Lokalisierung durch Messung von WLAN-Signallaufzeiten

Mario Haustein

Chemnitzer Linux-Tage

19. März 2011

Anliegen

- Wie funktioniert eine Laufzeitmessung überhaupt?
- Wie geht das unter Linux?
- Welche Software benötigt man dazu?
- Sind Patches notwendig? Wenn ja wo?

Motivation

- GPS kann in Gebäuden nicht zur Lokalisierung eingesetzt werden.
- WLAN ist eine weit verbreitete Infrastruktur.

$\Rightarrow$ Nutzen WLAN zur Lokalisierung.

- Elektromagnetische Wellen breiten sich mit Lichtgeschwindigkeit c aus.

$$r = c \cdot t$$

- Diese Korrelation ist sehr stark.
- Feldstärken-Werte sind ebenfalls mit der Entfernung korreliert.
  - Sie werden aber durch bauliche Gegebenheiten stark gestört.

- Es gab bereits ähnliche Ansätze.[1]

[1] vgl. André Günther; Christian Hoene: Measuring Round Trip Times to Determine the Distance Between WLAN Nodes In: *NETWORKING 2005*. S. 768–779

Anforderungen

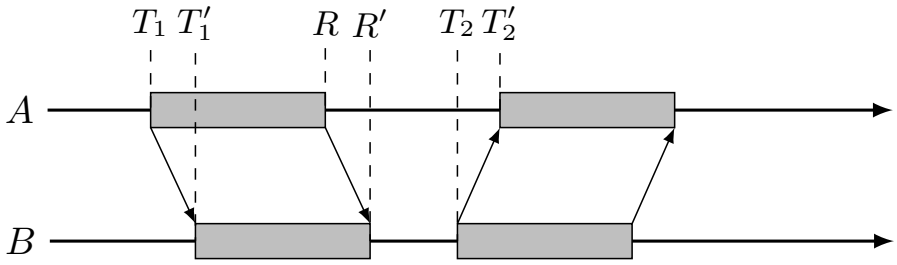
Ziel: Weite Verbreitungsmöglichkeit sicherstellen.

- Keine Spezial-Hardware $\Rightarrow$ „Off-the-shelf“
- Keine Hardware-Modifikationen
- Keine Firmware-Modifikationen

- Reine Software-Lösung

Wie misst man die Signallaufzeit?

- Time-of-Flight-Verfahren: Nachrichtenaustausch $A \rightarrow B \rightarrow A$



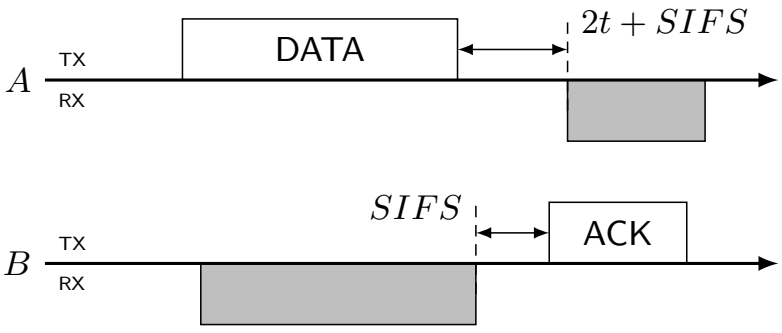
- Es ist keine Uhrensynchronisation notwendig.

$$\Delta = T_2' - T_1 = \underbrace{(T_1' - T_1)}_{t} + \underbrace{(R' - T_1')}_{d} + \underbrace{(T_2 - R')}_{g} + \underbrace{(T_2' - T_2)}_{t} = 2t + d + g$$

- Nach Umstellen erhält man: $t = \frac{1}{2} \cdot (\Delta - d - g)$

Die IEEE 802.11 MAC-Schicht

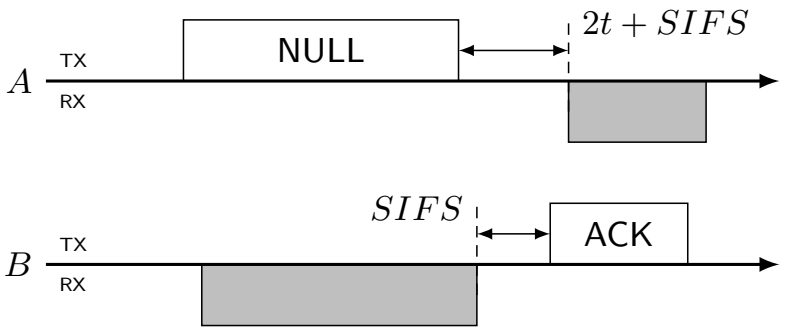
- IEEE 802.11 stellt den Zustellung von Frames bereits auf MAC-Ebene sicher.



- Dieser Mechanismus stellt eine TOF-Sequenz dar.
  - Wurde bereits in der Vergangenheit zur Lokalisierung genutzt.

