Kurz-Link: http://www.pro-linux.de/-0102688

WLAN-AP mit dem Raspberry Pi

Der Raspberry Pi ist ein sehr vielseitiges Gerät, das unter anderem auch zum Betreiben vielfältiger Netzwerkdienste verwendet werden kann. So ist es auch möglich, einen eigenständigen WLAN Access Point (AP) mit ihm zu betreiben, was hier näher erläutert werden soll.

Von Marvin Gülker[1]

Der Betrieb eines Wireless Local Area Network (WLAN) gehört mittlerweile wohl zu fast jedem Haushalt dazu. Schließt man heute einen Vertrag über die Zurverfügungstellung von Internet ab, so erhält man im Regelfalle von seinem jeweiligen Anbieter zum Abschluss des Vertrags ein Multifunktionsgerät, das neben der reinen Funktion, eine Verbindung zum größeren Netz des Internetanbieters (Wide Area Network, WAN) und mithin zum Internet herzustellen, das heißt als Modem zu fungieren, auch die Möglichkeit bietet, ein Drahtlosnetzwerk aufzubauen. Die interne Funktionsweise dieser Geräte bleibt von Seiten des Herstellers meist unter Verschluss, geschweige denn, dass man einen Konsolenzugriff auf das zumeist als Linux-Variante ausgeführte Betriebssystem erhielte. Prominente Ausnahmen von dieser Regel sind Hersteller wie Linksys oder AVM, die ihren Abnehmern einen relativ problemlosen Zugriff auf Shell und Betriebssystem gewähren.

Router sind zumeist hochspezialisierte Geräte, die darauf getrimmt sind, ihren Haupteinsatzzweck gut und anderweitige Aufgaben überhaupt nicht erfüllen zu können. In den meisten Geräten kommen stromsparende RISC-Prozessoren zum Einsatz; in neuester Zeit kommen mehr und mehr Geräte auf den Markt, deren Prozessor in ARM-Architektur gefertigt ist. Diese Rahmenbedingungen machen es denkbar schwierig, selbst bei einem verhältnismäßig offenen System beliebige Dienste auf der Hardware zu betreiben. Zu schnell erreicht man die Kapazität des oft nur geringen RAMs, anspruchsvollere Rechenaufgaben geraten zur Geduldsprobe. Dieser Artikel will sich daher damit beschäftigen, wie man mit gewöhnlicher Hardware einen eigenen WLAN Access Point betreibt, zu dem weitere Dienste nach Belieben zu- oder abgeschaltet werden können und dessen Hardware den angestrebten Aufgaben entsprechend größtenteils frei gewählt werden kann. Zum Einsatz kommt im konkreten Fall der Einplatinencomputer Raspberry Pi[2], doch sind die Erklärungen generisch und können auf jedes vollwertige Linux-System auf Systemd-Basis angewandt werden.

Voraussetzungen

Für den Betrieb eines eigenen WLANs sind einige Rahmenbedingungen erforderlich. Als erster und wichtigster Punkt ist hier eine funktionsfähige WLAN-Hardware zu nennen, die zwingend im Master-Modus betrieben werden können muss. Die Suche nach einer solchen gestaltet sich schnell schwierig, denn obwohl die Unterstützung der einzelnen Linux-Treiber für spezifische Chipsätze der Website des Linux-Kernels[3] entnommen werden kann, ist es aus den Produktbeschreibungen meist nicht ersichtlich, welcher Chipsatz in einem USB-WLAN-Adapter verbaut ist, bis man die Gelegenheit hatte, ihn selbst zu testen. Persönlich hat der Autor einen TL-WN722N von TP-Link im Einsatz, der allerdings nicht mehr hergestellt wird. Der verbaute Chipsatz stammt von Atheros und wird vom renommierten ath9k-Modul vollständig unterstützt.

```
$ lsusb | grep 802.11
Bus 001 Device 006: ID 0cf3:9271 Atheros Communications, Inc. AR9271 802.11n
```

Speziell für den Raspberry Pi ist zum Betrieb eines WLANs mit einem solchen WLAN-USB-Adapter zusätzlich ein USB-Hub mit eigener Stromversorgung erforderlich, da die vom Pi über die USB-Ports gelieferte Spannung nicht ausreicht, um einen stabilen WLAN-Betrieb zu gewährleisten. Häufige Ausfälle und Verbindungsabbrüche sind die Folge. Läuft alles gut, könnte die Interface-Liste so aussehen:

```
$ ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
[...]
4: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether b8:27:eb:fe:1c:c1 brd ff:ff:ff:ff:ff:ff
5: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether f8:1a:67:21:d2:a7 brd ff:ff:ff:ff:ff:ff
[...]
```

`eth0` stellt hierbei das am Pi in Revision B ohnehin vorhandene Ethernet-Interface da, `wlan0` repräsentiert den USB-WLAN-Adapter.

