Freifunk S"u+00f"dholstein Documentation

Release 1.3

Grotax, ffsh

30.05.2020

Inhalte

KAPITEL 1

Infrastruktur

1.1 Netzwerk

Network IPv4: `10.144.0.0/16`

Network IPv6: `fddf:0bf7:80::/48`

1.2 Gateways

Name	ULA	IPv6	RFC1918	DHCP	ICVPN-Transit	Mesh MAC(s)	B.A.T.M.A.N. adv. MAC(s)	Standort	Betreuer	Exit/VPN-Dienst	Status
Barnitz	fddf:0bf7:801:48::1	10.144.48.1	10.144.48.2-10.144.63.254	n/a	00:5b:27:80:00:48	00:5b:27:81:00:48	Hetzner (Nbg)	ul	direkt	online
Beste	fddf:0bf7:801:64::1	10.144.64.1	10.144.64.2-10.144.79.254	n/a	00:5b:27:80:00:64	00:5b:27:81:00:64	Hetzner (Fsn)	ul	direkt	online
Bille	fddf:0bf7:801:80::1	10.144.80.1	10.144.80.2-10.144.95.254	n/a	00:5b:27:80:00:80	00:5b:27:81:00:80	Neocup	fr	direkt	online
Brunsbach	fddf:0bf7:801:96::1	10.144.96.1	10.144.96.2-10.144.111.254	n/a	00:5b:27:80:00:96	00:5b:27:81:00:96				n/a
Heilsau	fddf:0bf7:801:112::1	10.144.112.1	10.144.112.2-10.144.127.254	n/a	00:5b:27:80:01:12	00:5b:27:81:01:12	Hetzner (Hel)	ul	direkt	online
Hopfenbach	fddf:0bf7:801:128::1	10.144.128.1	10.144.128.2-10.144.143.254	n/a	00:5b:27:80:01:28	00:5b:27:81:01:28				n/a
Krummbach	fddf:0bf7:801:144::1	10.144.144.1	10.144.144.2-10.144.159.254	n/a	00:5b:27:80:01:44	00:5b:27:81:01:44	Hetzner(fsn)	ks	direkt	online
Piependbek	fddf:0bf7:801:160::1	10.144.160.1	10.144.160.2-10.144.175.254	n/a	00:5b:27:80:01:60	00:5b:27:81:01:60	Hetzner(hel)	swo	direkt	online
Strusbek	fddf:0bf7:801:176::1	10.144.176.1	10.144.176.2-10.144.191.254	n/a	00:5b:27:80:01:76	00:5b:27:81:01:76				n/a
Sylsbek	fddf:0bf7:801:192::1	10.144.192.1	10.144.192.2-10.144.207.254	n/a	00:5b:27:80:01:92	00:5b:27:81:01:92	Neocup	ul	direkt	online
Trave	fddf:0bf7:801:208::1	10.144.208.1	10.144.208.2-10.144.223.254	n/a	00:5b:27:80:02:08	00:5b:27:81:02:08	Hetzner (Fsn)	ul	direkt	online
Viehbach	fddf:0bf7:801:224::1	10.144.224.1	10.144.224.2-10.144.239.254	n/a	00:5b:27:80:02:24	00:5b:27:81:02:24	Hetzner(fsn)	ks	Mullvad 1 / direkt	offline

1.3 Karte

Die Karte kann unter https://map.freifunk-suedholstein.de erreicht werden. Die Karte wird auf dem Gateway Hopfenbach betrieben. Sie basiert auf dem meshviewer von Freifunk Regensburg. Unsere Konfiguration findet man in unserem fork auf GitHub.

Unsere Grafana instanz ist unter https://map.freifunk-suedholstein.de/grafana erreichbar.

```
# meshviewer.json
https://map.freifunk-suedholstein.de/data/meshviewer.json
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# nodes.json (v2)
https://map.freifunk-suedholstein.de/data/nodes.json
# nodelist.json
https://map.freifunk-suedholstein.de/data/nodelist.json
```

1.4 Jenkins

Unsere Firmware wird mit Jenkins gebaut die Jenkins Instanz ist unter https://jenkins.grotax.de erreichbar und wird von Grotax betrieben. Die Images werden automatisch unter https://firmware.grotax.de veröffentlicht. Eine offizielle Veröffentlichung auf den Firmware-Servern erfolgt erst nach manueller Prüfung und Signatur.

1.5 GitHub

Fast alle Daten, welche für den Betrieb notwendig sind, werden unter https://github.com/ffsh gespeichert.

KAPITEL 2

Gateway Konfiguration

2.1 Allgemeines

Diese Anleitung ist auf Debian 10 ausgerichtet

Es wird neben dem OS nicht viel Plattenplatz auf dem System benötigt. In Vorbereitung zur Installation wird in dieser Anleitung von folgendem ausgegangen:

- Einen Hardware-Server oder ein virtualisiertes System auf dem man Kernel-Module kompilieren und laden kann (z.b. kvm)
- default-Installation von Debian 10
- Einen User auf dem Server mit sudo-Rechten

Alle Befehle werden als der User ausgeführt wenn dies nicht explizit abweichend angegeben ist. In den Beispielen benutzen wir den als Editor `nano`. Es funktioniert natürlich auch jeder andere Texteditor.

2.2 Hinweise zur Nutzung

Diese Dokumentation soll (auch) eine vollständige Anleitung sein um ein Gateway aufzusetzen. Daher

- Ist das gesamte Dokument in der Reihenfolge aufgebaut, in der die einzelnen Elemente benötigt werden
- Es sollten sich alle Befehle per Copy&Paste aus diesem Dokument in die Shell übernehmen lassen, wobei natürlich die individuellen Anpassungen noch von Hand vorgenommen werden müssen. Sollte das der Fall sein wird jeweils auf den Aufruf des Editors hingewiesen.

Um den zweiten Schritt zu ermöglichen wurde ein kleiner „Hack“ angewandt. Da unser Anliegen natürlich immer auch Erkenntnisgewinn ist wollen wir das hier näher erläutern. Grundsätzlich haben alle Kommandos auf der Bash eine Standard-Eingabe und eine Standard-Ausgabe, die man aber auch manipulieren kann. Wir nutzen das hier mit dem Befehl `cat`. cat macht erst mal nichts anderes als eine Datei zeilenweise zu lesen und auf die Standard-Ausgabe auszugeben. Diese Standard-Ausgabe ist der Bildschirm.

```
cat README.txt
```

Macht also nicht anderes, als die Datei README.txt auf den Bildschirm auszugeben. Diese Ausgabe kann man über > umlenken.

```
cat README.txt > WRITEME.txt
```

Liest den Inhalt aus README.txt und schreibt ihn nach WRITEME.txt. Wichtig ist: Der Befehl überschreibt das Ziel ohne Rückfrage. Wenn es WRITEME.txt also vorher gegeben haben wollte wäre der Inhalt in dem Beispiel komplett überschrieben. Wenn man was an eine bestehende Datei anhängen will nutzt man >>.

```
cat README.txt >> WRITEME.txt
```

Hängt also den Inhalt von README.txt an WRITEME.txt an. Achtung! Wir nutzen hier beide Möglichkeiten, das ist also Absicht ob da 1 oder 2 > verwendet werden.

Ebenso kann man über < auch den Standard-Input verbiegen. Das ist für unseren Anwendungsfall wichtig da wir ja in den meisten Fällen auf dem System gar keine Dateien haben die wir nutzen wollen sondern die ja erst aus dieser Dokumentation erzeugen wollen.

```
cat < README.txt
```

Macht im Prinzip das Selbe wie „cat README.txt“. Im Beispiel ohne „<“ wird der Dateiname als Parameter an des Kommando übergeben. Der Parameter besagt „Lies den Standard-Input aus der Datei mit diesem Namen“ in der Variante mit „<“ erhält der Befehl keinen Parameter, allerdings verbiegt „<“ den Standard-Input von der Tastatur auf die Datei README.txt. Wer das verifizieren möchte kann mal `cat` ohne jeden Parameter ausführen. Dann wird cat starten und Daten vom Standard-Input (=Tastatur) erwarten und an den Standard-Output (=Bildschirm) weiterreichen. Ctrl-C beendet den Befehl.

