - → Bürger investieren in Eigenregie Zeit, Geld und Enthusiasmus
- Nicht nur „kostenloses Internet“ ⇒ „freies Netzwerken“
  - Lokal intressante Dienste zur Verfügung stellen (Webcams)
  - Text, Musik und Filme über das interne Freifunk-Netz übertragen
  - Über lokale Dienste Chatten oder Telefonieren

Einleitung

Wie es funktioniert

- Freifunker stellen WLAN-Router für sich selbst und den Datentransfer der anderen Teilnehmer zur Verfügung
  - ggf. mit Anschluss an das www (für VPN)
- Benachbarte Router verbinden sich und spannen ein sogenanntes Mesh-Netzwerk auf
- Nicht benachbarte Router verbinden sich mittels VPN-Tunnel zum Freifunk
- Jegliche Verbindung ins www wird hierüber umgeleitet, um Risiken der Störerhaftung zu entgehen

Einleitung

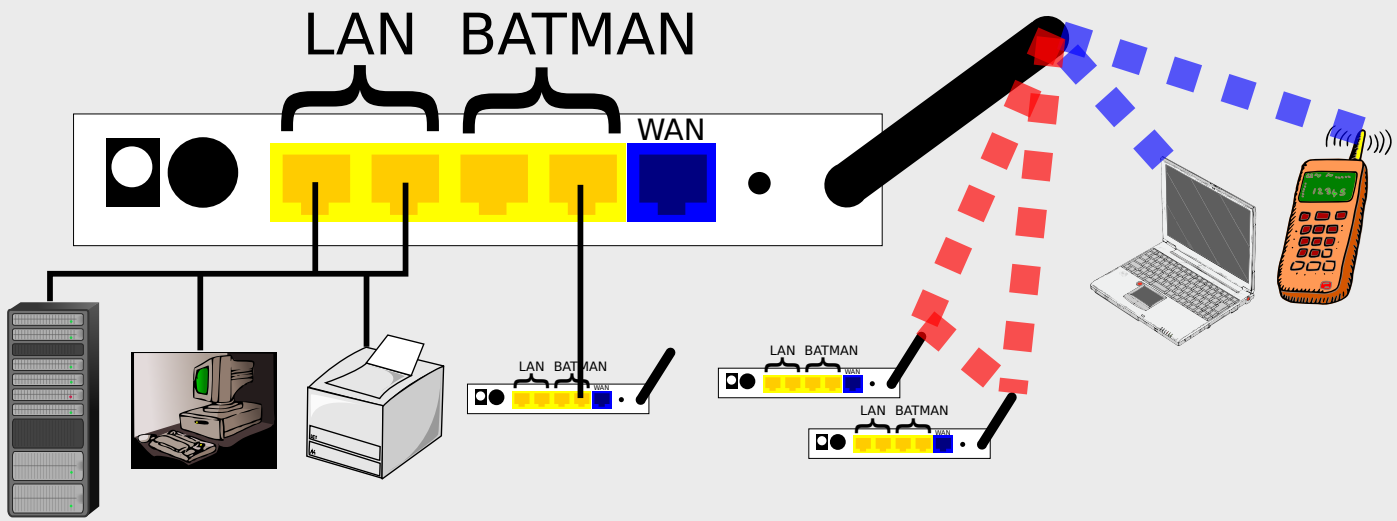
Was braucht man?

- Ein günstiger, unterstützter Router (ab ca. 17€)
- Eine spezielle Firmware
- Die Zustimmung zum „Pico-Peering Agreement“
  - Regelwerk, das grundsätzliche Eigenschaften eines Freifunk-Netzwerkes sichert
    1. Freier Transit
    2. Offene Kommunikation
    3. Keine Garantie (Haftungsausschluss)
    4. Nutzungsbestimmungen
    5. Lokale (individuelle) Zusätze
  - Die Freifunk Firmware implementiert diese Grundsätze standardmäßig