Auf der Softwareseite ist die Installation des WLAN-Daemons hostapd[4] erforderlich. Will man IP-Adressen nicht statisch vergeben, so ist zudem die Installation eines IP-Management-Daemons wie dnsmasq[5] ratsam, der sowohl zur Vergabe von IPv4-Adressen per DHCP als auch zur Versendung von IPv6-Router-Advertisements verwendet werden kann. Die entsprechenden Pakete sollten in den Repositorien aller gängigen Distributionen aufzufinden sein.

Die Notwendigkeit beider Dienste zeigt die klare Aufgabenteilung: hostapd ist nur für die Verschlüsselung der Drahtlosverbindung zuständig. Clients, die diese erste Hürde mit korrektem Passwort überwinden, wären ohne einen dahinter verfügbaren DHCP-Server (oder eine anderweitige IP-Konfiguration) trotzdem nicht in der Lage, eine Verbindung zum Netzwerk aufzubauen.

Konfiguration: Kernel

Der Pi muss die vom Ethernet-Interface angenommenen Pakete auf das WLAN-Interface weiterleiten und umgekehrt. Mit anderen Worten: Er fungiert als Router, also als Knotenpunkt zwischen zwei physikalisch getrennten Netzen (hier dem Kabelnetz und dem Drahtlosnetz). Diese »Durchleitungsfunktion« von Linux ist aus Sicherheitsgründen standardmäßig deaktiviert, kann aber leicht wie folgt aktiviert werden:

```
# sysctl net/ipv4/conf/all/forwarding=1
# sysctl net/ipv6/conf/all/forwarding=1
```

Diese Befehle setzen für alle vorhandenen Netzwerkinterfaces den Weiterleitungsstatus für den momentanen Boot. Eine dauerhafte Aktivierung lässt sich durch Anlegen der Datei **/etc/sysctl/90-ipforwarding.conf** (auf einigen Distributionen durch Anpassung der Datei **/etc/sysctl.conf**) erreichen:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
```

Selbstverständlich können diese Optionen auch nur für diejenigen Interfaces gesetzt werden, für die es gewünscht wird.

Konfiguration: hostapd

Der WLAN-Dienst hostapd wird mittels der Datei **/etc/hostapd/hostapd.conf** konfiguriert. Standardmäßig ist diese Datei mit einer Fülle von Konfigurationsdirektiven bestückt, die allerdings nur als Beispiele zu verstehen sind und die dementsprechend nicht einschüchternd wirken sollten. hostapd ermöglicht hochkomplexe Setups mit WPA-Enterprise-Verschlüsselung[6], externem RADIUS-Server[7] und Authentifikation gegen LDAP[8], die für das übliche Heim-WLAN allerdings sowohl Overkill als auch zu schwer zu warten sind. Die Reduktion auf das Wesentliche ergibt eine Konfiguration, die der folgenden ähneln wird:

```
# Interface
interface=wlan0
driver=nl80211
```

```
# Main settings
ssid=Mein tolles WLAN
channel=11
hw_mode=g
country_code=DE

# Security
wpa=3
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
wpa_passphrase=geheime_passphrase
auth_algs=1

# Logging
logger_syslog=-1
logger_syslog_level=2

# Misc
max_num_sta=60

# Files
deny_mac_file=/etc/hostapd/hostapd.deny
```

Diese Konfiguration setzt sich aus verschiedenen Elementen zusammen. Zunächst die grundlegenden Einstellungen: Die Einstellung `interface` gibt den Namen des Netzwerkinterfaces an, das von hostapd in den master-Modus versetzt werden soll, im konkreten Fall ist dies derjenige des WLAN-USB-Adapters. Mit `driver=nl80211` legt man den Interfacetyp fest, das heißt wie hostapd das Interface anspricht. Für alle gängigen WLAN-Verwendungen ist hier stets `nl80211` zu wählen, womit auf das mac80211-Subsystem des Kernels zurückgegriffen wird.

`ssid` legt den Namen des WLANs fest, so, wie er von den typischen Client-Programmen angezeigt wird. `channel` legt die Frequenz fest, auf der das WLAN gefunkt werden soll; eine Übersicht, in der Kanalnummer und Frequenz gegenübergestellt werden, kann in der englischen Wikipedia[9] eingesehen werden. Allerdings dürfen nicht alle theoretisch möglichen Frequenzen genutzt werden; hierzulande regelt die Bundesnetzagentur die für WLAN zulässigen Frequenzbereiche. Die für den Heimgebrauch freigegebenen Frequenzen können auf der Website der Behörde separat für WLAN im 2,4 GHz-Bereich[10] und 5 GHz-Bereich[11] eingesehen werden. Die Nutzung nicht genehmigter Frequenzen stellt nach §149 Abs. 1 Nr. 10 TKG[12] eine Ordnungswidrigkeit da, die nach §149 Abs. 2 Satz 1 3. Variante TKG[13] mit einem Bußgeld von bis zu 500.000 Euro geahndet werden kann. Der Parameter `country_code` instruiert hostapd, die im jeweiligen Land geltenden Vorschriften hinsichtlich der Kanal- und Sendeleistungsgrenzen zu beachten. Allein auf ihn verlassen sollte man sich aus naheliegenden Gründen jedoch nicht.

Die Kanäle 9 und 10 sind üblicherweise nicht zu empfehlen, da sie häufig Opfer von Interferenzen durch Mikrowellenherde werden. Ansonsten ist es ratsam, die in der Umgebung bereits genutzten Kanäle mit Werkzeugen wie iwlist(8) festzustellen und zum Zwecke möglichst geringer Interferenz einen noch nicht genutzten Kanal auszuwählen.

