autoupdate

ein automatisches Updateskript für Freifunk Berlin

Martin Hübner

21. März 2020

Inhaltsverzeichnis

Zusammenfassung

`autoupdate` möchte einen einfachen und dezentralen Ansatz für ein automatisches Update-System für Router mit Berliner Freifunk-Firmware bieten. Dezentral heißt hier, dass bei jedem Schritt des Update-Vorgangs eine Veränderung durch den Router-Betreiber möglich ist: Es können eigene Update-Branches angelegt und der Webspace zum Hosten der Download-Link-Listen kann verändert werden. Durch diese Kombination ist es auch möglich vollkommen eigene Firmware-Images an Gruppen von Routern oder einzelne Router auszuliefern. Denkbare Anwendungen wären z.B. Kiez-Setups.

1 Ausgangssituation

Mit Stand vom 15. März 2020 kennt die Berliner Freifunk-Karte[1] rund 800 Freifunk-Knoten in Berlin und Umland. Von diesen Knoten nutzen 200 Knoten das aktuelle Firmware-Release `hedy-1.0.4`. Weitere 180 Knoten nutzen ein älteres Release aus dem *Hedy*-Zweig, bzw. entsprechende developement-builds.

Alle weiteren Knoten nutzen entweder Firmwares, die nicht zugeordnet werden können *(unknown)*, oder veraltete Firmware. Bermerkenswert ist hier das Release `kathleen-0.0.0`[2], das noch auf 9 Routern läuft.

Die Gründe für dieses Firmware-Chaos können vielfältig sein. Vermutlich haben viele Menschen, die einen Freifunkrouter beherbergen, nicht das nötige Wissen oder die Zeit, Updates durchzuführen.

Abhilfe könnte ein Autoupdater schaffen, wie er z.B. in *Gluon*[3] vorhanden ist. Hierbei müssten jedoch einige Berlin-Spezifika beachtet werden:

- Im Gegensatz zu vielen Gluon-Netzen, wird in Berlin viel Wert auf Dezentralität gelegt. Zum Beispiel werden IP-Adressen nicht zentral durch einen DHCP-Server vergeben, sondern statisch auf den Routern konfiguriert.
- Es gibt einige Router, die in Türmen als Teil des *BBB*[4] betrieben werden. Diese müssen sehr zuverlässig laufen. Außerdem sind diese oft nicht sehr einfach zu erreichen, weshalb ein fehlgeschlagenes Update großen Aufwand verursachen würde.
- Die Firmware liegt aktuell[5] in drei wesentlichen Typen vor: `default`, `tunnel-berlin-tunneldigger` und `backbone`.

Folglich haben viele Router in Berlin eine quasi einzigartige Konfiguration.

2 Konzept und Überblick

`autoupdate` ist ein Shell-Skript[6]. Dadurch können recht einfach in der Firmware integrierte OpenWRT- bzw. busybox-Tools verwendet werden.

`autoupdate` wird einmal pro Woche per cron-job im Automatik-Modus *(siehe 3.2.2)* aufgerufen und lädt sich eine json-Datei herunter *(siehe 3.1.1)*. Diese enthält alle unterstützten Modelle mit, nach Firmware-Typ sortierten, Downloadlinks.

Um größtmögliche Flexibilität zu erreichen, ist sowohl Ort, von dem die json-Datei heruntergeladen wird, als auch der Name, der gleichzeitig den Update-Branch repräsentiert, frei wählbar *(siehe 3.2.1)*.