Anwendungsbeispiele

Anschlüsse

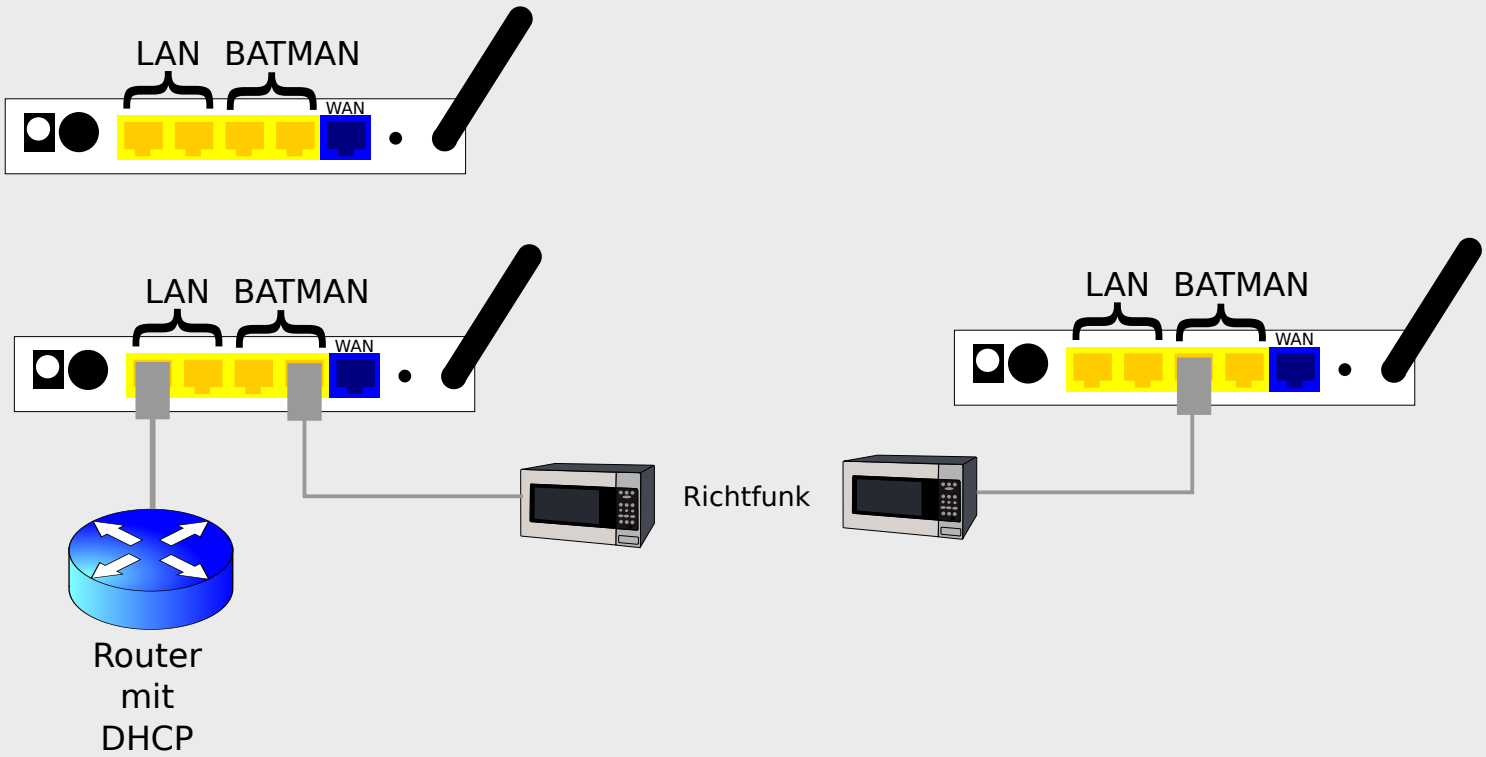
- Client-Ports & Infrastructure Funknetz: Wie ein großer Switch
- Batman-Ports & Ad-Hoc Funknetz: Mesh-Netz
- WAN-Ports: VPN Netz