Dazu nutzen wir eine Konstruktion namens „Heredoc“ bzw. „Herestring“. Dies erlaubt es uns, über die Shell auch mehrzeiligen Text an ein Programm zu übergeben. Kurz gefasst: Hier wird ein Ende-Zeichen angegeben und aller Text bis zu diesem Ende-Zeichen wird an das Programm übergeben. Das sieht dann so aus:

```
cat << EOF > WRITEME.txt
Dies wird die erste Zeile
Dies wird die zweite Zeile
EOF
```

Hier wird „cat“ auf der Standard-Eingabe der folgende Text übergeben und als Standard-Ausgabe wird die Datei WRITEME.txt angegeben.

Einen ziemlich guten Guide, was man mit Stanard-Eingabe und Standard-Ausgabe auf der Shell anstellen kann findet man z.B. unter http://mywiki.wooledge.org/BashGuide/InputAndOutput

Wer sich in der Anleitung nur für den Inhalt der jeweiligen Datei interessiert kann also einfach alles zwischen der „cat-Zeile und dem „EOF“ nehmen und z.B. per Copy&Paste in seinen Lieblings-Editor übertragen.

2.3 Installation der Debain-Pakete

Zunächst installieren wir mal alle Pakete die generell benötigt werden

- eine Build-Umgebung (build-essentials)
- Git (git)
- https-Support für apt (apt-transport-https)

- Zeitsynchronisation (ntp)
- Netzwerk-Tools (bridge-utils, net-tools)

```
sudo apt install build-essential git apt-transport-https ntp bridge-utils ntp net-
↪tools
```

2.4 Netzwerk Konfiguration

2.4.1 Interfaces Konfigurieren

Nun kommt das eigentlich wichtigste. Das Netzwerk muss eingerichtet werden, so das die einzelnen Schnittstelle bereitstehen und eine Art Brücke vom Freifunknetz in das Internet aufbauen.

Hinweis: diese Konfiguration ist allgemeingültig für unser Netz. Daher ist das jeweilige Gateway in den IP-Adressen mit `[GW Netz]` geschrieben. Diese Nummer muss natürlich durchgängig gleich sein, da sonst nichts funktionieren wird!

Interfaces werden in `/etc/network/interfaces` definiert. Es gibt aber die Möglichkeit, Konfigurationen aus eigenen Dateien zu sourcen. Um hier die Konfigurationen sauber zu halten lassen wir die defaults unangetastet. In :code: */etc/network/interfaces* sind unter Debian 10 keine Interfaces mehr angegeben, sie werden aus Dateien in /etc/network/interfaces.d/ gelesen. Hier sollte es eine Datei geben, in der z.B. loopback und eth0 definiert sind. Die fassen wir nicht an, wir erzeugen eine eigene Datei :code:‘/etc/network/interfaces.d/60-ffsh-init.cfg ‘

```
sudo cat << EOF > /etc/network/interfaces.d/60-ffsh-init.cfg
# Netwerkbruecke fuer Freifunk
# - Hier laeuft der Traffic von den einzelnen Routern und dem externen VPN zusammen
# - Unter der hier konfigurierten IP ist der Server selber im Freifunk Netz erreichbar
# - bridge_ports none sorgt dafuer, dass die Bruecke auch ohne Interface erstellt wird

auto br-ffsh
iface br-ffsh inet static
    address 10.144.[GW Netz].1
    netmask 255.255.0.0
    bridge_ports none

iface br-ffsh inet6 static
    address fddf:0bf7:80::[GW Netz]:1
    netmask 64


    post-up /sbin/ip rule add iif br-ffsh table 42
    pre-down /sbin/ip rule del iif br-ffsh table 42

# B.A.T.M.A.NAdvanced Advanced Interface
# - Erstellt das virtuelle Inteface fuer das B.A.T.M.A.N Advanced-Modul und bindet␣
↪dieses an die Netzwerkbruecke
# - Die unten angelegte Routing-Tabelle wird spaeter fuer das Routing innerhalb von␣
↪Freifunk (Router/VPN) verwendet
#
# Nachdem das Interface gestartet ist, wird eine IP-Regel angelegt, die besagt, dass␣
↪alle Pakete, die über das bat0-Interface eingehen,
# und mit 0x1 markiert sind, über die Routing-Tabelle 42 geleitet werden.
# Dies ist wichtig, damit die Pakete aus dem Mesh wirklich über das VPN raus gehen.
#
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
allow-hotplug bat0
iface bat0 inet6 manual
    pre-up batctl if add ffsh-mesh
    post-up ip link set address 00:5b:27:81:0:[GW Netz] dev bat0
    post-up ip link set dev bat0 up
    post-up brctl addif br-ffsh bat0
    post-up batctl it 10000
    post-up batctl gw server 100mbit/100mbit

    post-up ip rule add from all fwmark 0x1 table 42

    pre-down brctl delif br-ffsh bat0 || true
    down ip link set dev bat0 down
EOF
```

Und hinterher nicht vergessen, die Platzhalter zu ersetzen

```
sudo nano /etc/network/interfaces.d/60-ffsh-init.cfg
```

Die `/etc/hosts` mit Folgenden Zeilen ergänzen:

```
sudo cat << EOF >> /etc/hosts

[externe IP]                [GW Name].freifunk-suedholstein.de    [GW Name]
10.144.[GW Netz].1          ffsh
fddf:0bf7:80::[GW Netz]:1   ffsh
```

Und hinterher nicht vergessen, die Platzhalter zu ersetzen

```
sudo nano /etc/hosts
```

2.4.2 IP Forwarding

In der Konfigurationsdatei `/etc/sysctl.d/forwarding.conf` bitte die folgenden Zeilen eintragen, damit das IP-Forwarding für IPv4 und IPv6 laufen:

```
sudo cat << EOF > /etc/sysctl.d/forwarding.conf
# IPv4 Forwarding
net.ipv4.ip_forward=1

# IPv6 Forwarding
net.ipv6.conf.all.forwarding = 1
EOF
```

2.4.3 IP Tables

Lege die Konfigurationsdatei `/etc/iptables.up.rules` an mit Folgendem:

Damit werden alle Pakete, die über die Bridge rein kommen, mit dem 0x1-Flag markiert, und damit über Routing-Tabelle 42 geschickt. Es gibt noch 2 Regeln für DNS, dass auch DNS-Pakete (Port 53 TCP/UDP) über die Tabelle 42 geschickt werden.

```
sudo cat << EOF > /etc/iptables.up.rules
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
*mangle
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
COMMIT
EOF
```

Nun müssen die IP-Tables geladen werden. Bitte erstellt die Datei `/etc/network/if-pre-up.d/iptables` mit folgenden Zeilen:

```
sudo cat << EOF > /etc/network/if-pre-up.d/iptables
#!/bin/sh
/sbin/iptables-restore < /etc/iptables.up.rules
EOF
```

