L3-Freifunk Documentation

Release 0.0.1

Freifunk

19.08.2020

Inhaltsverzeichnis

Content:

KAPITEL 1

Komponenten des Netzes

1.1 Terminologie

Begriff	Bedeutung
Node	Freifunk-Router vor Ort
Gateway	Node mit der Fähigkeit Traffic ans Internet auszuleiten
Client	Rechner eines Nutzers, der mit einem Node verbunden ist.

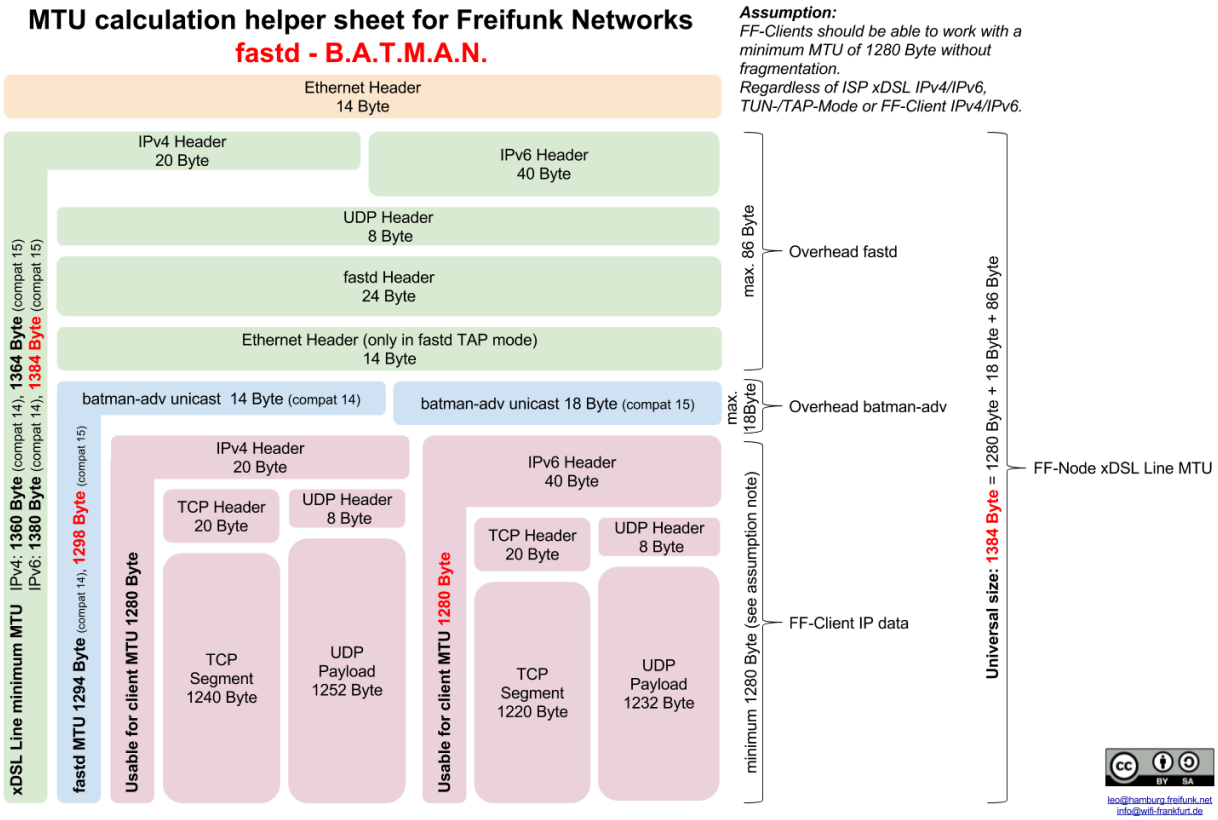
1.2 besondere Adressen/Netze

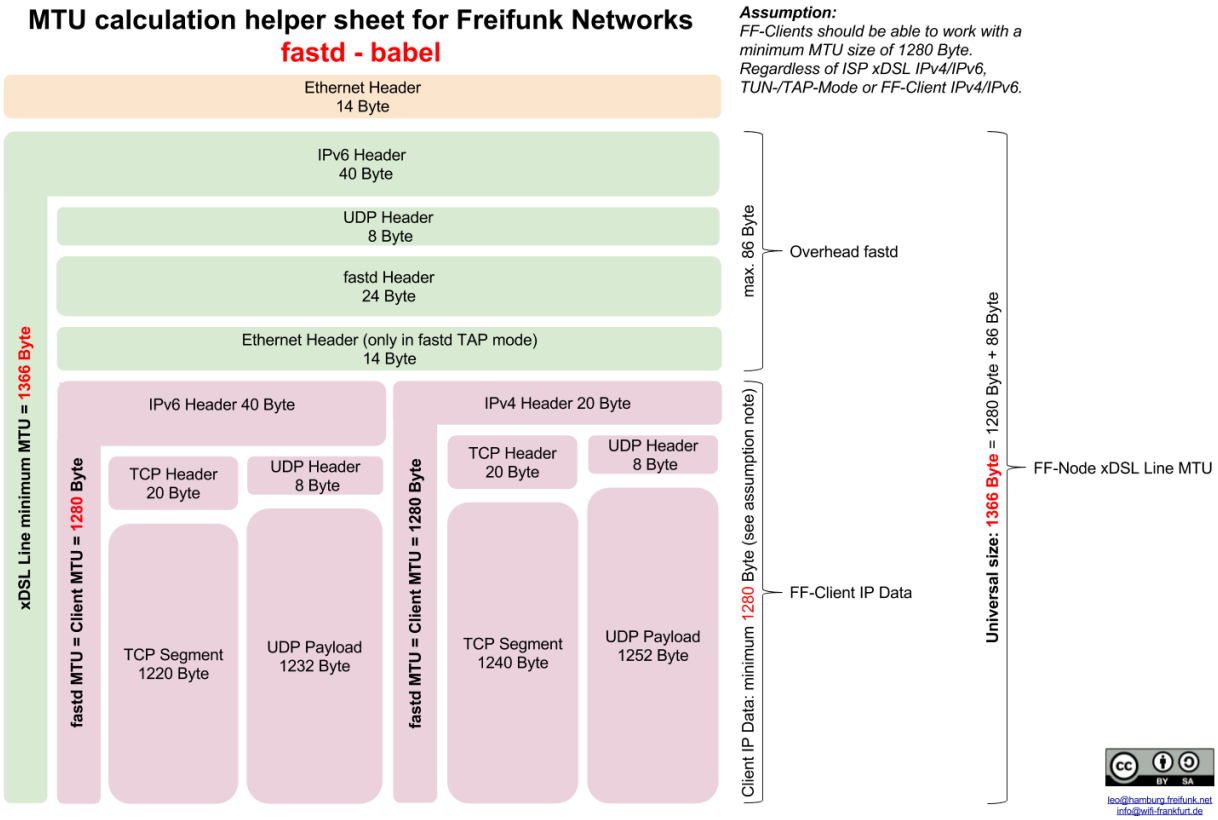
Adresse	Bedeutung
2001:DB8:3:1::1	next-node IP-Adresse: Diese Adresse ist jedem Node zugewiesen. Der Node ist für alle direkt verbundenen Clients unter dieser Adresse erreichbar.
2001:DB8:3:1::/128	Die bestimmte IP-Adresse in diesem Netz ist die local-node-IP-Adresse. Diese wird anhand der MAC-Adresse des Nodes bestimmt und dem lokalen Interface „lo“ zugewiesen.
2001:DB8:3:1::/64	Infrastruktur-Netz: In diesem Netz liegen Nodes und Serverkomponenten
2001:DB8:3:2::/64	Client-Netz: In diesem Netz liegen Clients. Dieses darf ein anderes als das node-Netz sein.

Jeder Node und jeder Client ist somit über eine öffentliche IPv6-Adresse erreichbar.