Anwendungsbeispiele

Autarkes Layer-2 Netz

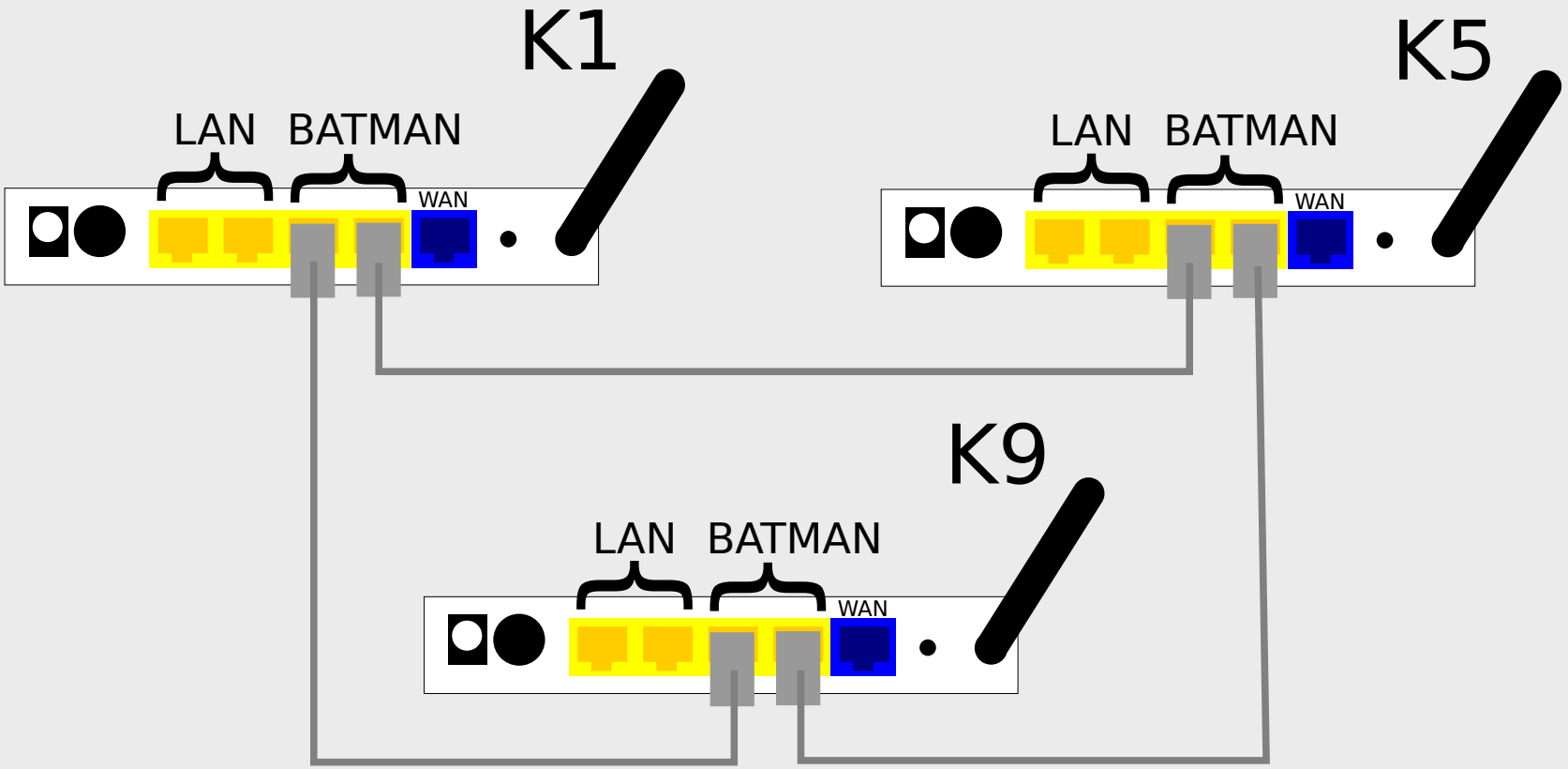
Eine Wolke kann ihr Netz von einem Router bekommen der an einen der Client-Ports angeschlossen wird.