Bitte nun noch eine Datei `/etc/fastd/ffsh/iptables\_ffsh.sh` erstellen, die alle Routing iptables Vorgaben enthält:

```
sudo cat << EOF > /etc/fastd/ffsh/iptables\_ffsh.sh
#!/bin/sh
/sbin/ip route add default via [EXTERNE-IPv4] table 42
/sbin/ip route add 10.144.0.0/16 dev br-ffsh src 10.144.[GW Netz].1 table 42
/sbin/ip route add 0/1 dev tun0 table 42
/sbin/ip route add 128/1 dev tun0 table 42
/sbin/ip route del default via [EXTERNE-IPv4] table 42
/sbin/iptables -t nat -D POSTROUTING -s 0/0 -d 0/0 -j MASQUERADE > /dev/null 2>&1
/sbin/iptables -t nat -I POSTROUTING -s 0/0 -d 0/0 -j MASQUERADE
/sbin/iptables -t nat -D POSTROUTING -s 0/0 -d 0/0 -o tun0 -j MASQUERADE > /dev/null␣
↪2>&1
/sbin/iptables -t mangle -D PREROUTING -s 10.144.[GW Netz].0/20 -j MARK --set-mark␣
↪0x1 > /dev/null 2>&1
/sbin/iptables -t mangle -I PREROUTING -s 10.144.[GW Netz].0/20 -j MARK --set-mark 0x1
/sbin/iptables -t mangle -D OUTPUT -s 10.144.[GW Netz].0/20 -j MARK --set-mark 0x1 > /
↪dev/null 2>&1
/sbin/iptables -t mangle -I OUTPUT -s 10.144.[GW Netz].0/20 -j MARK --set-mark 0x1
# IGMP/MLD segmentation
echo 2 > /sys/class/net/bat0/brport/multicast_router
EOF
```

Jetzt müssen die für Linux ausführbar werden.

```
sudo chmod +x /etc/network/if-pre-up.d/iptables
sudo chmod +x /etc/fastd/ffsh/iptables_ffsh.sh
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
sudo iptables-restore < /etc/iptables.up.rules
```

2.5 B.A.T.M.A.N Advanced und Fastd

B.A.T.M.A.N Advanced ist das in Südholstein verwendete Routing Verfahren. B.A.T.M.A.N Advanced benötigt ein Kernel Modul und batclt.

Damit B.A.T.M.A.N Advanced bei einem Kernel Update nicht verschwindet oder durch die alte OS-Version ersetzt wird, richten wir das Modul mit `dkms` ein.

2.5.1 B.A.T.M.A.N Advanced Kernel Modul und batctl

Zur Sicherheit prüfen wir erst mal ob es ein geladenes B.A.T.M.A.N Advanced-Modul gibt:

```
lsmod | grep batman
```

Sollte das der Fall sein einmel entladen, wir wollen unser eigened laden:

```
modprobe -rf batman_adv
```

Dann können wir die Pakete installieren die wir für das Bauen des Kernel-Moduls benötigen:

```
sudo apt install linux-headers-amd64 libnl-3-dev libnl-genl-3-dev libcap-dev pkg-
↪config dkms
```

Und nun widmen wir uns dem eigentlichen B.A.T.M.A.N Advanced - Modul.

Erst mal die notwendigen Sourcen runterladen und entpacken:

```
sudo su -
cd /usr/src
wget https://downloads.open-mesh.org/batman/releases/batman-adv-2020.1/batman-adv-
↪2020.1.tar.gz
tar xzfv batman-adv-2020.1.tar.gz

wget https://downloads.open-mesh.org/batman/releases/batman-adv-2020.1/batctl-2020.1.
↪tar.gz
tar xzvf batctl-2020.1.tar.gz

exit
```

Nun wird das B.A.T.M.A.N Advanced-Modul gebaut. Dafür müssen wir uns erst mal eine :code: *dkms.conf* zusammenbauen

```
sudo cat << EOF > /usr/src/batman-adv-2020.1/dkms.conf
PACKAGE_NAME=batman-adv
PACKAGE_VERSION=2020.1

DEST_MODULE_LOCATION=/extra
BUILT_MODULE_NAME=batman-adv
BUILT_MODULE_LOCATION=net/batman-adv
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
MAKE="'make' CONFIG_BATMAN_ADV_BATMAN_V=n"
CLEAN="'make' clean"

AUTOINSTALL="yes"
EOF
```

Und nun können die Module gebaut und werden:

```
sudo dkms add -m batman-adv -v 2020.1
sudo dkms build -m batman-adv -v 2020.1
sudo dkms install -m batman-adv -v 2020.1
```

Und das dazugehoerende Control- und Management-Tool:

```
cd /usr/src/batctl-2020.1/
make
sudo make install
```

Damit B.A.T.M.A.N Advanced bei jedem Neustart auch geladen wird muss er nun noch in /etc/modules hinterlegt werden:

```
sudo cat << EOF >> /etc/modules
batman_adv
EOF
```

2.6 fastd

fastd v18 ist in Debian 9 und 10 bereits in den Repositories enthalten. Unter Debian 8 findet man es in den jessie-backports.

```
sudo apt install fastd
```

2.6.1 fastd-Konfiguration

Wir brauchen für den neuen Server die Schlüssel für fastd. Diese sind in Südholstein für 12 Gateways bereits in der Firmware eingetragen und den privaten Schlüssel gibt es über das NOC-Team (noc@freifunk-suedholstein.de).

Im Folgenden wird der sichere private Schlüssel als [SERVER-SECRET-KEY] aufgeführt und müssen durch die erzeugten Schlüssel sinnvoll ersetzt werden!

Wir benötigen einen System-User, als der fastd ausgeführt werden kann. Dafür versorgen wir :code: *adduser* mit folgenden Parametern:

- **--system** Es handelt sich um einen Systemuser
- **--no-create-home** Es soll kein Homeverzeichnis angelegt werden
- **--disabled-password** Der User hat kein Passwort
- **--disabled-login** Anmelden ist nicht möglich
- –home /nonexistent Das Home-Verzeichnis gibt es wirklich nicht

```
sudo adduser --system --no-create-home --disabled-password --disabled-login --home /
↪nonexistent ffsh
```

Es ist eine Konfigurationsdatei für fastd notwendig. In der folgenden Konfiguration bitte die `[EXTERNE-IPv4]` durch die echte IP vom Server ersetzen. Wenn es auch eine IPv6 gibt, kann die entsprechende Zeile aktiviert werden und benötigt die echte IPv6 `[EXTERNE-IPv6]`.

```
sudo cat << EOF > /etc/fastd/ffsh/fastd.conf
# Bind to a fixed address and port, IPv4 and IPv6 at Port 1234
bind any:10000 interface "eth0";
# bind [EXTERNE-IPv6]:1234 interface "eth0";

# Set the user, fastd will work as
user "ffsh";

# Set the interface name
interface "ffsh-mesh";

# Set the mode, the interface will work as
mode tap;

# Set the mtu of the interface (salsa2012 with ipv6 will need 1406)
mtu 1426;

# Set the methods (aes128-gcm preferred, salsa2012+umac preferred for nodes)
method "null";
method "salsa2012+umac";

#hide ip addresses yes;
#hide mac addresses yes:

# Secret key generated by `fastd --generate-key`
secret "[SERVER-SECRET-KEY]";

# Log everything to syslog
log to syslog level debug;

# Include peers from our git-repos
#include peers from "peers/"; #optional eigene peers anlegen zb den eigenen toaster␣
↪mit fastd oder so
include peers from "gateways/"; #git repo klonen in /etc/fastd/ffsh/ git clone ␣
↪https://github.com/ffsh/gateways.git

# Configure a shell command that is run on connection attempts by unknown peers (true␣
↪means, all attempts are accepted)
on verify "true";
# on verify "/etc/fastd/fastd-blacklist.sh $PEER_KEY";

# Configure a shell command that is run when fastd comes up
on up "
 ip link set dev $INTERFACE address 00:5b:27:80:0X:XX           # X für das GW Netz,␣
↪zB 2:24 für 10.144.224.0/20
 ip link set dev $INTERFACE up
 ifup bat0
 sh /etc/fastd/ffsh/iptables_ffsh.sh
";

on down "
 ifdown bat0
";
EOF
```

Und noch die Variablen ersetzen:

```
sudo nano /etc/fastd/ffsh/fastd.conf
```

Das Beste ist, wenn man nun die fastd-Konfiguration mal überprüft. Vorher muss der Server neugestartet werden, damit die vorher durchgeführten Anpassungen auch Wirkung zeigen :-)

```
sudo reboot
```

Dann auf der Konsole mit folgender Zeile die fastd Einstellungen prüfen:

```
sudo fastd -c /etc/fastd/ffsh/fastd.conf
```