1.3 fastd

Nodes, die keine direkte WLAN-Verbindung haben, können über die VPN-Software fastd Teil eines gemeinsamen Netzes werden. In einem Batman-Netz werden die IP-Pakete des Freifunk-Netzes in Batman-Frames eingepackt und diese in fastd-Pakete gesteckt. Im Babel-Netz haben wir eine Verschachtelungsebene weniger, die IP-Pakete aus dem Freifunknetz können direkt auf den fastd-Interfaces transportiert werden. Die Datenströme zur Verwaltung des Netzes liegen daneben.





Die MTU von fastd berechnet den im fastd-Paket liegenden Ethernet-Header nicht mit ein. Bei einer fastd-MTU

von 1288 ist bereits fragmentierungsfreie IPv6-Kommunikation im Netz und außerhalb möglich. Im Mesh werden durch den mmfd Pakete in UDP-Pakete eingepackt. 1288 ist dadurch die minimale MTU, weil das kleinstmögliche fragmentierungsfreie Paket mit 1280 + 8-Byte UDP-Header über den fastd-Tunnel verschickt werden muss.

Der Einsatz anderer VPN-Technologien (auch im Parallelbetrieb zu fastd) ist denkbar. In Frankfurt haben wir uns entschieden nicht zu viele Komponenten gleichzeitig auszutauschen. Fastd hat sich bewährt und wird weiter eingesetzt.

1.4 babeld

babeld liest seinen Input für die Optimierung und Verteilung der Routen aus der main Routing-Tabelle. Der Rest funktioniert automatisch. Dabei ist auf dem gateway für 2001:DB8::/40 die folgende Konfiguration ein guter Startwert:

```
reflect-kernel-metric true
ipv6-subtrees true
export-table 10
import-table 11
import-table 12

interface mesh-vpn-1312
interface eth2
redistribute src-prefix 2001:DB8::/40 metric 25
```

1.5 l3roamd

l3roamd läuft auf jedem Node, damit Clientrouten ermittelt und übermittelt werden können.

1.6 mmfd

mmfd kann genutzt werden um multicast (gegenwärtig Broadcasts) in einem Layer-3-Meshnetz umzusetzen. Benötigt wird das um respondd auf den Nodes anzustoßen, Daten für die Map und das Monitoring bereitzustellen.

1.7 prefixd

eine Shell-Implementierung eines prefixd ist in einer Entwicklungsversion verfügbar. Dadurch wird die Verteilung eines eigenen prefix an lokale Clients ermöglicht.

1.8 respondd

Der Dienst wird wie im Batman-Netz auch genutzt um Monitoring und Mapdaten im Netz zu transportieren. Im Gegensatz zu derzeiten Batman-Netzen kommt der respondd auf einer ff05-Adresse zum Einsatz.

KAPITEL 2

Firmware

2.1 Netzwerke

2.1.1 Mesh

Dieses Netz enthält alle Interfaces, die zur Aufrechterhaltung des Mesh genutzt werden. In der Standardkonfiguration ist das das mesh0-Interface über welches Nodes mit WLAN-Kontakt ins Netz eingebunden werden und das mesh-vpn-Interface welches genutzt wird um Nodes per Internetverbindung zusammenzuschließen. Diese Interfaces werden anhand `proto = gluon_mesh` identifiziert.

2.1.2 Client

Dieses Netz enthält alle Interfaces über die Clients eine Verbindung zum Node herstellen können.

2.1.3 WAN

Dieses Netz ist das LAN des Betreibers des Nodes.

2.2 Pakete

2.2.1 l3roamd

l3roamd ermittelt Host-Routen von Clients und übergibt diese über die Routingtabelle 11 an Babel. Der L3roamd kommuniziert über einen Netzwerkport. Der Dienst muss auf allen Nodes im Netzwerk ausgeführt werden.

2.2.2 gluon-l3roamd

Dieses Paket enthält ein Initialisierungsscript für den l3roamd. Die Initialisierung ist bewusst in einem separaten Paket untergebracht. Sobald der l3roamd ebenfalls dynamisch per Socket konfiguriert werden kann, entfällt dieses Paket. Die Initialisierung erfolgt dann per setup.d-script in gluon-mesh-babel.