Anwendungsbeispiele

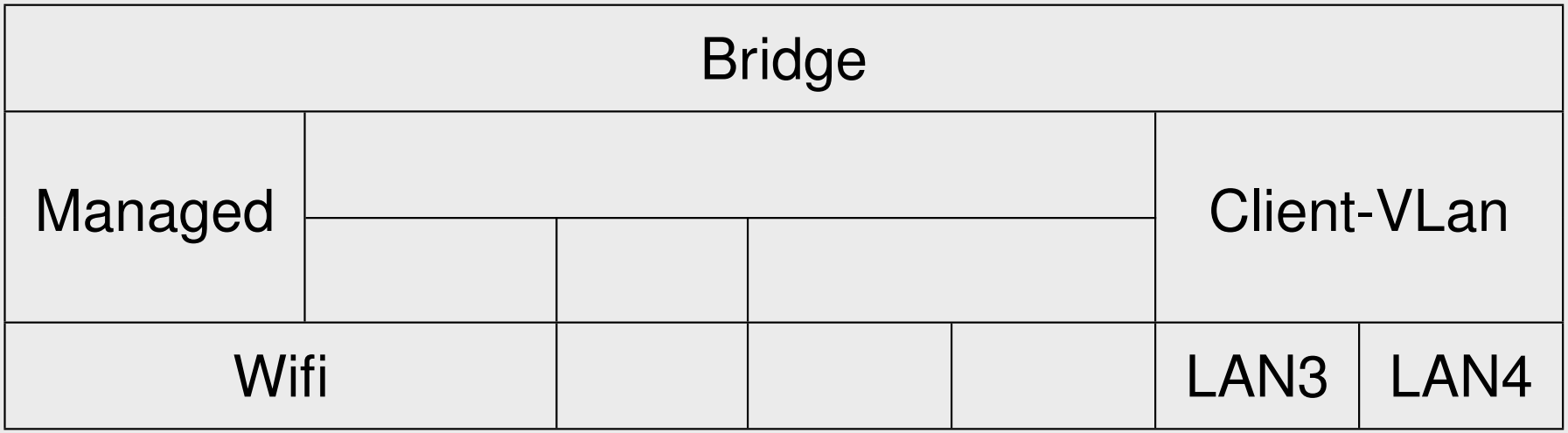
Standort mit mehreren Kanälen

Mehrere Knoten können beliebig untereinander per Kabel verbunden werden.



Mesh Router

Jeder Knoten ist wie ein großer Switch



- Clients, die sich vor Ort per WLan verbinden
- Clients, die sich per Kabel verbinden

Mesh Router

Jeder Knoten ist wie ein großer Switch

<table>
<tr><td colspan="6">Bridge</td></tr>
<tr><td rowspan="2">Managed</td><td colspan="3" rowspan="2">B.A.T.M.A.N</td><td colspan="2" rowspan="2">Client-VLan</td></tr>
<tr></tr>
<tr><td colspan="3">Wifi</td><td>LAN3</td><td>LAN4</td></tr>
</table>

- Clients, die sich vor Ort per WLan verbinden
- Clients, die sich per Kabel verbinden
- Das Freifunk-Netz

Mesh Router

Das Freifunk-Netz besteht aus

<table>
<tr><td colspan="6">Bridge</td></tr>
<tr><td rowspan="2">Managed</td><td colspan="3">B.A.T.M.A.N</td><td colspan="2" rowspan="2">Client-VLan</td></tr>
<tr><td>Ad-Hoc</td><td></td><td colspan="2">Node-VLan</td></tr>
<tr><td colspan="2">Wifi</td><td></td><td>LAN1</td><td>LAN2</td><td>LAN3</td><td>LAN4</td></tr>
</table>

- Knoten, die sich über WLan verbinden
- Knoten, die sich über Kabel verbinden

Mesh Router

Das Freifunk-Netz besteht aus

<table>
<tr><td colspan="6">Bridge</td></tr>
<tr><td rowspan="2">Managed</td><td colspan="3">B.A.T.M.A.N</td><td colspan="2" rowspan="2">Client-VLan</td></tr>
<tr><td>Ad-Hoc</td><td>VPN</td><td colspan="2">Node-VLan</td></tr>
<tr><td colspan="2">Wifi</td><td>WAN</td><td>LAN1</td><td>LAN2</td><td>LAN3</td><td>LAN4</td></tr>
</table>

- Knoten, die sich über WLan verbinden
- Knoten, die sich über Kabel verbinden
- Knoten, die über VPN verbunden werden

Firmware

Wie wird die Firmware gebaut?

- git clone ~~git://git.freifunk-ol.de/ffol/user/reddog/firmware.git~~
- cd firmware
- ./buildscript

Firmware

Wie wird die Firmware gebaut?