Bei der Installation von :code: *GW_Bille* ist fastd nicht automatisch gestartet. Er muss als Dienst in systemd hinterlegt werden. Mangels Wissen (und es war schon spät) haben wir uns mit einem Workaround beholfen, sobald wir den korrekten Weg gefunden haben werden wir das auch hier korrigieren

```
sudo nano /etc/default/fastd
```

Dort gibt es einen Parameter :code: *AUTOSTART="none"*, der muss auskommentiert werden. Damit greift der default `all`:

```
# This is the configuration file for /etc/init.d/fastd

#
# This configuration file is DEPRECATED! Please set autostart to "none" in
# this file and use the instanced systemd unit fastd@.service
#

#
# Start only these VPNs automatically via init script.
# Allowed values are "all", "none" or space separated list of
# names of the VPNs. If empty, "all" is assumed.
#
#AUTOSTART="none"
```

Wenn das erfolgreich war, kann nun fastd eingeschaltet und gestartet werden:

```
sudo systemctl enable fastd
sudo systemctl start fastd
```

2.6.2 Zusätzliche Sicherheit - derzeit nicht genutzt !

Wichtig: In der Konfiguration wird jeder Router reingelassen. Das mag nicht jeder, aber es vereinfacht die Integration der Router und damit auch die Verteilung. Wenn man das nicht möchte, müsste jeder Router separat mit seinem öffentlichen Schlüssel unter `.../peers/` hinterlegt werden. Auskommentiert ist eine Zeile bei `on verify` die eine Blacklist führt. Damit kann man unliebsame Genossen aussperren.

```
sudo cat << EOF > /etc/fastd/fastd-blacklist.sh
#!/bin/bash
PEER_KEY=$1
if /bin/grep -Fq $PEER_KEY /etc/fastd/fastd-blacklist.json; then
    exit 1
else
    exit 0
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
fi
EOF
```

Und da es sich um ein Skript handelt müssen wir es auch ausführbar machen:

```
sudo chmod +x /etc/fastd/fastd-blacklist.sh
```

Wie die weiteren Dateien mit der Blacklist aussehen, findet man unter diesem Link <*https://github.com/ffruhr/fastdbl*>

2.7 DHCP

Jetzt brauchen wir noch den dhcp innerhalb des FFSH-Netzes

```
sudo apt install radvd isc-dhcp-server
```

2.7.1 DHCP radvd IPv6

Es wird für IPv6 die Konfigurationsdatei `/etc/radvd.conf`

```
sudo cat << EOF > /etc/radvd.conf
interface br-ffsh {
    AdvSendAdvert on;
    IgnoreIfMissing on;
    AdvManagedFlag off;
    AdvOtherConfigFlag on;
    MaxRtrAdvInterval 200;
    AdvLinkMTU 1280;
    prefix fddf:0bf7:80::/64 {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
    };

    RDNSS fddf:0bf7:80::[GW Netz]:1 {
    };
};
EOF
```

Und die Variablen ersetzen

```
sudo nano /etc/radvd.conf
```

Jetzt kann radvd gestartet werden:

```
sudo service radvd restart
```

2.7.2 DHCP isc-dhcp-server IPv4 und IPv6

Die Konfigurationsdatei `/etc/dhcp/dhcpd.conf` wird für IPv4 mit folgenden Zeilen benötigt:

```
sudo cat << EOF > /etc/dhcp/dhcpd.conf
ddns-update-style none;
option domain-name ".ffsh";

# möglichst kurze Leasetime
default-lease-time 120;
max-lease-time 600;

log-facility local7;

subnet 10.144.0.0 netmask 255.255.0.0 {
    authoritative;
    range 10.144.[GW Netz].2 10.144.[GW Netz + 15].254;

    option routers 10.144.[GW Netz].1;

    option domain-name-servers 10.144.[GW Netz].1; # für die eigenen DNS-Einträge
    # option domain-name-servers 85.214.20.141; # weitere anonyme DNS
    # option domain-name-servers 213.73.91.35;
}

include "/etc/dhcp/static.conf";
```

Und wieder die Variablen anpassen:

```
sudo nano /etc/dhcp/dhcpd.conf
```

Bitte eine leere Datei `/etc/dhcp/static.conf` erzeugen.

```
sudo echo > /etc/dhcp/static.conf
```

Und den User dhcpstatic anlegen

```
sudo useradd -m -s /bin/bash dhcpstatic
```

Und als der User die Static aus dem Git laden

```
sudo su - dhcpstatic

git clone https://github.com/ffsh/dhcp-static.git

chmod +x dhcp-static/updateStatics.sh

exit
```

Und einmal mit dem auch aus dem Git gezogenen Befehl den dhcpd aktualisieren

```
sudo /home/dhcpstatic/dhcp-static/updateStatics.sh
```

Und über cron dafür sorgen dass das regelmäßig passiert.

```
sudo cat << EOF > /etc/cron.d/ffsh_dhcpstatic
*/5 * * * * root /home/dhcpstatic/dhcp-static/updateStatics.sh > /dev/null 2>&1
EOF
```

Auf dem DHCP-Server muss noch das Bridge-Interface für IPv4 festgelegt werden. Bitte die Datei `/etc/default/isc-dhcp-server` mit folgender Option ergänzen:

```
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="br-ffsh"
INTERFACESv6=""
```

Am Besten wird der DHCP-Server vor dem Start und Betrieb noch mal geprüft. Bitte vorher den Server rebooten

```
sudo reboot
```

und dann auf der Konsole folgende Zeile ausführen:

```
sudo dhcpd -f -d
```

War das erfolgreich, so kann der DHCP-Server als root gestartet werden:

```
sudo systemctl restart isc-dhcp-server
```

2.8 DNS-Server (BIND)

```
apt install bind9
```

Für das interne Freifunknetz ist nun noch der DNS-Server bind9 mit den Konfigurationsdateien wie folgt zu konfigurieren:

Erstmal diese Datei `/etc/bind/named.conf.options`

```
sudo cat << EOF > /etc/bind/named.conf.options
options {
    directory "/var/cache/bind";
    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk.  See http://www.kb.cert.org/vuls/id/800113
    // If your ISP provided one or more IP addresses for stable
    // nameservers, you probably want to use them as forwarders.
    // Uncomment the following block, and insert the addresses replacing
    // the all-0's placeholder.
    forwarders {
        8.8.8.8;
        8.8.4.4;
    };
    //========================================================================
    // If BIND logs error messages about the root key being expired,
    // you will need to update your keys.  See https://www.isc.org/bind-keys
    //========================================================================
    // dnssec-enable yes;
    // dnssec-validation yes;
    dnssec-validation no;
    // dnssec-lookaside auto;
    // recursion yes;
    // allow-recursion { localnets; localhost; };
    auth-nxdomain no;    # conform to RFC1035
    listen-on-v6 { any; };
};
EOF
```

Dann in der Datei `/etc/bind/named.conf.local` folgendes am Ende ergänzen:

```
sudo cat << EOF >> /etc/bind/named.conf.local
// Do any local configuration here
// Consider adding the 1918 zones here, if they are not used in your organization

include "/etc/bind/zones.rfc1918";

zone "stormarn.freifunk.net" {
       type master;
       file "/etc/bind/db.net.freifunk.stormarn";
  };

zone "freifunk-stormarn.de" {
       type master;
       file "/etc/bind/db.de.freifunk-stormarn";
  };

zone "lauenburg.freifunk.net" {
       type master;
       file "/etc/bind/db.net.freifunk.lauenburg";
  };

zone "freifunk-lauenburg.de" {
       type master;
       file "/etc/bind/db.de.freifunk-lauenburg";
  };

zone "freifunk-suedholstein.de" {
        type master;
        file "/etc/bind/db.de.freifunk-suedholstein";
};

zone "ffshev.de" {
     type master;
     file "/etc/bind/db.de.ffshev";
};
EOF
```

Die zugehörigen Zone Dateien werden in einem Repository verwaltet. Diese sollen automatisch aktualisiert werden.