2.2.3 mmfd

Das Paket enthält ein mmfd-binary. Der mmfd flutet Nachrichten ins Layer 3-Netz.

2.2.4 babeld

Das Babel Protokoll ist für die Verteilung und Optimierung von Routen über verschiedene Nodes zuständig. Im Zuge der gluon-babel Entwicklung wurde das Babel 1.8-Paket in openwrt integriert. So kann zur Laufzeit auf die Konfiguration von babeld über einen Socket auf ::1 mit Port 33123 Einfluss genommen werden. Zu diesem Socket baut der mmfd eine Verbindung auf um alle nodes im Netz zu ermitteln.

Babel liest auf allen Nodes die Routingtabellen 11 und 12. In Tabelle 12 können Babel statische Routen übergeben werden. Tabelle 10 wird für das Routing genutzt, sofern die Quell-IP im Mesh- oder Clientprefix liegt.

2.2.5 gluon-mesh-babel

Dieses Paket integriert die einzelnen Komponenten mmfd, babeld, und die Firewall. Babeld und mmfd werden durch procd (neu)gestartet, sobald sich Interfaces ändern. Ob ein Interface für das mesh und damit für die Dienste Babel, l3roamd, mmfd relevant sind, wird anhand des hinterlegten Protokolls `proto = gluon_mesh` erkannt.

Tab. 1: Firewall

Zone	WAN	Client	Mesh	l3roamd	mmfd
Interfaces	br-wan	br-client	mesh-vpn, meshX	l3roam0	mmfd0
Protokolle OUT	fastd, dns	–	–	–	–
Protokolle IN	ssh	ssh, dns, http, ntp	http, babel, l3roamd, ssh, ntp, dns,mmfd	–	respondd
Protokolle both	dhcp, ICMPx	ICMPx	ICMPx, respondd	–	–
Policy IN	DROP	DROP	DROP	ACCEPT	DROP (erlaube nur Traffic vom lokalen Node)
Policy OUT	ACCEPT	ACCEPT	ACCEPT	ACCEPT	ACCEPT
Policy FORWARD	DROP	DROP, (erlaube Traffic von und nach mesh sowie von und nach client)	DROP (erlaube forward von und nach client und von und nach mesh)	DROP (erlaube von und nach client)	DROP

Auf dem Gerät laufen zwei Instanzen von dnsmasq. Eine Instanz dient der Möglichkeit der Namensauflösung für fastd. Per Firewall-Rule wird dem fastd-User erlaubt auf diese dnsmasq-Instanz auf Port 54 zuzugreifen. Die zweite

dnsmasq-Instanz dient als dns-Cache für angeschlossene Clients. Hier wird der AAAA-Record „nextnode“ in die nextnode-IP-Adresse aufgelöst.

Interfaces, die das mesh-Protokoll ausführen, werden durch das gluon-core Paket automatisch der mesh-Firewall-Zone zugeordnet.

2.2.6 gluon-core

Nodes verteilen mittels Router Advertisement ein nutzbares Prefix im Freifunk-Netz. Außerdem wird per rdnssd der aktuelle Node als DNS-Server propagiert. Der DNS-Cache wird wie folgt konfiguriert:

```
dns = {
cacheentries = 4096,
servers = { '2001:DB8:3:53::53' , } ,
},
next_node = {
name = 'nextnode',
...
}
```

- cacheentries gibt an, wie viele Einträge der Cache enthalten soll. Ein Eintrag benötigt ca 90 Byte RAM. Der RAM wird beim Starten des Nodes zugewiesen.
- servers enthält die Liste der Upstream-DNS-Server an welche die ankommenden Anfragen im Fall eines Cache-Miss weitergeleitet werden
- Der dnsmasq löst den unter next_node.name angegebenen Namen, hier „nextnode“, in die next-node IP-Adresse auf.