- git clone ~~git://git.freifunk-ol.de/ffol/user/reddog/firmware.git~~
- cd firmware
- ./buildscript

```
Please select a Board-Support-Package using:

./buildscript selectbsp
```

Firmware

Wie wird die Firmware gebaut?

- git clone ~~git://git.freifunk-ol.de/ffol/user/reddog/firmware.git~~
- cd firmware
- ./buildscript
- ./buildscript selectbsp bsp/board_wr1043nd.bsp
- ./buildscript selectcommunity community/franken.cfg
- ./buildscript

Firmware

```
Working with bsp/board_wr1043nd.bsp and community/franken.cfg

This is the Build Environment Script of the Freifunk Community
Oldenburg.
Usage: ./buildscript command
command:
      selectcommunity [communityfile]
      selectbsp [bsp file]
      prepare
      config
      build
      flash
      download

If you need help to one of these options just type
./buildscript command help
```

Community File

community/franken.cfg:

```
BATMAN_CHANNEL=1
ESSID_AP=franken.freifunk.net
ESSID_MESH=batman.franken.freifunk.net
BSSID_MESH=02:CA:FF:EE:BA:BE
NETMON_IP=fe80::ff:feee:1
VPN_PROJECT=fff
NTPD_IP=fe80::ff:feee:1%br-mesh
UPGRADE_PATH=http://[fe80::ff:feee:1%br-mesh]/dev/firmware/current/
```

Board-Support-Package

bsp/board_wr740n.bsp:

```
machine=wr740n
target=$builddir/$machine

board_prepare() {
}
board_prebuild() {
}
board_postbuild() {
    cp $target/bin/ar71xx/openwrt-ar71[..]-squashfs-*.bin ./bin/
}
board_flash() {
    echo "nothing implemented"
}
board_clean() {
    /bin/rm -rf $target bin/*$machine*
}
```

Board-Support-Package

bsp/wr740n/

- .config
- root_file_system/etc/
  - rc.local.board
  - config/
    - board
    - network
    - system
  - crontabs/
    - root

Was macht das Buildscript?

- Das BSP file wird als dot Script eingeladen
- Community file generiert dynamisches sed-Script
  - → Templates mit richtigen Werten zu füllen

Was passiert beim prepare?

- Sourcen Downloaden
  - → separater Source Folder
    - OpenWRT
    - Sämtliche Packages
      - → ggfs werden Patches angewandt
- Ein ggfs altes Target wird gelöscht
- OpenWRT wird ins Target exportiert (kopiert)
- Eine OpenWRT Feed-Config wird mit dem lokalen Source Verzeichnis als Quelle angelegt
- Die Feeds werden geladen
- Spezielle Auswahl an Paketen wird geladen
- Patches werden angewandt
- board_prepare wird aufgerufen (wird. z.B. für Patches für eine bestimmte HW verwendet)

Was passiert beim build?

- prebuild
  - $target/files aufräumen
  - Files aus default-bsp und bsp kopieren
  - OpenWRT- und Kernel-Config kopieren
  - board_prebuild
  - Templates transformieren
  - Versionen speichern: $target/files/etc/firmware_release
- OpenWRT: make
- postbuild
  - board_postbuild

Config Targets?

- prebuild
- OpenWRT: make menuconfig ; make kernel_menuconfig
- Speichern, y/n?
- Config-Format vereinfachen
- Config ins BSP zurück speichern

Allgemeines

- Verwendetes VPN: fastd
- Layer-II Netz
- Wir nutzen keine Verschlüsselung (! :-O)

fastd

There are no server and client roles defined by the protocol, this is just defined by the usage.