Als erstes legen wir einen neuen Benutzer an.

```
sudo useradd -m -s /bin/bash dnsbind
```

Dann wechseln wir zu diesem Nutzer.

```
sudo su - dnsbind
```

Und klonen das Repository

```
git clone https://github.com/ffsh/bind.git
```

Danach verlassen wir den Nutzer.

```
exit
```

Und legen einige Cron jobs an.

```
sudo cat << EOF > /etc/cron.d/ffsh_dnsbind
*/15 * * * * root /home/dnsbind/bind/updatestofrei.sh > /dev/null 2>&1
*/15 * * * * root /home/dnsbind/bind/updatelauen.sh > /dev/null 2>&1
*/15 * * * * root /home/dnsbind/bind/updateffsh.sh > /dev/null 2>&1
EOF
```

Zum Schluss starten wir bind neu.

```
sudo systemctl restart bind9
```

2.9 Mesh Announce

Um als Gateway, Server oder alles was kein Freifunk Router ist auf der Karte zu erscheinen kann mesh-announce installiert werden.

Dafür müssen folgende Dinge vorhanden sein: lsb_release, ethtool, python3 (>= 3.3)

```
sudo apt install ethtool python3
```

Mesh Announce kann auch im alfred Stil Daten broadcasten das wollen wir aber nicht.

```
sudo git clone https://github.com/ffnord/mesh-announce /opt/mesh-announce
sudo cp /opt/mesh-announce/respondd.service /etc/systemd/system/respondd.service
nano /etc/systemd/system/respondd.service
```

Den Systemd Service passen wir jetzt an unser Netzwerk und Gateway an. Erstmal das Konzept. Wir starten respondd.py mit einigen argumenten:

```
respondd.py -d /opt/mesh-announce/providers -i <your-clientbridge-if> -i <your-mesh-
↪vpn-if> -b <your-batman-if> -m <mesh ipv4 address>
```

```
your-clientbridge-if - br-ffsh
your-mesh-vpn-if     - ffsh-mesh
your-batman-if       - bat0
mesh ipv4 address    - GW-IPV4
```

Im folgenden Beispiel ist Hopfenbach das Gateway dort sind die Interfaces so wie in der Anleitung benannt und die IP ist `10.144.128.1`.

```
[Unit]
Description=Respondd
After=network.target

[Service]
ExecStart=/opt/mesh-announce/respondd.py -d /opt/mesh-announce/providers -i br-ffsh -
↪i ffsh-mesh -b bat0 -m 10.144.128.1
Restart=always
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

[Install]
WantedBy=multi-user.target
```

Wir erzeugen die Datei `/etc/systemd/system/respondd.service`

```
sudo echo << EOF > /etc/systemd/system/respondd.service
[Unit]
Description=Respondd
After=network.target

[Service]
ExecStart=/opt/mesh-announce/respondd.py -d /opt/mesh-announce/providers -i br-ffsh -
↪i ffsh-mesh -b bat0 -m 10.144.[GW Netz].1
Restart=always
Environment=PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

[Install]
WantedBy=multi-user.target
```

Und passen das GW-Netz an:

```
sudo nano /etc/systemd/system/respondd.service
```

Dann mit `hostname` prüfen ob der erwünschte Gateway-Name eingetragen ist. Wenn dem so ist kann der nächste Schritt übersprungen werden. Andernfalls (oder zur Sicherheit) hinterlegen wir es an passender Stelle:

```
sudo echo << EOF > /opt/mesh-announce/providers/nodeinfo/hostname.py
import providers
import socket
class Source(providers.DataSource):
    def call(self):
        return "[GW hostname]"
EOF
```

Und den gewünschten Hostname eintragen

```
sudo nano /opt/mesh-announce/providers/nodeinfo/hostname.py
```

Dann den Service aktivieren

```
sudo systemctl daemon-reload
sudo systemctl start respondd
sudo systemctl enable respondd
```

Das System sollte in kürze auf der Karte auftauchen, je nachdem wie der Kartenserver konfiguriert ist.

2.10 Munin

Damit das Gateway auch in den Statistiken unter http://stats.freifunk-suedholstein.de/ auftauch den Munin Node installieren

```
sudo apt install munin-node
```

und die :code: */etc/munin/munin-node.conf* anpassen.

```
cat << EOF > `/etc/munin/munin-node.conf
#
# Example config-file for munin-node
#
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
log_level 4
log_file /var/log/munin/munin-node.log
pid_file /var/run/munin/munin-node.pid

background 1
setsid 1

user root
group root

# This is the timeout for the whole transaction.
# Units are in sec. Default is 15 min
#
# global_timeout 900

# This is the timeout for each plugin.
# Units are in sec. Default is 1 min
#
# timeout 60

# Regexps for files to ignore
ignore_file [\#~]$
ignore_file DEADJOE$
ignore_file \.bak$
ignore_file %$
ignore_file \.dpkg-(tmp|new|old|dist)$
ignore_file \.rpm(save|new)$
ignore_file \.pod$

# Set this if the client doesn't report the correct hostname when
# telnetting to localhost, port 4949
#
#host_name localhost.localdomain

# A list of addresses that are allowed to connect.  This must be a
# regular expression, since Net::Server does not understand CIDR-style
# network notation unless the perl module Net::CIDR is installed.  You
# may repeat the allow line as many times as you'd like

allow ^127\.0\.0\.1$
allow ^::1$
allow ^176\.9\.83\.60$
allow ^159\.69\.191\.196$
allow ^2a01:4f8:1c17:44d1::1$


# If you have installed the Net::CIDR perl module, you can use one or more
# cidr_allow and cidr_deny address/mask patterns.  A connecting client must
# match any cidr_allow, and not match any cidr_deny.  Note that a netmask
# *must* be provided, even if it's /32
#
# Example:
#
# cidr_allow 127.0.0.1/32
# cidr_allow 192.0.2.0/24
# cidr_deny  192.0.2.42/32
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# Which address to bind to;
# host *
# host 127.0.0.1
host 5.181.50.231
host 2a03:4000:3f:4db:7824:4eff:fe98:638

# And which port
port 4949
EOF
```

Node restarten

```
systemctl restart munin-node
```

2.11 Extras

2.11.1 VPN (Mullvad)

Wenn der Übergang in das Internet nicht auf diesem Knoten liegen soll kann man ein VPN nutzen. Im Folgenden ist die Konfiguration beschrieben. Die meisten Gateways nutzen das derzeit nicht.

Achtung: Kopiere bitte nicht die Konfigurationsdateien von einem Gateway auf andere Gateways!

Für das VPN werden diese Dateien benötigt, die alle nach `/etc/openvpn/` müssen:

```
ca.crt
crl.pem
mullvad.crt
mullvad.key
mullvad_linux.conf
```

Die Datei `mullvad\_linux.conf` muss noch um folgende Zeilen am Ende ergänzt werden:

```
#custom
route-noexec
up /etc/openvpn/mullvad_up.sh
up /etc/fastd/ffsh/iptables_ffsh.sh
```

Mullvad hat an seinen Konfigurationen seit mehreren Sicherheitslücken bei OpenVPN und Snowden/NSA geändert. Es kann sein, dass ein Fehler zur Cipher-Liste angezeigt wird. Dann muss in der `mullvad\_linux.conf` die Zeile zur TLS-Verschlüsselung beginnend tls-cipher auskommentiert werden. Wenn kein IPv6 am Server ins Internet möglich ist, kann auch tun-ipv6 auskommentiert werden.