[1] https://hopglass.berlin.freifunk.net/
[2] erchienen im Okt. 2014
[3] https://wiki.freifunk.net/Gluon
[4] Berlin-Backbone
[5] Stand März 2020
[6] benutzt den busy-box- Interpreter `ash`

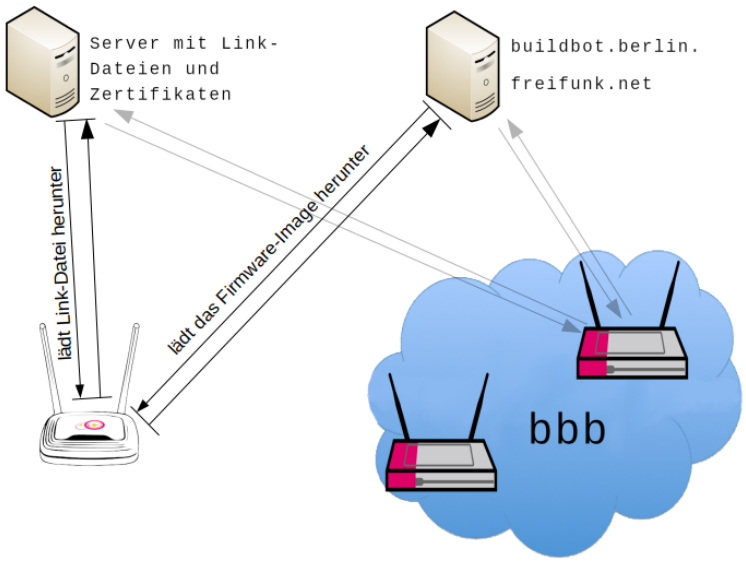

Abbildung 1: Prinzipskizze für `autopdate`: Der Server, der die Linklisten bereitstellt ist frei wählbar. Auch mehrere verschiedene Server sind denkbar.

Die json-Datei wird auf einem öffentlich erreichbaren Server gehostet. So ist sichergestellt, dass auch Router, welche nicht mit dem *BBB* verbunden sind, Updates herunterladen können.

Um die Authentizität der Downloadlinks zu belegen, wird die Datei signiert *(siehe 3.1.3)*.

Das eigentliche Update erfolgt über das OpenWRT-Tool `sysupgrade`.

3 Ablauf

3.1 Vorbereitungen

3.1.1 Dateiformat

`autoupdate` erwartet die Links als json-Datei, die ein Erstellungsdatum und die Router enthält. Das Datum liegt als Unix-Timestamp vor. Unterhalb der Routernamen werden die Uplink-Typen mit dem dazugehörigen Downloadlink gespeichert *(siehe Abbildung 2)*.

3.1.2 Link-Files erstellen mit json-creator

`json-creator`[7] ist ein python-Skript, das die Link-Listen erstellt. Dazu wird die Routertabelle im Firmware-Artikel[8] des Wikis ausgewertet und anschließend alle Links auf

[7] https://github.com/Akira25/json-creator
[8] https://wiki.freifunk.net/Berlin:Firmware#Unterst.C3.BCtzte_WLAN-Router

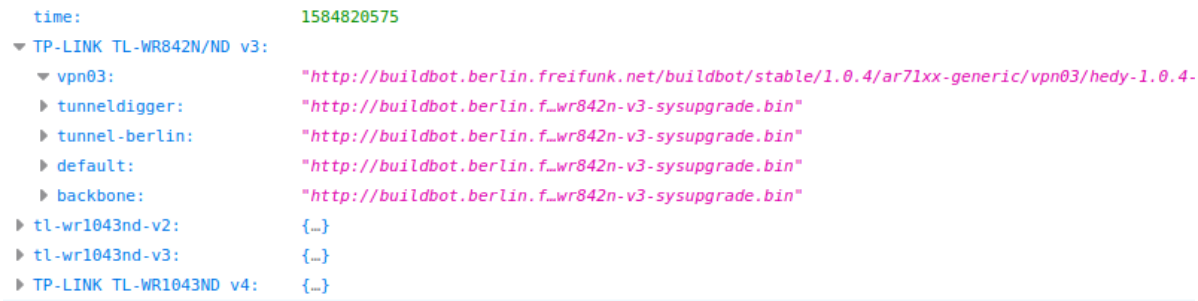

Abbildung 2: Dateiformat der Link-Listen

```
Usage:
        ./json-creator.py <arg> [arg]

Options:
 -r     generate release
 -t     generate testing
 -d     generate development

 -s     generate an (empty) strings-dictionary file.
 -a     write the complete list fetched from wiki (just for debugging)
 -h     print this message
```

Abbildung 3: Aufruf von `json-creator`

den hinterlegten buildbot-Seiten geparst.