Das NULL-ACK-Verfahren

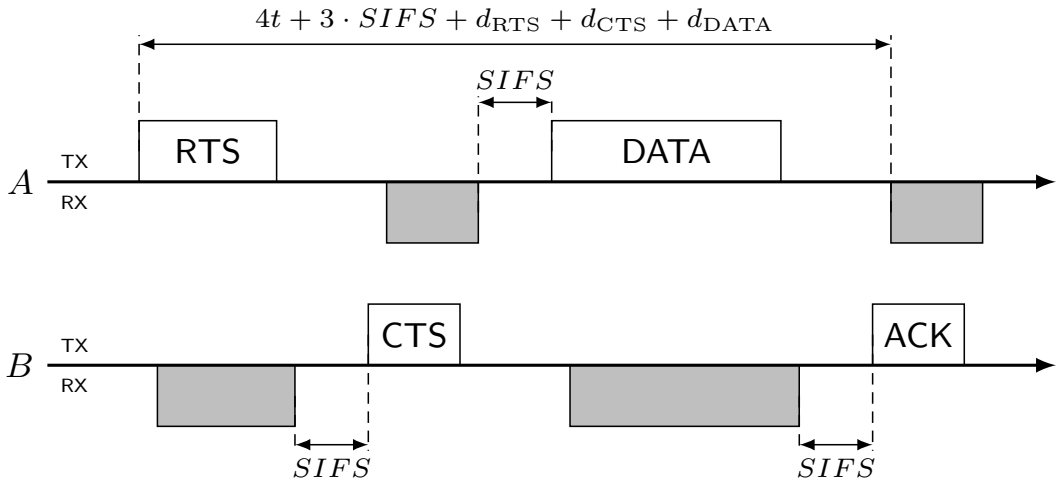
- NULL-Frames transportieren keine Nutzlast.
- Sie dienen der Umsetzung von Stromsparfunktionen.



- ⊕ Keine Assoziierung mit dem Access Point notwendig
- ⊕ Keine Belastung des Netzwerks hinter dem Access Points
- ⊕ Keine Beeinflussung durch Verschlüsselung
- ⊖ Erzeugung nicht so einfach wie bei DATA-ACK-Sequenzen

Das RTS-CTS-NULL-ACK-Verfahren

- RTS-CTS-Handshakes regulieren den Medienzugriff.



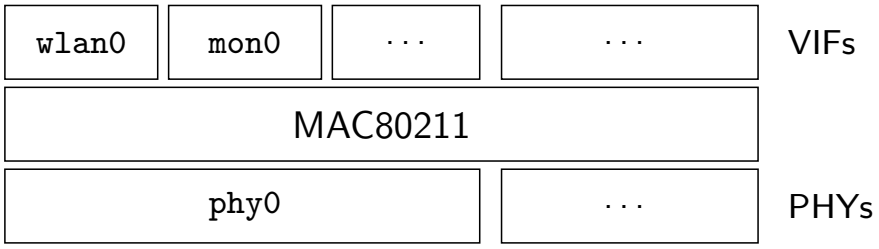
- Signallaufzeit t geht vier statt zwei mal ein.

$\Rightarrow$ Müssen Möglichkeit schaffen, ein RTS-CTS-Handshake auszuführen.

MAC80211

- Mit Version 2.6.22 in den Linux-Kernel aufgenommen.
- Verwaltet u.a. Netzwerkinterfaces (VIF) von Wireless LAN Adaptern (PHY).
- Die Treiber erfüllen nur noch Backend-Funktionen.

```
$ iw phy phy0 interface add wlan0 type managed
$ iw phy phy0 interface add mon0  type monitor
```



- Jedes VIF arbeit in einem bestimmten Modus
  - In den Modi `managed`, `ibss` und `master` werden Ethernet-Frames ausgetauscht.
  - Im Modus `monitor` werden Radiotap-Frames ausgetauscht.

Monitor-Schnittstellen

- Am Monitor-Interface können alle eingehenden Frames beobachtet werden.
- Sogar die Gegenrichtung ist möglich: Einschleusen (manipulierter) Frames.

Das Radiotap-Protokoll

- Austausch von Metainformationen[a] zwischen Anwendung und MAC80211
  - Kanal, TSF-Zeitstempel
  - Bitrate
  - TX-Flags