Die Datei `/etc/openvpn/mullvad\_up.sh` gibt es noch nicht.Also bitte die Datei mit folgenden Zeilen anlegen:

```
#!/bin/sh
ip route replace 0.0.0.0/1 via $5 table 42
ip route replace 128.0.0.0/1 via $5 table 42

service dnsmaq restart
exit 0
```

Diese Datei muss nun auch als `root` ausführbar gemacht werden:

```
chmod +x /etc/openvpn/mullvad\_up.sh
```

Damit Linux auch diese VPN-Schnittstelle kennt, muss tun in der Datei `/etc/modules` bekannt gemacht werden. OpenVPN benötigt ein tun-Interface. Trage einfach in eine eigene neue Zeile dies ein

```
tun
```

Bitte nun als `root` über die Konsole tun aktivieren und den VPN starten mit:

```
modprobe tun
service openvpn start
```

2.11.2 VPN-Connect regelmäßig überprüfen

Es ist sinnvoll regelmäßig zu prüfen, ob die VPN Verbindung noch aktiv ist. Dazu wird ein Script auf dem Server abgelegt, dass dann über den CRON immer neu den VPN-Connect prüft.

`/ffsh/check-vpn.sh`

```
#!/bin/bash

# Test gateway is connected to VPN
test=$(ping -q -I tun0 8.8.8.8 -c 4 -i 1 -W 5 | grep 100 )

if [ "$test" != "" ]
    then
    echo "VPN nicht da - Neustart!"
    service openvpn restart      # Fehler - VPN nicht da - Neustart
else
    echo "alles gut"
fi
```

Dann noch das Script ausführbar machen:

```
chmod ug+x /ffsh/check-vpn.sh
```

Danach in die Datei `/etc/crontab` das Skript alle 10 Minute auszuführen und damit regelmäßig der VPN-Status geprüft wird.

```
# Check VPN via openvpn is running, if not service restart
*/10 * * * * root /ffsh/check-vpn.sh > /dev/null
```

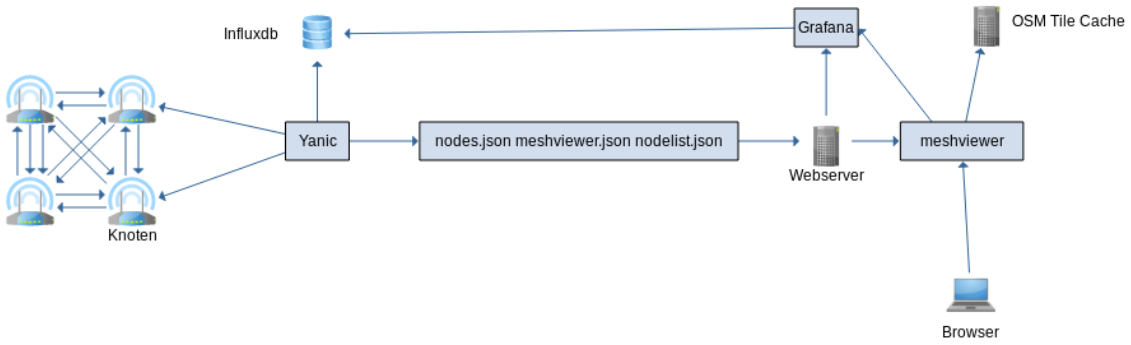
Die Änderungen übernehmen durch einen Neustart des Cron-Dämonen:

```
service cron restart
```

KAPITEL 3

meshviewer

Die Folgenden Schritte dienen dazu, eine Karte anzuzeigen. Dies lässt sich auch auf einem vom Gateway getrennten System durchführen.



3.1 Vorraussetzungen

Für einen Kartenserver brauchen wir einen Server mit einem aktuellen Linux, diese Anleitung geht von Debian 9 aus. Die Hardware Anforderung hängen von der Größe und der zu erwartenden Last der Community ab. Bei uns läuft die Karte auf einem der Gateways mit 4 VCores und 4GB RAM und es gab keine Probleme.

Eine Anbindung an das entsprechende Freifunk Netzwerk ist natürlich auch notwendig.

3.2 yanic

yanic sammelt von den Knoten Daten, welche dann auf einer Karte angezeigt werden können, früher wurde hierfür

Alfred benutzt. yanic ist in go geschrieben also installieren wir eine neue Version von go. golang

Als erstes wechseln wir in den *root* user.

```
sudo su
```

```
wget https://dl.google.com/go/go1.12.4.linux-amd64.tar.gz
# Bitte sha256 vergleichen https://golang.org/dl/
tar -C /usr/local -xzf go1.12.4.linux-amd64.tar.gz
rm go1.10.8.linux-amd64.tar.gz
```

In `~/.bashrc`

```
GOPATH=/opt/go
PATH=$PATH:/usr/local/go/bin:$GOPATH/bin
```

Hier musst du dich einmal abmelden und neu anmelden damit die Variablen auch gesetzt werden.

Nach dem Anmelden kann man prüfen ob die Variablen korrekt gesetzt wurden.

```
echo $GOPATH
/opt/go
```

Mit `whereis` go prüfen ob go gefunden wird:

```
go: /usr/local/go /usr/local/go/bin/go
```

Dann wird yanic installiert.

```
go get -v -u github.com/FreifunkBremen/yanic
```

Die Konfiguration von Yanic wird in `/etc/yanic.conf` angelegt:

```
[respondd]
enable           = true
synchronize      = "1m"
collect_interval = "1m"

[respondd.sites.ffsh]
domains            = ["ffsh", "ffod", "ffrz"]

[respondd.sites.ffod]
domains            = []

[respondd.sites.ffrz]
domains            = []

[[respondd.interfaces]]
ifname          = "bat0"

[[respondd.interfaces]]
ifname            = "bat0"
multicast_address = "ff02::2:1001"

[webserver]
enable  = false
bind    = "127.0.0.1:8080"
webroot = "/var/www/html/meshviewer"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
[nodes]
state_path    = "/var/lib/yanic/state.json"
prune_after   = "14d"
save_interval = "2m"
offline_after = "10m"

[[nodes.output.geojson]]
enable   = true
path = "/var/www/map/data/nodes.geojson"

[[nodes.output.meshviewer-ffrgb]]
enable   = true
path = "/var/www/map/data/meshviewer.json"

[nodes.output.meshviewer-ffrgb.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[[nodes.output.meshviewer]]
enable = true
version    = 2
nodes_path = "/var/www/map/data/nodes.json"
graph_path = "/var/www/map/data/graph.json"

[nodes.output.meshviewer.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[[nodes.output.nodelist]]
enable   = true
path = "/var/www/map/data/nodelist.json"

[nodes.output.nodelist.filter]
no_owner = true
blacklist = ["f4f26d4a2ecc"]

[database]
delete_after    = "30d"
delete_interval = "1h"

[[database.connection.influxdb]]
enabled  = true
address  = "http://localhost:8086"
database = "ffsh"
username = ""
password = ""

[database.connection.influxdb.tags]

[[database.connection.graphite]]
enable   = false
address  = "localhost:2003"
prefix   = "freifunk"

[[database.connection.respondd]]
enable   = false
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
type     = "udp6"
address  = "stats.bremen.freifunk.net:11001"

[[database.connection.logging]]
enable   = false
path     = "/var/log/yanic.log"
```

Wir können testen ob yanic funktioniert in dem wir eine manuelle Anfrage stellen hier an das Gateway Hopfenbach:

```
yanic query --wait 5 br-ffsh "fddf:0bf7:80::128:1"
```

Damit yanic auch als Deamon läuft legen wir noch einen Service an.

```
sudo cp /opt/go/src/github.com/FreifunkBremen/yanic/contrib/init/linux-systemd/yanic.
↪service /lib/systemd/system/yanic.service
sudo systemctl daemon-reload
```

3.3 influxdb

Influxdb dient als Datenbank für `yanic`

```
sudo apt install influxdb influxdb-client
```

Nun sichern wir die influxdb ab `/etc/influxdb/influxdb.conf`

Hier werden nur die empfohlenen Anpassungen beschrieben: Noch vor der `[meta]` Sektion setzen wir, sonst wäre der port 8088 überall offen.

```
bind-address = "localhost:8088"
```

Weiter unten bei `[admin]` das gleiche:

```
bind-address = "localhost:8083"
```

kurz danach in `[http]`

```
bind-address = "localhost:8086"
```

```
systemctl restart influxdb
```

Nun sollte influxdb nur noch auf localhost erreichbar sein, prüfen kann man dies mit `netstat -tlpn`

3.4 Grafana

Grafana kann Graphen erstellen welche im meshviewer eingebunden werden können. Hier wird Grafana über eine Repository installiert.

```
deb https://packagecloud.io/grafana/stable/debian/ stretch main
curl https://packagecloud.io/gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get install grafana
```

Da Grafana bei uns hinter einem Proxy laufen soll, setzen wir auch hier alle IPs auf `localhost`. Am besten einmal am Ende prüfen ob alles richtig konfiguriert ist mit `netstat -tlpn`.