Schon während des parsens werden Router, für die ihr *Router-String* bekannt ist, mit diesem benannt *(siehe 3.2.3)*. Das ist nötig, da der buildbot in der Schreibweise der Routernamen oft von den Strings abweicht, die auf den Routern gespeichert sind.

Aus der Gesamtmenge aller Image-Links werden anschließend nur die gewünschten Dateien erzeugt.

release und *testing* sind dabei inhaltlich exakt gleich. *testing* ist nach Vorstellung des Autors dazu bestimmt, den Update-Vorgang vor einem Rollout des Releases an einer kleineren Gruppe von Routern zu testen.

3.1.3 Signieren mit usign / signify-openbsd

Signiert werden die Link-Listen mit `signify`[9]. Unter OpenWRT heißt das entsprechende Paket `usign`. Für weitere Informationen zu deren Nutzung, bitte die entsprechenden Man-Pages aufrufen.

`autoupdate` führt das Update nur durch, wenn eine Mindestanzahl von gültigen Zertifikaten vorliegt. Die Anzahl kann frei gewählt werden. Tabelle 1 versteht sich als ein Vorschlag, wie hoch die Anzahl von Zertifikaten für die einzelnen Branches mindestens sein sollte.

[9]Das Paket heißt bei Ubuntu `signify-openbsd`

Branch	Anzahl
`release`	4
`testing`	2
`development`	1

Tabelle 1: Vorschlag für die Mindestanzahl gültiger Zertifikate für ein Update

```
config settings 'backup'
        option client_name 'complete-hostname'
        option client_user 'user'
        option client_path '/complete/path/to/your/backups/'

config settings 'router'
        option model 'auto'
        option type 'auto'

config settings 'automode'
        option automode 'false'
        option branch 'release'

config settings 'internal'
        option json_link_server 'http://freifunk.some-server.eu/'
        option log_file '/root/autoupdate.log'
        option send_string_url 'http://freifunk.some-server.eu'
        option minimum_signatures '4'
        option last_upgr '19700101'
```

Abbildung 4: Listing von `/etc/config/autoupdate`

3.1.4 Ablegen auf dem Webserver

`autoupdate` sucht die Link-Liste und die Zertifikate unter der URL, die in der Konfiguration angegeben wurde. Beide müssen im gleichen Ordner liegen. Als Zertifikat wird alles das interpretiert und zum Abgleichen heruntergeladen, was mit `$BRANCH*` matcht.

3.2 Update

3.2.1 Konfiguration

`autoupdate` wird über eine UCI-Datei unter `/etc/config/autoupdate` konfiguriert *(siehe Abbildung 4)*. Die wichtigsten Optionen können auch über LuCI verändert werden[10].

config settings 'backup' Hier kann ein remote-backup konfiguriert werden. Immer wenn dieses über die Option `-r` aufgerufen wird, wird mit `sysupgrade -b` ein Backup

[10] *Administration → System → Autoupdater*

erzeugt und über scp auf den definierten Rechner kopiert.

config settings 'router' Mit diesen Parametern kann die automatische Erkennung des Routermodells und des Firmware-Typs überschrieben werden. Bei manchen Modellen ist dies von vornherein nötig, da sie z.B. die falsche Hardware-Revision angeben. *(siehe 3.2.3)*

config settings 'automode' Hier werden der Branch festgelegt und `autoupdate` an- und abgeschaltet. Der Branch ist ein frei wählbarer String, wodurch auch das Anlegen von beliebigen eigenen Branches möglich wird. Dazu kann auch in der Sektion *internal* ein eigener Server definiert werden, um von anderen Servern unabhängig zu sein. Die Option *automode* darf neben Booleans auch Werte wie '0' oder 'no' enthalten.