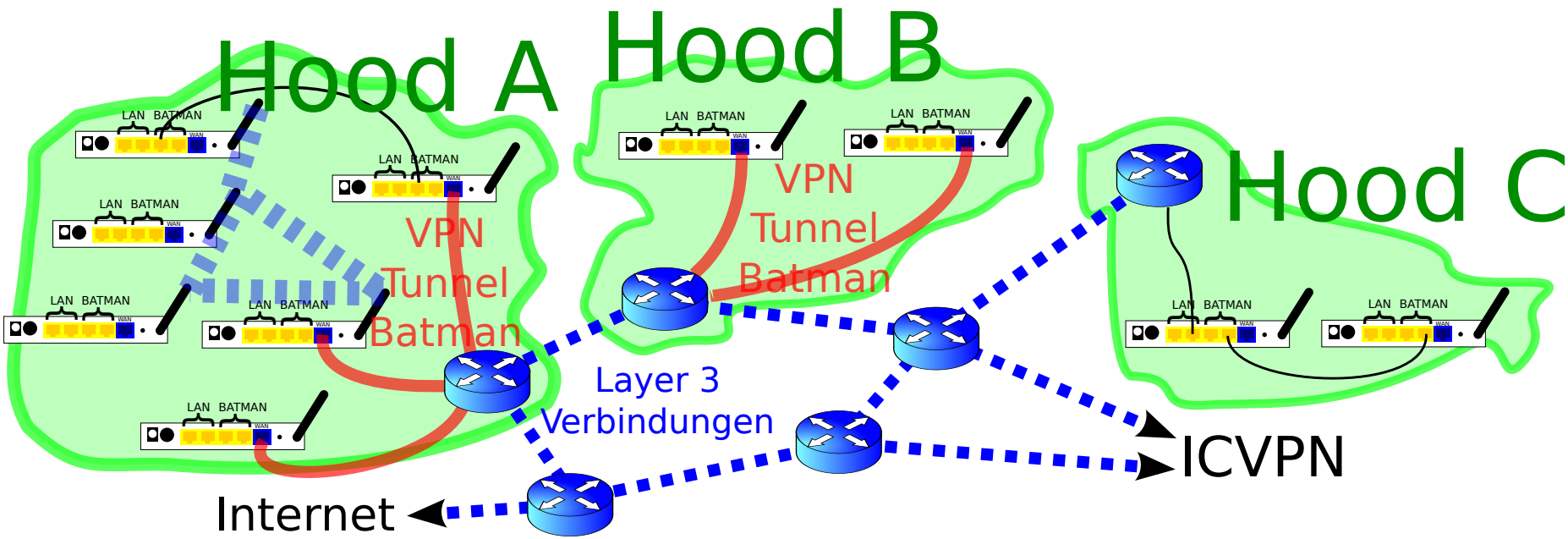
- Only one instance of the daemon is needed on each host to create a full mesh
- If no full mesh is established, a routing protocol is necessary to enable hosts that are not connected directly to reach each other

Allgemeines

- fastd Integration durch
  - fastdstart.sh auf der Client-Seite [1]
  - $project_$hood_fastd.sh auf der Server-Seite [2]
  - VPN-KeyXchange als Schlüsseltausch [3]
- Aufteilung in ,,hood"s:
  - Stellt ein Layer-II Netz dar
  - Ein Gateway kann mehrere Layer-II Netze bedienen

[1] bsp/default/root_file_system/etc/fastdstart.sh.tpl
[2] auf den Gateways (!) Todo: befreien!
[3] `https://github.com/FreifunkFranken/VPNkeyXchange`

Freifunk Hoods



Unser Freifunknetz ist in mehrere Layer-2 Inseln, die per Layer-3 miteinander verbunden sind, aufgeteilt. Die Hoods A und B in diesem Beispiel sind über unser VPN jeweise mit einem Router im Internet verbunden. Hood C hat einen lokalen Router, der direkt am LAN-Port hängt.

fastdstart.sh

- Testet Internet-Connectivität
- Legt fastd Konfiguration /etc/fastd/$project/ im tmpfs an: [4]
  - $project.conf
  - up.sh
  - down.sh
  - peers/
- Erzeugt Pub/Priv-Keypaar
- Startet fastd
- Meldet sich beim VPN-KeyXchange an
- Lädt Liste mit Peers
- Refresht fastd
-

[4] $project ist bei uns immer ,,fff”.

$project_$hood_fastd.sh

-
- Legt fastd Konfiguration /etc/fastd/$project.$hood/ an: [4]
  - $project.conf
  - up.sh
  - down.sh
  - peers/
- Erzeugt Pub/Priv-Keypaar
- Startet fastd
- Meldet sich beim VPN-KeyXchange an
- Lädt Liste mit Peers
- Refresht fastd
- Löscht verwaiste Peers

[4] $project ist bei uns immer ,,fff”.