Ein wichtiger Punkt ist der öffentliche Zugang, damit die Statistiken auch von Besuchern abgerufen werden können.

```
#################################### Anonymous Auth ##########################
[auth.anonymous]
# enable anonymous access
enabled = true

# specify organization name that should be used for unauthenticated users
org_name = Freifunk Südholstein

# specify role for unauthenticated users
org_role = Viewer
```

Die Organisation kann man als Admin in Grafana anlegen.

3.5 meshviewer

Für den Meshviewer installieren wir als erstes nodejs und yarn

3.5.1 nodejs

Wir brauchen ein aktuelles nodejs das finden wir auf nodejs.org Wir benutzen die LTS Variante 8.x

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs
```

3.5.2 yarn

Dann installieren wir yarn

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.
↪list.d/yarn.list
apt install yarn
```

3.5.3 meshviewer-rgb

Nun installieren wir den meshviewer selbst. Im web Verzeichnis `/var/www/`

```
git clone https://github.com/ffsh/meshviewer.git
cd meshviewer
yarn
```

Nun muss die Konfiguration in `meshviewer/config.js` eventuell noch angepasst werden.

Danach yarn run gulp Nun muss nur noch ein Webserver meshviewer/build ausliefern.

3.6 Tile-cache mit nginx

Für den Meshviewer benötigt man einen Tile-Server, der die Karte als einzelne Kacheln ausliefert. Wir verwenden dabei das kostenlose und freie Angebot von OpenStreetMap. Damit die Server von OpenStreetMap weniger stark belastet werden verwenden wir einen Tile-Cache. Bei einer Anfrage für eine Karten-Kachel fragt der Browser den Cache, hat dieser die Datei bereits, so liefert er sie direkt aus. Hat er sie nicht, so fragt er bei den OpenStreetMap Servern und speichert die Datei in seinem Cache. Für die einfache Umsetzung haben ein paar Freifunker an einer Konfiguration für nginx gearbeitet, welche genau das umsetzt.

Voraussetzungen: - nginx erreichbar unter der entsprechenden Domain - TLS mit gültigem Zertifikat (Let's Encrypt) - ein wenig Speicherplatz

Die für uns angepasste Version ist so konfiguriert das sie sich hinter einem apache-Server befindet und deshalb keine TLS Konfiguration braucht, passe sie für deinen Anwendungsfall an.

```
#
# Nginx >= 1.9.15 - 1.10.1 recommended
#
# Thanks to https://github.com/cbricart
proxy_cache_path /var/www/tilecache/osm
  levels=1:2 inactive=7d
  keys_zone=tilecache:64m
  max_size=500M;

upstream osm_tiles {
        server a.tile.openstreetmap.org;
        server b.tile.openstreetmap.org;
        server c.tile.openstreetmap.org;
        keepalive 16;
}

server {
        listen 127.0.0.1:8090 ;

        access_log              /var/www/tilecache/logs/access.log;
        error_log               /var/www/tilecache/logs/error.log;

        root /var/www/tilecache/html;

        location / {
                try_files $uri @osm;
        }

        location @osm {
                proxy_pass http://osm_tiles;
                proxy_http_version      1.1;
                proxy_set_header    Connection "";
                proxy_set_header    Accept-Encoding "";
                proxy_set_header    User-Agent "Mozilla/5.0 (compatible; OSMTileCache/
↪1.0; +mailto:noc@freifunk-suedhosltein.de; +https://map.freifunk-suedholstein.de/)";
                proxy_set_header    Host tile.openstreetmap.org;

                proxy_temp_path     /var/www/tilecache/temp;
                proxy_cache         tilecache;
                proxy_store         off;
                proxy_cache_key     $uri$is_args$args;
                proxy_ignore_headers Expires Cache-Control;
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
                proxy_cache_valid   200 301 302 7d;
                proxy_cache_valid   404 1m;
                proxy_cache_valid   any 1m;
                proxy_next_upstream   error timeout invalid_header http_500 http_502␣
↪http_503 http_504 http_403 http_404;
                proxy_cache_use_stale error timeout updating invalid_header http_500␣
↪http_502 http_503 http_504 http_403 http_404;

                proxy_hide_header   Via;
                proxy_hide_header   X-Cache;
                proxy_hide_header   X-Cache-Lookup;

                expires 7d;
        }
}
```

KAPITEL 4

Firmware

4.1 site-repo

4.1.1 Struktur

Es gibt vier Hauptzweige

1. stable
2. rc
3. testing
4. dev

Dabei wird stets angestrebt von oben nach unten eine immer stabilere Firmware zu erstellen. Um dies sicherzustellen kommen bei `stable` überwiegend Gluon point-releases zum Einsatz also z.B. `v2018.1`. Der Zweig `rc` ist wiederum dazu gedacht, Gluon point-releases vor der Veröffentlichung in `stable` zu testen. Der Zweig `testing` basiert stets auf einem Gluon main-release also `v2018.1.x`. Während die point-releases als tag (git tag) auf einen bestimmten Commit zeigen und sich nicht mehr ändern, handelt es sich bei den main-releases um Zweige die regelmäßig Updates erhalten. Daher erhält der `testing` Zweig auch häufiger Updates. Der letzte Zweig ist `dev` dieser wird verwendet, um den `master` oder `next` Zweig von Gluon in unserer Umgebung frühzeitig zu testen. Er erhält nach Bedarf Updates.

Für das Git-Repository bedeutet dies, dass Änderungen in der Regel zu erst in `dev` landen. Die anderen Zweige basieren auf `dev` und haben nur einen weiteren Commit der die nötigen Anpassungen (andere Gluon-Version) enthält.

4.2 Versionierung

Während wir die Gluon-Versionierung übernehmen fügen wir eine „build number“ als auch den Zweig zur Version hinzu.

```
gluon-ffsh-$gluonVersion-$buildNumber-$Zweig-$Gerät
gluon-ffsh-2018.1.1-147-dev-tp-link-archer-c7-v4-sysupgrade.bin
```

- $gluonVersion: entspricht der Gluon-Version
- $buildNumber: wird mit jedem versuch eine Firmware zu bauen hochgezählt
- $Zweig: stable, rc, testing, dev
- $Gerät: das entsprechende Gerät

Die Gluon-Version bietet den eindeutigen Vorteil das auf einen Blick erkennbar ist welche Version von Gluon auf dem Gerät läuft. Die „build number“ sorgt dafür, dass innerhalb einer Gluon-Version mehrere Versionen verteilt werden können. Dies kann Nötig werden wenn ein Paket vergessen wurde. Den Zweig in der Version zu haben, lässt den Betrachter sofort erkennen, mit welcher Version er es zu tun hat.

Die Reihenfolge wurde gewählt um Updates von einen Zweig auf den anderen zu ermöglichen ob dies empfehlenswert ist, steht auf einem anderen Blatt. Da es jedoch in der Vergangenheit zu wechseln des Zweiges kam wurde dies als wichtige Funktion betrachtet.