Die Änderung ist bereits in gluon Verfügbar und auch in einem Batman-Netz nutzbar. In Frankfurt ist es bereits im Einsatz. Die DHCP-Server auf den Gateways müssen beim Einsatz in einem Batman-Netz so konfiguriert werden, dass diese als DNS-Server die nextnode IPv4-Adresse bekanntgeben.

KAPITEL 3

Gateway setup

Gateways connect the mesh by allowing access to a local VPN. For Babel networks, the wireguard VPN is recommended. Fastd is still possible albeit slower.

The gateway setup is described in docker files that will be run with host-network mode. The following services are recommended:

3.1 Wireguard/babel/mmfd/l3roamd

The docker image klausdieter371/wG-docker which is hosted on https://github.com/christf/wg-docker.git provides the essential mesh and vpn services. The container can be started using the compose file. It requires a bit of environment:

3.1.1 network

The address in the environment variable OWNIP must be assigned to one of the network devices of the gateway. It is possible to use an ifup script for that:

```
#!/bin/bash
ip -6 r d default
ip -6 r a default via fe80::1 dev eth0 src 2a01:4f8:1c1c:71b5::1

# lookup clat prefix in freifunk routing table
ip -6 ru a to fdff:ffff:ffff::/48 lookup 10
ip -6 ru a to fdff:ffff:fffe::/48 lookup 10

# reach the rest of the batman network
ip -6 r a fda9:26e:5805::/64 dev backend-gw2 proto static

ip -6 a a fda9:26e:5805:bab1:aaaa::1/64 dev eth0
ip -6 r a fda9:26e:5805::2 dev backend-gw2 proto static t 12
ip -6 r a fda9:26e:5805::2 dev backend-gw2 proto static t 10
ip -6 r a 2000::/3 from fda9:26e:5805::/48 dev backend-gw2 proto static t 10
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
ip -6 r a 2000::/3 from fda9:26e:5805::/48 dev backend-gw2 proto static t 12
ip -6 r a fda9:26e:5805::/48 dev backend-gw2 proto static t 10
ip -6 r a fda9:26e:5805::/48 dev backend-gw2 proto static t 12
ip6tables -I INPUT 1 -i babel-wg-+ -s fe80::/64  -p udp -m udp --dport 6696  -j ACCEPT
ip6tables -I INPUT 1 -i babel-wg-+ -s fe80::/64  -p udp -m udp --dport 27275  -j␣
↪ACCEPT
ip6tables -I INPUT 1 -i babel-wg-+ -s fda9:026e:5805:bab1::/64  -p udp -m udp --dport␣
↪6696  -j ACCEPT
ip6tables -I INPUT 1 -i babel-wg-+ -s fda9:026e:5805:bab1::/64  -p udp -m udp --dport␣
↪27275  -j ACCEPT
ip6tables -I INPUT 1 -i babel-wg-+ -p udp -m udp --dport 5523  -j ACCEPT
ip6tables -t mangle -A FORWARD -o babel-wg-+ -p tcp -m tcp --tcp-flags SYN,RST SYN -j␣
↪TCPMSS --clamp-mss-to-pmtu
iptables -t mangle -A FORWARD -o babel-wg-+ -p tcp -m tcp --tcp-flags SYN,RST SYN -j␣
↪TCPMSS --clamp-mss-to-pmtu
ip6tables -t mangle -A OUTPUT -o babel-wg-+ -p tcp -m tcp --tcp-flags SYN,RST SYN -j␣
↪TCPMSS --clamp-mss-to-pmtu
iptables -t mangle -A OUTPUT -o babel-wg-+ -p tcp -m tcp --tcp-flags SYN,RST SYN -j␣
↪TCPMSS --clamp-mss-to-pmtu
exit 0
```