VPN-KeyXchange

HTTP Schnittstelle

```
http://mastersword.de/~reddog/$project/?mac=$mac&name=$hostname&port=$port&key=$pubkey
```

Rückmeldung

```
####romauplink.conf
#name "romauplink";
key "9a8ee8b797eed5d3b06778c47fe670987a0eda791ca557da56e6198be45f24c6";
remote ipv4 "109.163.229.254"port 10000 float;

####fff.conf
#name "fff";
key "543ebc33b36210b10edf62fdb560e3ceac6c64b4ed9ad852f39954522934cd8a";
remote ipv4 "192.168.30.23"port 10000 float;

####90F652F45C34.conf
#name "ChamHH8Dachboden1";
key "d0e4900fae535d189ef070a527dd00ce2688f3ff7f2b31c4f2846cb658b855fb";
remote ipv4 "93.194.173.193"port 10000 float;
```

VPN-KeyXchange

- Knoten Identifizierung über MAC, alternativ über Name
- Jeweils pro hood:
  - Clients bekommen eine Liste aller Gateways
  - Gateways bekommen eine Liste aller Clients+Gateways

VPN-KeyXchange

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
3	ab:	default-1	abf	1	10000	false	1
4	cd:	default-2	aca	2	10000	false	1
5	ef:	fuerth-1	acb	3	10000	false	2

VPN-KeyXchange

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
3	ab:	default-1	abf	1	10000	false	1
4	cd:	default-2	aca	2	10000	false	1
5	ef:	fuerth-1	acb	3	10000	false	2

?mac=ab:&name=default-1&..

```
SELECT ID,gw,hood FROM nodes WHERE mac = $mac;
if (gw)
        SELECT * FROM nodes WHERE hood=$hood
else
        SELECT * FROM nodes WHERE hood=$hood & gw=true
```

VPN-KeyXchange

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
3	ab:	default-1	abf	1	10000	false	1
4	cd:	default-2	aca	2	10000	false	1
5	ef:	fuerth-1	acb	3	10000	false	2

?mac=ab:&name=default-1&..

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1

VPN-KeyXchange

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
3	ab:	default-1	abf	1	10000	false	1
4	cd:	default-2	aca	2	10000	false	1
5	ef:	fuerth-1	acb	3	10000	false	2

?mac=&name=nue1.fuerth&..

```
SELECT ID,gw,hood FROM nodes WHERE name = $name;
if (gw)
        SELECT * FROM nodes WHERE hood=$hood
else
        SELECT * FROM nodes WHERE hood=$hood & gw=true
```

VPN-KeyXchange

ID	mac	name	key	ip	port	gw	hood
0		roup	abc	254	10000	true	1
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
3	ab:	default-1	abf	1	10000	false	1
4	cd:	default-2	aca	2	10000	false	1
5	ef:	fuerth-1	acb	3	10000	false	2

?mac=&name=nue1.fuerth&..

ID	mac	name	key	ip	port	gw	hood
1		roup.fuerth	abd	254	10001	true	2
2		nue1.fuerth	abe	129	10001	true	2
5	ef:	fuerth-1	acb	3	10000	false	2

Gateway

Aktueller Zustand

- Gateway in Rumänien
  - → Direkt Internet
- Gateway in Nürnberg
  - → OpenVPN über Berlin

⟶ Das Beispiel hier stammt vom Gateway in Nürnberg

Gateway

Voraussetzungen

- Server (kein Spielzeug; also kein Pi)
- Richtiges Betriebssystem (z.B. ein Debian)
- Policy Based Routing
- DHCP- DNS-Server
- batman + fastd (fff_fuerth_fastd.sh)
- Lokale Netz Konfiguration
- Freifunk-VPN (z.B. ip-ip)
- olsr
- Ein Tunnel um Traffic ab zu werfen

Gateway

Setup

- IP Bereich
- .. was soll er tun? .. kurz Wiki durch gehen

Gateway

- Policy Based Routing
- Trennung der Routing Tabellen

/etc/iproute2/rt_tables:

```
10  fff
```

Gateway

/etc/default/isc-dhcp-server:

```
INTERFACES="bat1"
```

/etc/dhcp/dhcpd.conf:

```
option domain-name "freifunk-franken.de";
option domain-name-servers 10.50.16.1;
authoritative;
# fuerth
subnet 10.50.32.0 netmask 255.255.248.0 {
    range 10.50.35.0 10.50.36.255;
    option routers 10.50.32.2;
    option domain-name-servers 10.50.32.2;
}
```