Wenn in Zukunft ein neuer Branch eingeführt würde könnten mit dem `au-changer` Paket alle Knoten auf dem alten Zweig auf diesen eingestellt werden. Und es wäre nicht abhängig vom Namen oder der Gluon-Version ob ein update erfolgt oder nicht. Dieser extra Aufwand ist notwendig weil der autoupdater in Gluon sehr einfach aufgebaut ist und die Zweige keine Gewichtung oder ähnliches haben. Ob ein Update erfolgt oder nicht, wird durch Vergleichen der eigen Version, mit der Version im Dateinamen entschieden. Dabei kommt im Grunde nur ein < Vergleich zum Einsatz.

4.3 build.py

Das `build.py` Skript ist ein in Python geschriebenes Programm zum Bauen der Gluon Firmware. Es ist nicht Perfekt und könnte ein Reafctoring gebrauchen aber es erledigt seinen Job. Im folgenden werden zunächst zwei Tabellen gezeigt und dann anhand eines Beispiels das Bauen der Firmware präsentiert.

4.3.1 Tabellen

`build.py` unterstützt verschiedene Befehle (commands):

command	value	make „equivalent“	Kommentar
-c	update	make update	lädt opwenwrt und wendet gluon patches an
-c	build	make build	baut die firmware
-c	clean	make clean	löscht alle packages des targets
-c	dirclean	make dirclean	löscht alle targets und die toolchain
-c	sign	n/a	signiert die firmware

Außerdem gibt es eine Reihe von Argumenten.

command	value	default	name	pflicht	Kommentar
-b	dev or testing or rc or stable	dev	Branch	ja	der Firmware branch
-w	site	n/a	Workspace	ja	Pfad zum site Repository
-n	42	n/a	Build Number	ja	build Nummer wird von jenkins automatisch hochgezählt wird im firmware Namen verwendet
-t	ar71xx-generic or ...	all targets	Target	nein	ohne Angabe werden alle Targets gebaut, mit Angabe nur der angegebene Target
-s	<pfad zu secret>	n/a	Secret	nein	wird nur beim signieren benötigt
-d	<pfad zu public directory>	n/a	Directory	nein	wird nur bei -publish benötigt
–commit	der verwendete commit	n/a	Commit	ja	commit sha, dient als Referenz im build.json
–cores	1 bis N	1	Cores	nein	Anzahl der zu verwenden Threads, Empfehlung: CPU-Kerne+1
–log	V=w or V=s | / | Log | ja				Log level w: nur Warnungen/Fehler, s: alles

4.3.2 Beispiel

```
1 git clone --recurse https://github.com/ffsh/site.git
2 cd site
3 ./build.py -c update -b grotax -n 1 -w $(pwd) --commit $(git rev-parse HEAD) --log
  ↪"V=w"
4 ./build.py -c build -t "ar71xx-tiny" -b grotax -n 1 -w $(pwd) --commit $(git rev-
  ↪parse HEAD) --log "V=w"
```

In Zeile 1 wird das Repository inklusive der „submodules“ geklont. Danach in Zeile 2 wechseln wir in das „site“ Verzeichnis. Dort führen wir zum ersten mal das `build.py` Skript aus (Zeile 3).

- `-c update` (Wir wollen die Abhängigkeiten von Gluon aktualisieren)
- `-b grotax` (Hier kann ein beliebiger Name eingesetzt werden)
- `-n 1` (Es ist unser erster build)
- `-w $(pwd)` (Der Workspace in diesem Fall das Aktuelle Verzeichnis (pwd))
- `--commit $(git rev-parse HEAD)` (Der Commit-Hash wird hier ermittelt)

In Zeile 4 besteht der unterschied dann nur in dem `-c build` (wir wollen nun bauen) und dem `-t "ar71xx-tiny"` (hier wird nur für ein target gebaut).

4.4 Mirror

Die Firmware wird zentral auf einer Storage Box von Hetzner gehostet. Diese Storage Box wird als Netzwerklaufwerk eingebunden.

Als root-User oder mit sudo

```
nano /etc/fstab
```

Die Datei um folgenden Eintrag ergänzen:

```
//u205465.your-storagebox.de/backup /mnt/firmware       cifs    iocharset=utf8,rw,
↪credentials=/etc/firmware-credentials.txt,uid=0,gid=0,file_mode=0644,dir_mode=0744␣
↪0       0
```

Dann erstellen wir die Credentials-Datei.

```
nano /etc/firmware-credentials.txt
```

Inhalt so wie hier Daten gibts beim NOC

```
username=
password=
```

Rechte anpassen, cifs installieren und einbinden

```
chmod 600 /etc/firmware-credentials.txt
apt install cifs-utils
systemctl daemon-reload
systemctl restart remote-fs.target
```

Jetzt hast du ein Netzwerklaufwerk :)

```
ls /mnt/firmware
```

Für nginx gibt es eine vorbereitete Konfiguration.

```
#
# Firmware Mirror configuration
#
server {
    listen 80;
    listen [::]:80;

    #
    # SSL configuration
    #
    #listen 443 ssl;
    #listen [::]:443 ssl;
    #ssl_certificate
    #ssl_certificate_key
    #    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    #    ssl_ciphers         HIGH:!aNULL:!MD5;

    root /mnt/firmware;

    index index.html index.htm index.nginx-debian.html;

    server_name firmware.freifunk-suedholstein.de firmware.ffshev.de fw.ffshev.de;

    location / {
        try_files $uri $uri/ =404;
        autoindex on;

        #
        # Wide-open CORS config for nginx
        #
        if ($request_method = 'OPTIONS') {
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
            #
            # Custom headers and headers various browsers *should* be OK with but aren
↪'t
            #
            add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-
↪With,If-Modified-Since,Cache-Control,Content-Type,Range';
            #
            # Tell client that this pre-flight info is valid for 20 days
            #
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain; charset=utf-8';
            add_header 'Content-Length' 0;
            return 204;
         }

        if ($request_method = 'POST') {
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-
↪With,If-Modified-Since,Cache-Control,Content-Type,Range';
            add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';
         }
         if ($request_method = 'GET') {
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-
↪With,If-Modified-Since,Cache-Control,Content-Type,Range';
            add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';
        }
    }
}
```

4.5 jenkins

Jenkins Projekt Bash Befehle

```
export PYTHONUNBUFFERED=1
./build.py -c update -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit $
↪{GIT_COMMIT} --log "V=w"

# Befehle die nur manchmal notwenig sind
#./build.py -c dirclean -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit $
↪{GIT_COMMIT}
#./build.py -c clean -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit $
↪{GIT_COMMIT}

./build.py -c build -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit $
↪{GIT_COMMIT} --cores "9" --log "V=w"
./build.py -c sign -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit ${GIT_
↪COMMIT} -s ${SECRET}
./build.py -c publish -b ${GIT_BRANCH} -n ${BUILD_NUMBER} -w ${WORKSPACE} --commit $
↪{GIT_COMMIT} -d "/var/www/firmware.grotax.de"
```

4.6 Changelog

Dieses Changelog stellt nur die durch uns durchgeführten Änderungen da. Für Änderungen an Gluon (bsp unterstützte Geräte) verweisen wir auf die Dokumentation von Gluon. Die Versionierung entspricht der Versionierung von Gluon. Dieses Changelog bezieht sich auf den `stable` Zweig die anderen Zweige können davon abweichen.

4.6.1 2018.2.2

- /

4.6.2 2018.2.1

- ntp server changes

4.6.3 2018.2

- fix: fehlendes Paket für die vpn-Konfiguration

4.6.4 2018.1.1

- release ohne viele Änderungen (bereits in >=134 enthalten)
- fehlendes Paket für die vpn Konfigruation

4.6.5 2018.1

- erste gemeinsame version für lauenburg und stormarn
- einührung der drei domains ffsh, ffod, ffrz
  - Freifunk Südholstein: ffsh
  - Freifunk Stormarn: ffod
  - Freifunk Lauenburg: ffrz
- bereits mit patches für autoupdater
- mit Einführung der neuen Karte wurde alfred entfernt (auch in 2017.1.x)
- Fehler im Autoupdater (2018.1 build < 131)

KAPITEL 5

Indices and tables

- genindex
- modindex
- search