

# Freifunk-Firmware Testframework

BSc Praktikum WiSe 2015/16



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



darmstadt.freifunk.net



Chaos Darmstadt e.V.



## Was ist Freifunk?

- ▶ nichtkommerzielle Initiative
- ▶ Ziel: Aufbau und Betrieb eines freien WLAN-Meshnetzes
- ▶ dezentrales Netz besteht aus den Routern vieler einzelner Betreiber
- ▶ ca. 20.000 Freifunk-Router in Deutschland
- ▶ OpenWrt basierte Firmware von Enthusiasten in den Communities vor Ort entwickelt



## Was ist Freifunk?

- ▶ nichtkommerzielle Initiative
- ▶ Ziel: Aufbau und Betrieb eines freien WLAN-Meshnetzes
- ▶ dezentrales Netz besteht aus den Routern vieler einzelner Betreiber
- ▶ ca. 20.000 Freifunk-Router in Deutschland
- ▶ OpenWrt basierte Firmware von Enthusiasten in den Communities vor Ort entwickelt

Um die Qualität der ausgelieferten Firmware über Releases hinweg sicherzustellen ist es notwendig diese vor dem Ausrollen ausführlich auf verschiedenen Modellen zu testen. Diese mühsame und zeitaufwendige Prozedur soll in diesem Projekt automatisiert werden.



Konstruktion eines Testframeworks, das SOHO-Router mit einer Firmware beschreibt, konfiguriert und anschließend testet.

Hierzu wird die folgende Hardware bereitgestellt:

- ▶ Managed Switch, 16 Port
- ▶ regelbare Steckdosenleiste (z.B. Ubiquiti mPower)
- ▶ Embedded ARM-Board, z.B. RasPi2 (Controller)
- ▶ WLAN-Adapter 802.11ac



### Vorgaben zur Realisierung

- ▶ für Debian-basierte Linuxdistributionen (z.B. Raspbian)
- ▶ mit Python 3 ( $\geq 3.3$ ) nach gängigen Standards (z.B. PEP8)
- ▶ unter Revisionskontrolle mit Git und sprechenden Commit-Messages<sup>1</sup>
- ▶ lizenziert unter MIT, LGPL oder vergleichbaren Lizenzen

Die Kommunikation mit den Routern erfolgt im Regelfall per Telnet/SSH bzw. in späteren Ausbaustufen auch per tftp (Recovery, Image flashen) bzw. HTTP (Konfiguration/Ersteinrichtung über Weboberfläche).

---

<sup>1</sup>vgl. <http://chris.beams.io/posts/git-commit/>



Für den Regelbetrieb ist ein Prüfstand mit folgendem Aufbau vorgesehen:

- ▶ Bis zu 6 SOHO Router
- ▶ LAN- und WAN-Port mit einem Managed Switch verbunden
- ▶ LAN per Trunk, getagged an den Controller (RasPi2) angebunden
- ▶ WAN mit Dual-Stack Internet-Anbindung
- ▶ VLAN und ggf. Network Namespaces auf Controller



Das Testframework soll unter anderem folgende Funktionen bieten:

- ▶ Hardwareerkennung (Model, Revision, Primäre MAC-Adresse)
- ▶ Kaltstart der Geräte
- ▶ Tests für einzelne Geräte starten/stoppen
- ▶ Auswahl und Flashen der zu testenden Firmware
- ▶ Auszuführende Tests auswählen
- ▶ Abbruch einzelner Tests
- ▶ Statusübersicht über laufende Tests
- ▶ Ergebnisreport



Das Testframework soll unter anderem folgende Funktionen bieten:

- ▶ Steuerung der Tests per Weboberfläche und SSH
- ▶ Erkennung von Software-Defekten und Firmware-Recovery
- ▶ Automatischer Abruf neuer Firmware-Images vom Buildserver
- ▶ Unattended Betriebsmodus
- ▶ Ergebnisreport per Email
- ▶ Archivierung von Testberichten und Logdateien (syslog, dmesg)





- ▶ Durchlaufen des Konfigurationsassistenten
- ▶ Log-Prüfung, z.B. auf Kernel Panics
- ▶ Verbindung als WLAN-Client
- ▶ Erkennung von Mesh-Verbindungen
- ▶ Prüfung auf IPv6-Konnektivität
- ▶ Erreichbarkeit der Gateways
- ▶ Verbindung zu Update-Server
- ▶ Funktion des Autoupdaters
- ▶ Zeitsynchronisation mit NTP
- ▶ Internet-Verbindung für Clients
- ▶ Überprüfung der laufenden Dienste