```
# ip link set wlan0 up
# iwlist wlan0 scan | grep -A1 Channel:
  Channel:1
  Frequency:2.412 GHz (Channel 1)
--
  Channel:6
  Frequency:2.437 GHz (Channel 6)
--
  Channel:104
  Frequency:5.25 GHz
# ip link set wlan0 down
```

Im Beispiel sind die Kanäle 1, 6 und 104 bereits von WLANs belegt, wobei letzterer ein Kanal aus dem 5 GHz-Bereich ist und dementsprechend ohnehin erst einmal zu ignorieren ist (die meisten handelsüblichen USB-WLAN-Adapter sind nicht in der Lage, 5 GHz im Master-Modus zu stemmen). Es sollte daher ein Kanal gewählt werden, der von den belegten Kanälen 1 und 6 möglichst weit entfernt ist; in der Beispielkonfiguration ist dies Kanal 11.

Die abschließende Angabe `hw_mode=g` legt den Modus fest, in dem hostapd das Interface betreiben wird. Es handelt sich dabei um den letzten Buchstaben des entsprechenden IEEE 802.11-Standards, wie folgender Tabelle entnommen werden kann:

WLAN-Frequenzbereiche		
Wert	**Standard**	**Frequenz**
a	IEEE 802.11a	5 GHz
b	IEEE 802.11b	2,4 GHz
g	IEEE 802.11g	2,4 GHz
ad	IEEE 802.11ad	60 GHz

Aufbauend auf diese Standards kann zusätzlich IEEE 802.11n aktiviert werden, indem die weiteren Direktiven `80211n` und `ht_capab` gesetzt werden. Dies ist für die ordnungsgemäße Funktion eines gewöhnlichem Heim-WLANs jedoch nicht erforderlich und soll daher auch nicht weiter behandelt werden. Generell sollte von g als solidem Standardwert nur nach vorangegangener Überlegung abgewichen werden.

Die im Block »Security« angebenen Einstellungen regeln, wie hostapd die Sicherheit des WLANs konzipiert. Der maßgebliche Parameter an dieser Stelle ist `wpa=3`, womit nach Wahl des Clients sowohl WPA als auch WPA2 als Verschlüsselungsmethode erlaubt wird. Wer nur WPA2 will, setzt `wpa=2`. `wpa_key_mgmt=WPA-PSK` legt das Verfahren der Schlüsselverwaltung fest. WPA-PSK steht dabei für das Verfahren des Pre-shared Keys (PSK)[14], also des vorgegebenen Netzwerkschlüssels. Weniger technisch ausgedrückt meint dies einfach das, was man vom üblichen Heim-WLAN kennt: Es wird ein einzelner WLAN-Schlüssel vergeben, der für jedermann gilt. Optional können mit diesem Verfahren auch mehrere zulässige WLAN-Schlüssel verwaltet werden (dies erfordert allerdings die Zurhilfenahme einer weiteren Option `wpa_psk_file`). Der andere erlaubte Wert für `wpa_key_mgmt` ist WPA-EAP, besser bekannt als »WPA Enterprise«, bei dem mit Benutzernamen und -passworten gearbeitet werden kann, was sich offenbar insbesondere an Universitäten großer Beliebtheit erfreut. Dessen Konfiguration ist allerdings weitaus komplexer und soll hier nicht Gegenstand sein.

`wpa_pairwise` legt die für WPA zu verwendenden Verschlüsselungsalgorithmen fest, `rsn_pairwise` diejenigen für WPA2. Zur Auswahl stehen das sicherere CCMP[15] und das für ältere Geräte geeignete TKIP[16] (oder auch beide), wobei Geräte, die WPA2 unterstützen, im Regelfall auch CCMP als Verschlüsselungsalgorithmus verstehen. Lediglich einige Windows-Treiber haben wohl Probleme mit der Kombination aus WPA und CCMP, sodass man beim Betrieb eines WPA-WLANs von dessen Verwendung absehen sollte. Da WPA nicht mehr als sicher angesehen wird[17] und üblicherweise WPA2 als Alternative empfohlen wird, sollte die Notwendigkeit für das angreifbare TKIP im Laufe der nächsten Zeit vermutlich absinken; noch gibt es aber wohl genügend Geräte, die WPA2 überhaupt nicht unterstützen. Die folgende Tabelle stellt die Kombination der einzelnen Direktiven noch einmal klar:

Verschlüsselungsmethoden		
wpa=n	**Verschlüsselung**	**Anwendbare Direktiven**
wpa=1	WPA	wpa_pairwise
wpa=2	WPA2	rsn_pairwise
wpa=3	WPA und WPA2	wpn_pairwise und rsn_pairwise

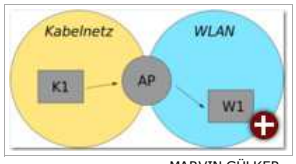
`wpa_passphrase` schließlich legt – wenig überraschend – den Netzwerkschlüssel fest, dessen Verwendung zuvor mittels `wpa_key_mgmt=WPA-PSK` angefordert wurde. Mittels `auth_algs=1` wird der Authentifizierungsalgorithmus als »open auth« festgelegt. Die Werte 2 und 3 sind nur für WEP-Verschlüsselung sinnvoll und können, da gegen WEP ohnehin genügend Angriffe bekannt sind[18], getrost außen vor gelassen werden.

Übrig bleiben nur noch einige Feineinstellungen. `logger_syslog=-1` instruiert hostapd, die Nachrichten aller Komponenten gleichermaßen in das Syslog zu schreiben, `logger_syslog_level=2` filtert Nachrichten, deren Wichtigkeit unter »info« liegt, heraus. Von größerem Interesse ist die Option `max_num_sta`, mit der angegeben wird, wie viele Clients sich maximal am AP anmelden können. Hier sollte man einen Wert wählen, der grob der Anzahl der erwarteten Nutzer entspricht und die Hardware (Raspberry Pi!) bei gleichzeitiger Nutzung des WLANs durch all diese Nutzer nicht überfordert; IEEE 802.11 legt als Obergrenze 2007 Clients fest. Über `deny_mac_file` lassen sich Clients, deren MAC-Adressen denen in der angegebenen Datei entsprechen, von der Nutzung des WLANs ausschließen.

Da MAC-Adressen sehr leicht zu fälschen sind, sollte hierin aber keinesfalls eine große Sicherheitsbarriere gesehen werden. Die Datei selbst enthält einfach nur eine zeilenweise Auflistung verbotener MAC-Adressen, etwa so:

```
# List of MAC addresses that are not allowed to authenticate (IEEE 802.11)
# with the AP.
00:20:30:40:50:60
00:ab:cd:ef:12:34
00:00:30:40:50:60
```

Damit ist die Konfiguration von hostapd abgeschlossen. Umfangreiche Dokumentation ist in Form einer Beispieldatei[19] sowie auf der Website von Linux[20] verfügbar.



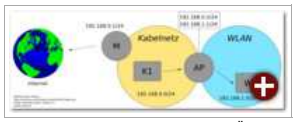
MARVIN GÜLKER

Grundlegender physikalischer Netzwerkaufbau

dnsmasq

Ein laufendes WLAN ist ja schön und gut, wenn allerdings kein DHCP-Server verfügbar ist, der IP-Adressen vergibt, hilft das wenig. Bei der Zuteilung von IP-Adressen an das WLAN gibt es grundsätzlich mehrere Möglichkeiten, die um das Problem kreisen, wie Pakete von der einen physikalischen Seite des Netzwerks (Kabelnetz) auf die andere (WLAN) geroutet werden können und umgekehrt. Von den unten vorgestellten Varianten kommt nur Möglichkeit 1 ohne Subnetz für das WLAN aus, da in allen anderen Fällen die Pakete zunächst den AP als Zwischenstation passieren müssen, wo sie bis zur IP-Ebene »ausgepackt« und dann auf dem neuen Interface wieder »eingepackt« und weitergeschickt werden (das ist die typische Routerfunktion). Daraus ergibt sich, dass ein Client im Kabelnetz K1 einen Client im WLAN-Netz W1 zwar über die IP adressieren, nicht aber direkt erreichen kann. K1 muss die Zwischenstation, den AP, kennen, um zu wissen, wohin er sein Paket zunächst versenden soll. Angelehnt an die Syntax von ip(8) könnte man sagen, er muss wissen, via welchen Knoten er das Paket versenden soll. Mit diesem Wissen wird das an W1 adressierte Paket zunächst zum AP geschickt, der seinerseits direkt sowohl K1 als auch W1 ohne weitere Zwischenstation erreichen kann.

1. Ein externer DHCP-Server (etwa das bekannte Multifunktionsgerät des Internetanbieters) übernimmt die Adressvergabe auch im WLAN, ohne eigenes Subnetz für dasselbe. Dies erfordert einen Bridge-fähigen WLAN-Chipsatz, der allerdings nur selten verfügbar ist.
2. Auf dem AP läuft ein DHCP-Daemon, der die Adressvergabe nur für das WLAN übernimmt. Der Rest des Netzwerks wird von einem anderen, schon vorhandenen DHCP-Daemon verwaltet. In dieser Variante würde man ein eigenes Subnetz für das WLAN verwenden, um K1 mitteilen zu können, wohin es die an WLAN-Clients adressierten Pakete überhaupt zunächst hin verschicken muss.
3. Auf dem AP läuft ein DHCP-Daemon, der die Adressvergabe für das gesamte Netzwerk, also sowohl das Kabel- als auch das WLAN-Netz, übernimmt. Auch hier würde man separate Subnetze für Kabel- und WLAN-Teil des Netzwerks wählen.



MARVIN GÜLKER

Netzwerkaufbau mit IPv4-Adressen

In Ermangelung spezialisierter WLAN-Bridge-Hardware stellt die zweite Möglichkeit den einfachsten gangbaren Weg dar. Nimmt man als kabelgebundenes Netz 192.168.0.0/24 sowie für das WLAN-Netz 192.168.1.0/24 an und geht ferner davon aus, dass sich das Multifunktionsgerät des Internetanbieters, das die Verbindung zum Internet ermöglicht, im Kabelnetzwerk befindet und auf die Adresse 192.168.0.1/24 hört, so ergibt sich das in der Abbildung oben auf dieser Seite dargestellte erweiterte Schema. Da der Access Point als Übergangspunkt zwischen zwei Netzen fungiert, benötigt er zwei IP-Adressen, je eine aus einem Netz. Konkret auf den Raspberry Pi bezogen ließe sich dieses Schema folgendermaßen umsetzen:

```
# ip addr add 192.168.0.2/24 dev eth0
# ip addr add 192.168.1.1/24 dev wlan0
```

Mit diesem Vorwissen kann die Konfiguration von dnsmasq, einem DHCP-Daemon, in Angriff genommen werden. Der Dienst wird über die Datei **/etc/dnsmasq.conf** konfiguriert, die wie folgt aussehen könnte:

```
# Target interface
interface=wlan0

# DHCP range for Wifi
dhcp-range=192.168.1.2,192.168.1.254,6h

# DNS
dhcp-option=option:dns-server,192.168.0.1

# log-dhcp
```

Die erste Direktive ist relativ selbsterklärend; sie legt das Interface fest, auf dem dnsmasq lauschen soll. Lässt man sie ganz weg, lauscht dnsmasq auf allen verfügbaren Interfaces. Mit der nächsten Direktive wird der Adressraum angegeben, aus dem der Daemon IPv4-Adressen vergibt. Hier sollte man darauf achten, sowohl die reservierten Adressen für Netzwerk und Broadcast (192.168.1.0, 192.168.1.255) als auch die Adresse auszunehmen, die der AP im WLAN-Netz für sich selbst beansprucht (192.168.1.1), um unnötige Prüfungen auf Adresskollisionen zu vermeiden.

dnsmasq ist ein ausgesprochen vielseitiger Daemon. Neben der bloßen Funktion als DHCP-Server kann er unter anderem auch dazu verwendet werden, DNS-Dienste und IPv6-Router-Advertisements bereitzustellen. Da der vorliegende Artikel sich jedoch ausschließlich mit der Bereitstellung eines funktionsfähigen WLANs befasst, soll diese erweiterte Konfiguration Aufgabe des Lesers sein; für den Moment soll es ausreichen, den anfragenden WLAN-Clients als DNS-Server das Multifunktionsgerät des

Internetanbieters zu empfehlen, was mithilfe der Direktive `dhcp-option` geschieht. Das DHCP-Protokoll unterstützt eine ganze Reihe von Optionen, die an die Clients übermittelt werden und die über die bloße Vergabe von IP-Adressen weit hinausgehen; alle offiziell von dnsmasq unterstützten Optionen lassen sich mithilfe des Befehls

```
$ dnsmasq --help dhcp
```

mit Optionsnummer und -name ausgeben. Die im Beispiel auskommentierte Direktive `log-dhcp` veranlasst dnsmasq, die an jeden einzelnen Client versendeten DHCP-Optionen detailliert im Syslog zu protokollieren; wer nachschaut, wird feststellen, dass der Daemon schon standardmäßig eine ganze Reihe von Optionen sendet, von denen insbesondere die netzwerkspezifischen herauszuheben sind:

```
[...]
dnsmasq-dhcp[4725]: 3013853912 sent size:  4 option:  1 netmask 255.255.255.0
dnsmasq-dhcp[4725]: 3013853912 sent size:  4 option: 28 broadcast 192.168.1.255
dnsmasq-dhcp[4725]: 3013853912 sent size:  4 option:  3 router 192.168.1.2
dnsmasq-dhcp[4725]: 3013853912 sent size:  4 option:  6 dns-server  192.168.0.1
[...]
```

Solange dnsmasq mithilfe der `dhcp-option`-Direktive nichts Abweichendes mitgeteilt wird, geht der Dienst davon aus, dass er selbst auf dem für die Clients maßgeblichen Router läuft. Daraus ergibt sich, dass die für die Clients notwendigen Routingoptionen wie die Netzmaske/Präfixlänge (`netmask`, hier 255.255.255.0 = /24), der Broadcast des Netzes (`broadcast`, hier 192.168.1.255) und ganz besonders der Default-Gateway (`router`, hier 192.168.1.2) alle der Netzwerkkonfiguration des Rechners entnommen werden, auf dem dnsmasq läuft. Daher entsprechen die oben gezeigten Optionen bis auf die manuell gesetzte Option 6 (`dns-server`) auch haargenau der Netzwerkkonfiguration des Raspberry Pi. Die so übersendeten DHCP-Optionen werden vom DHCP-Client auf dem Client-Rechner ausgewertet, der sich um die Konfiguration des Netzwerkinterfaces des Clients kümmert.

Die vollständige Dokumentation zur Konfigurationsdatei von dnsmasq kann dessen Manpage dnsmasq(8) entnommen werden.

Das Multifunktionsgerät

Auch das bisher nur am Rande vorgekommene Multifunktionsgerät des Internetanbieters muss ein wenig umkonfiguriert werden, um ihm mitzuteilen, wie es die Clients in Kabel- und WLAN-Netz erreichen kann. Jedes etwas bessere Gerät bietet die Möglichkeit, die fest zugewiese IP und deren Präfixlänge zu verändern sowie statische Routen zu hinterlegen. Da die Clients im Kabelnetzwerk ihre IP-Adresse nicht vom AP, sondern vom Multifunktionsgerät beziehen, besteht für sie keine Möglichkeit zu wissen, wohin sie Pakete für das WLAN-Netz schicken sollen. Sie senden, sofern nicht jeder Client einzeln entsprechend konfiguriert wird, alles an ihr Standardziel (default gateway), in diesem Falle das Multifunktionsgerät. Es ist daher zweckmäßig, diesem beizubringen, an das WLAN-Subnetz adressierte Pakete an den AP zu routen, der diese dann entsprechend weiterleiten kann. Da die Weboberflächen dieser Geräte von Hersteller zu Hersteller denkbar verschieden aufgebaut sind, können an dieser Stelle leider keine genauen Hinweise gegeben werden, wie dies zu bewerkstelligen ist; oftmals muss man einen »Experten-« oder »erweiterten Modus« anwählen, bevor man die »sensiblen« Einstellungen betreffend das Routing verändern kann. Zunächst ist als neue statische Adresse und Präfixlänge 192.168.0.1/24 zu wählen. Für ältere Router, die statt nach Präfixlängen nach den veralteten Subnetzmasken fragen, kann man eine entsprechende Konvertierungstabelle[21] bemühen. Sodann fügt man eine einzelne statische Route ein, die alle an das WLAN-Netz adressierten Pakete an den AP weiterleitet. Das entsprechende Shell-Kommando würde so aussehen:

```
# ip route add 192.168.1.0/24 via 192.168.0.2
```

Als Zieladresse ist - logischerweise - die Kabelnetz-Adresse des Access Points zu wählen. Auch hier kann natürlich zur Konvertierung Präfixlänge/Subnetzmaske eine Tabelle bemüht werden.

dnsmasq teilt sich wie zuvor schon beschrieben den WLAN-Clients ohnehin selbst als Default-Gateway mit, sodass eine Konfiguration der einzelnen WLAN-Clients hinsichtlich der Routen unnötig ist. Der AP fungiert für die WLAN-Clients sowohl als Gateway zum Internet als auch zum Kabelnetz. Umgekehrt senden die Kabel-Clients ihre Pakete alle an das Multifunktionsgerät des Internetanbieters, auf dem gerade eben die neue statische Route hinterlegt wurde; erreicht dieses ein an einen WLAN-Client adressiertes Paket, so wird es dieses an 192.168.0.2, den AP, weiterleiten, der weiß, wohin es weiter zu übermitteln ist. Darüber hinaus ist es allerdings wichtig zu wissen, dass ohne die Hinterlegung der statischen Route im Multifunktionsgerät ein nicht unerhebliches Routingproblem entstünde: Zwar würden Pakete aus dem WLAN-Netz korrekt ins Internet geroutet, die Antwort des entfernten Internetservers jedoch würde nur bis zum Multifunktionsgerät gelangen, das nicht wüsste, wohin mit dem unbekannten Subnetz. Die Pakete würden verworfen und der WLAN-Client würde nie eine Antwort auf seine Anfragen erhalten.

Nicht vergessen werden sollte auch, den DHCP-Server des Multifunktionsgeräts so einzustellen, dass er keine Adressen im Bereich des DHCP-Servers auf dem AP, d.h. keine Adressen im Raum 192.168.1.0/24, vergibt.

Services

Die Startreihenfolge von dnsmasq und hostapd ist nicht ganz willkürlich. dnsmasq verweigert schlicht den Dienst am Interface, wenn sich dieses nicht in irgendeinem aktiven Zustand befindet. Unglücklicherweise hängt (zumindest unter Arch Linux) die Service-Datei von dnsmasq in keinster Weise von derjenigen von hostapd ab, sodass die Funktionalität des DHCP-Servers beim Systemstart ein bloßes Zufallsprodukt ist. Um dem Abhilfe zu schaffen, muss die Servicedatei von dnsmasq so angepasst werden, dass sie auf hostapd Rücksicht nimmt. Dazu wird zunächst die Standard-Service-Datei in den Admininistrationsbereich kopiert (die Dateien in **/etc** haben für Systemd Vorrang vor denen in **/usr**):