config settings 'internal' Für den fortgeschrittenen Nutzer dürften hier vor allem die Optionen *json_link_server* und *minimum_signatures* interessant sein. Mit der Ersten wird das Verzeichnis definiert, in dem die Link-Listen und Zertifikate zu finden sind. Die Zweite legt fest, wie viele Zertifikate mindestens gültig sein müssen, um ein Update durchzuführen.

3.2.2 Aufruf

`autoupdate` kann in seiner derzeitigen Fassung nur *eine* Option pro Aufruf verarbeiten. Ein Listing aller verfügbaren Optionen ist in Abbildung 5 zu sehen.

`autoupdate` wird jede Woche Donnerstag per cron-job mit den Parametern `-a` und `-s` aufgerufen.

-a; automatic mode Es wird ein automatisches Update durchgeführt (sofern das Modell unterstützt wird). **Achtung:** Bevor diese Option benutzt wird, sollte unbedingt mit der Option `-p` überprüft werden, ob Routermodell und Tunneltyp korrekt erkannt werden und notfalls die richtigen Werte in `/etc/config/autoupdate` eingetragen werden!

Die Option `-a` kann nur genutzt werden, wenn in den Einstellungen der Wert automode auf true gesetzt wird.

In diesem Modus wird auch ein backup angelegt. Dieses wird unter `/root/backup/` abgelegt. Der Ordner ist persistent bei Updates über `sysupgrade`.

-m; manual mode Der Manual Mode führt ein Update interaktiv durch. Er schließt allerdings kein Backup mit ein.

-p; print router-string Diese Option gibt den Router-String und Uplink-Typ aus, der automatisch ermittelt wird. Bevor der Aotomatik-Modus eingeschalten wird, sollte so überprüft werden, ob der Router-String auch dem Modell (mit Hardware-Revision) enspricht.

```
Usage: autoupdate <option>

Options:
  -a    automatic mode
  -m    manual mode (requires user interaction)

  -p    print router-string and uplink-type
  -r    do a remote backup
  -s    send routerstring to the developers. (for maintenance of definition files)
```

Abbildung 5: Aufruf von `autoupdate`

-r; remote backup Sofern die entsprechenden Werte in `/etc/config/autoupdate` gesetzt sind, wird ein Backup erzeugt und per scp auf dem Remote-Rechner gespeichert *(siehe 3.2.1)*.

-s; send routerstring Diese Option überträgt den Router-String an den Server, der unter *send_string_url* eingestellt ist. Diese Strings werden dazu verwendet, um das Router-String-Dictionary zu aktualisieren.

3.2.3 Router-Strings

Router-Strings sind die Modellbezeichnungen, die auf dem Router unter `/proc/cpuinfo` in der Zeile *machine* stehen. Diese weichen in manchen Fällen bei den Hardware-Revisionen ab.

Wenn das der Fall ist, sollte der korrekte Router-String in der Konfiguration angegeben werden *(siehe 3.2.1)*.

4 Einschränkungen

`autoupdate` ...

- ... ist beta-Software!
- ... unterstützt nur Updates, die das Tool `sysupgrade` unterstützt.
- ... sollte nicht in backbone-Standorten eingesetzt werden.
- ... behandelt keine Fehler, die durch Updates in der Konfiguration entstehen können (z.B., wenn zwischen Releases sich Konfigurationsparamter ändern oder alte Konfigurationen inkompatibel werden). Hier müssen andere Mechanismen angewendet werden[11].

[11] z.B. das migration-script https://github.com/freifunk-berlin/firmware-packages/tree/master/utils/freifunk-berlin-migration