Gateway

- Bind
- Namesauflösung für Internet-Anfragen

/etc/bind/named.conf.options:

```
//listen-on-v6 { any; };
allow-recursion { 10.50.0.0/16; };
```

Gateway

- batman-adv-2013.4.0
- batctl-2013.4.0

→ ./configure && make install

/etc/modules:

```
batman-adv
```

Gateway

/etc/default/fastd:

```
AUTOSTART="none"
```

/etc/fff_fuerth_fastd.sh:

%todo!

Gateway

/etc/network/interfaces:

```
auto ffffuerthVPN
iface ffffuerthVPN inet manual
    post-up /usr/local/sbin/batctl -m bat1 if add $IFACE
    post-up ifconfig $IFACE up
    post-up ifup bat1
    post-down ifdown bat1
    post-down ifconfig $IFACE down
```

Gateway

/etc/network/interfaces:

```
auto bat1
iface bat1 inet manual
    post-up ifconfig $IFACE up
    post-up ip addr add 10.50.32.2/21 dev $IFACE
    post-up ip rule add iif $IFACE table fff
    post-up ip rule add from 10.50.32.0/21 table fff
    post-up ip rule add to 10.50.32.0/21 table fff
    post-up ip route add 10.50.32.0/21 dev $IFACE table fff
    post-up invoke-rc.d isc-dhcp-server restart
    post-down ip route del 10.50.32.0/21 dev $IFACE table fff
    post-down ip rule del from 10.50.32.0/21 table fff
    post-down ip rule del to 10.50.32.0/21 table fff
    post-down ip rule del iif $IFACE table fff
    post-down ifconfig $IFACE down
```

Gateway

/etc/network/interfaces:

```
auto ro1
iface ro1 inet static
    address 192.168.230.2
    pre-up iptunnel add ro1 mode gre \
            local 193.192.41.129 \
            remote 109.163.229.254 ttl 255
    up ifconfig ro1 multicast
    pointopoint 192.168.230.1
    post-down iptunnel del ro1
```

Gateway

/etc/olsrd/olsrd.conf:

```
Interface "ro1" {
    Ip4Broadcast 255.255.255.255
    LinkQualityMult 192.168.230.2 0.5
}

Hna4 {
    10.50.32.0 255.255.248.0
}

LoadPlugin "olsrd_httpinfo.so.0.1" {
        PlParam "Port" "8080"
        PlParam "Net" "0.0.0.0 0.0.0.0"
}

RtProto 8
RtTable 10
RtTableDefault 10
RtTableTunnel 10
```

Netmon

- Nodewatcher
  - Generiert Status-Daten
- Configurator
  - Verknüpft Netmon und Knoten
- Crawler
  - Sammelt Status-Daten
- Netmon
  - Visualisiert Status-Daten

Nodewatcher

- Erzeugt XML Datei
- Läuft alle 5 Minuten auf den Knoten
- node.data über Webinterface downloadbar

Configurator

- Knoten kennt Netmon's Link-Local Adresse
- Knoten meldet alle 5 Minuten seine MAC Adresse ans Netmon
- Netmon meldet dabei zurück, dass der Knoten noch nicht eingetragen wurde
- Benutzer „übernimmt" Knoten im Netmon
- (Benutzer gibt dem Knoten einen Namen)
- Knoten meldet wieder seine MAC an Netmon
- Netmon meldet router_id, update_hash und api_key zurück
- Knoten trägt seine Link-Local Adresse im Netmon ein
- Netmon pollt einmal alle 10 Minuten nach Router-Daten
- Knoten pollt alle 5 Minuten nach seinem Hostname

Crawler

Pollt alle Router jeweils alle 10 Minuten

- Jeweils 20 Knoten nacheinander
- Ping (mit PsExecute für Timeout)
  - Bohrt neigh Tabelle auf
  - Eliminiert initial Timeout
- curl'ed node.data
- Dekodiert xml
- Füllt MySQL Tabellen

Netmon

- Wie sieht Netmon aus?
  → `http://netmon.freifunk-franken.de`

Ende

Vielen Dank für Eure Aufmerksamkeit ...