```
# cp /usr/lib/systemd/system/dnsmasq.service /etc/systemd/system
```

Die so neu gewonnene Datei namens **/etc/systemd/system/dnsmasq.service** wird um die Zeile `After=hostapd.service` im Abschnitt »Unit« ergänzt, sodass sie im Ergebnis wie folgt lautet:

```
[Unit]
Description=A lightweight DHCP and caching DNS server
After=network.target
After=hostapd.service
Documentation=man:dnsmasq(8)

[Service]
Type=dbus
BusName=uk.org.thekelleys.dnsmasq
ExecStartPre=/usr/bin/dnsmasq --test
ExecStart=/usr/bin/dnsmasq -k --enable-dbus --user=dnsmasq --pid-file
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
Alias=dbus-uk.org.thekelleys.dnsmasq
```

Nun können die Dienste in den Systemstart eingetragen und gestartet werden:

```
# systemctl enable hostapd
# systemctl enable dnsmasq
# systemctl start hostapd
```

```
# systemctl start dnsmasq
```

Läuft alles gut, so kann nun das neue, hausgebastelte WLAN verwendet werden.

Weiterführendes

Dieser Artikel beschreibt lediglich eine Minimalkonfiguration. Mit einem voll funktionstüchtigen AP lässt sich jedoch noch sehr viel mehr machen; so kann etwa ein SixXS-Tunnelendpunkt für IPv6[22] verwaltet und so das heimische Netz mit IPv6 ausgestattet, dnsmasq um DNS- und IPv6-Router-Advertisement-Funktionalität erweitert oder OpenVPN als sicherer Hafen für Fernverbindungen ins heimische Netz betrieben werden. Viele weitere Anwendungen sind denkbar, die nun nicht mehr von der verschlossenen Hardware des Internetanbieters blockiert werden. Es lebe die freie und quelloffene Software!

Autoreninformation

Marvin Gülker (Webseite[23]) ist Jura-Student, Programmierer und höchst interessiert an allem, was mit Netzwerken unter Linux zu tun hat.

Dieser Artikel ist in freiesMagazin 03/2014[24] (ISSN 1867-7991) erschienen. Veröffentlichung mit freundlicher Genehmigung.

Linkverweise:

[1] mailto:quintus@quintilianus.eu
[2] http://www.raspberrypi.org/
[3] http://wireless.kernel.org/en/users/Drivers
[4] http://hostap.epitest.fi/hostapd/
[5] http://www.thekelleys.org.uk/dnsmasq/doc.html
[6] https://de.wikipedia.org/wiki/Wi-Fi_Protected_Access
[7] https://de.wikipedia.org/wiki/RADIUS_%28Protokoll%29
[8] https://de.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
[9] https://en.wikipedia.org/wiki/List_of_WLAN_channels
[10] http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/2013_10_WLAN_2,4GHz_pdf.pdf?__blob=publicationFile&v=4
[11] http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/2010_07_WLAN_5GHz_pdf.pdf?__blob=publicationFile&v=3
[12] http://dejure.org/gesetze/TKG/149.html
[13] http://dejure.org/gesetze/TKG/149.html
[14] https://de.wikipedia.org/wiki/Pre-shared_key
[15] https://de.wikipedia.org/wiki/CCMP
[16] https://en.wikipedia.org/wiki/Temporal_Key_Integrity_Protocol
[17] https://de.wikipedia.org/wiki/Wi-Fi_Protected_Access#Angriffsm.C3.B6glichkeiten
[18] https://de.wikipedia.org/wiki/Wired_Equivalent_Privacy#Schwachstellen
[19] http://hostap.epitest.fi/cgit/hostap/plain/hostapd/hostapd.conf
[20] http://wireless.kernel.org/en/users/Documentation/hostapd
[21] http://www.rjsmith.com/CIDR-Table.html
[22] https://de.wikipedia.org/wiki/Liste_von_IPv6-Tunnelbroker
[23] http://www.quintilianus.eu/
[24] http://www.freiesmagazin.de/20140302-maerzausgabe-erschienen