3.1.2 Configuration of the container via environment-file

You can use the docker-compose file to start the container

```
 version: "2.1"
services:
  wg:
    image: klausdieter371/wg-docker
    network_mode: "host"
    privileged: true
    cap_add:
      - NET_ADMIN
    devices:
      - /dev/net/tun:/dev/net/tun
    sysctls:
      - net.ipv6.conf.all.forwarding=1
      - net.ipv6.conf.all.accept_redirects=0
      - net.ipv4.conf.all.rp_filter=0
    restart: unless-stopped
    env_file: /root/wg-docker/wg-docker-env
```

The following kernel modules must be loaded - it is suggested to place this content in /etc/modules-load.d/wg.conf

```
ip6table_filter
ip6_tables
wireguard
```

The Container is parametrized by the below environment file:

```
DEBUG=

WGSECRET=abcdefghi
HOST_IP=
APT_PROXY_PORT=
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
# this is the whole net-wide prefix
WHOLENET=fda9:26e:5805::/48

# the prefix for infrastructure. This must be in $WHOLENET
NODEPREFIX=fda9:26e:5805:bab1::/64

# the prefix in which the clients pick their address. This must be in $WHOLENET
CLIENTPREFIX=fda9:26e:5805:bab0::/64

# the next-node address according to site.conf
NEXTNODE=fda9:26e:5805:bab0::1

# assign this address to one of the docker hosts nic
OWNIP=fda9:26e:5805:bab1:aaaa::1


# tcp port where the broker listens for connections
BROKERPORT=40000

# start of udp portrange accepting wireguard connections
STARTPORT=40000

# end of udp portrange accepting wireguard connections
ENDPORT=41000

# control port of local babeld
BABELPORT=33123

#MTU of the VPN in this network
MTU=1374

# wireguard command
WG=/usr/bin/wg

# accepting inbound wireguard connections on this interface
WAN=eth0

# allow MAXCONNECTIONS concurrent vpn connections
MAXCONNECTIONS=150

# this file contains the secret key
PRIVATEKEY=/etc/wg-broker/secret

# this is the l3roamd socket
L3ROAMDSOCK=/var/run/l3roamd.sock
MMFDSOCK=/var/run/mmfd.sock
MESHIFS="backend-bab1 backend-bab2"
```

3.2 Firewall

There is no setup script for a firewall yet. Make sure the required services are allowed to use for the network zones as appropriate:

- mmfd

- l3roamd
- babeld
- dns64
- gre

3.3 DNS64 – klausdieter371/docker-dns64

The docker image klausdieter371/docker-dns64 enables a named as dns64 server

3.3.1 Network setup

/etc/network/if-up.d/docker-dns64 is a good place to set up the ip address on which the server will listen for dns queries:

```
#!/bin/bash
ip -6 a a fda9:26e:5805:bab1:53::1/64 dev eth0
```

3.3.2 Configuration of the container via environment file

```
# listen on this ipv6 IP address. this is the address from the domain configuration␣
↪in the firmware.
DNS64_IP6_LISTEN=fda9:26e:5805:bab1:53::1
# no need to listen on a given ipv4 ip
DNS64_LISTEN=127.0.0.1
# clients within this range may use named
CLIENT_ACL=fda9:26e:5805::/48
# this depends on your jool configuration - use the same prefix.
DNS64_PREFIX=64:ff9b::/96
```

3.3.3 Compose-File

Use the compose file from https://github.com/christf/docker-dns64.git

```
Version: "2.1"
services:
  dns64:
    image: klausdieter371/docker-dns64
    network_mode: "host"
    restart: unless-stopped
    env_file: /root/docker-dns64/dns64-env
```

3.4 NAT64

. . . this is the place were the setup will be documented once it is dockerized . . .

3.5 exit

There are two deployment schemes possible: with NAT6 or without. Freifunk Magdeburg uses the former, Freifunk Frankfurt the latter. If NAT is to be used, it is recommended to do this on the gateway itself, not on the exit.

3.6 GRE

- load nf_conntrack_proto_gre such that gre streams are recognized as valid packets - and place the module in /etc/modules-load.d/gre.conf
- use the configuration for a gre tunnel in /etc/network/interfaces.d

KAPITEL 4

Indices and tables

- genindex
- modindex
- search