[a] vgl. `http://www.radiotap.org/`

Der TSF-Zähler

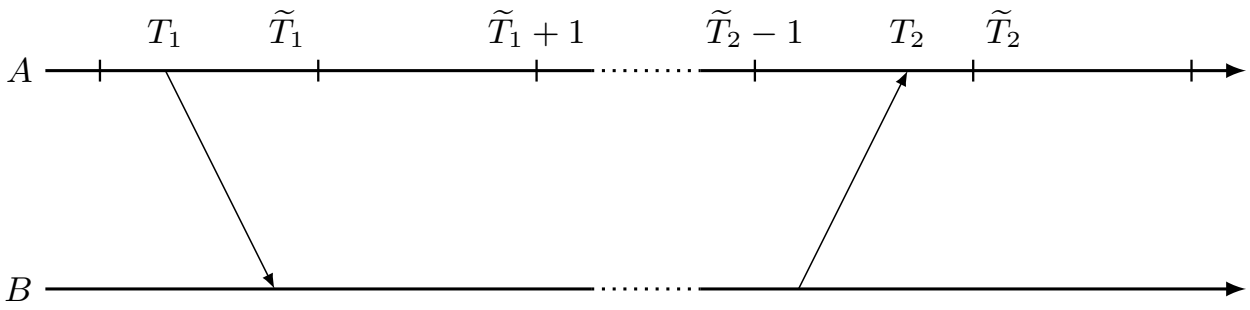
- Nach Kapitel 1.11 in IEEE 802.11 in jedem WLAN-Adpater vorhanden.
- Viele Treiber vermerken für empfangene Frames den Wert des TSF-Zählers zum Zeitpunkt des Empfangs im Radiotap-Header.
- Auflösung: $1\,\mu s$, Kein Jitter!

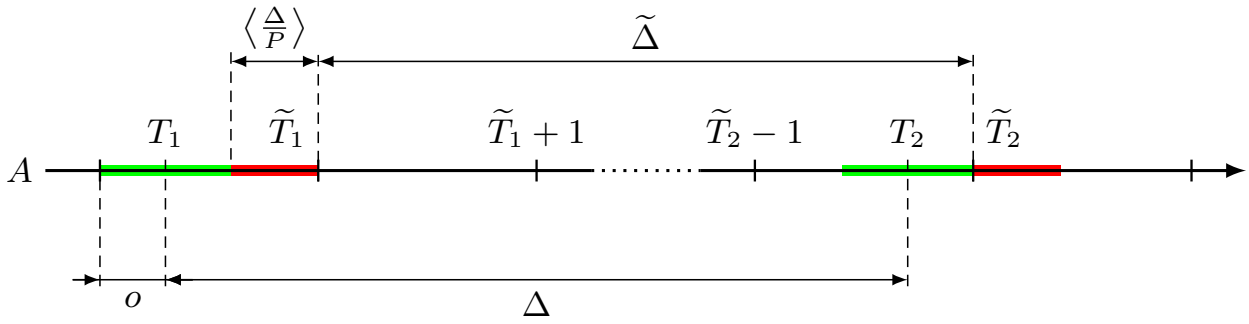
Frage

Die Auflösung des TSF reicht doch für eine nanosekundengenaue Messung gar nicht aus, oder?

Antwort

Doch! Wir müssen nur oft genug messen.





- $\widetilde{\Delta}$ hängt vom Offset o ab.
- Ist o gleichverteilt, folgt $\widetilde{\Delta}$ einer Zweipunktverteilung (P ... Auflösung).

$$\widetilde{\Delta} = \begin{cases} \left\lfloor \frac{\Delta}{P} \right\rfloor & 0 < \frac{o}{P} < 1 - \left\langle \frac{\Delta}{P} \right\rangle \\ \left\lfloor \frac{\Delta}{P} \right\rfloor + 1 & 1 > \frac{o}{P} \geq 1 - \left\langle \frac{\Delta}{P} \right\rangle \end{cases}$$

- Es gilt:

$$E[\widetilde{\Delta}] = \Delta$$

Laufzeitschätzung

- Der Schätzer

$$\widehat{\Delta} = \frac{P}{n} \sum_{i=1}^{n} \widetilde{\Delta}_i$$

  ist erwartungstreu:

$$E[\widehat{\Delta}] = \Delta$$

- Die Genauigkeit hängt von n und P ab:

$$\sqrt{D^2[\widehat{\Delta}]} = \frac{P}{\sqrt{n}} \cdot \sqrt{\left\langle \tfrac{\Delta}{P} \right\rangle \cdot \left(1 - \left\langle \tfrac{\Delta}{P} \right\rangle\right)} \leq \frac{P}{2\sqrt{n}}$$

- Bereits ab $n = 1000$ erhält man brauchbare Schätzungen.

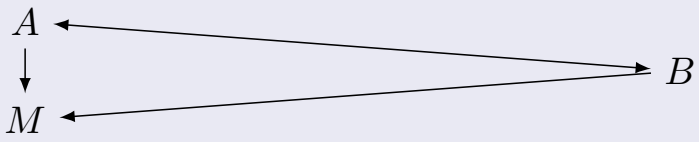
Die Notwendigkeit eines Monitor-Adapters

Problem

- *Die TOF-Methode benötigt Sendezeitstempel.*
- *Sendezeitstempel können nicht ohne Weiteres erfasst werden.*

Lösung

- *Ein zweiter Monitor-Adapter in Nähe von A*



$\Rightarrow$ *Bei M müssen nur Empfangsereignisse gemessen werden.*

$\Rightarrow$ *Die Uhr von M erfüllt die notwendige Unabhängigkeit.*

Wie kommt man an die Zeitstempel heran?

- Die Zeitstempel stehen im Radiotap-Header.
- Zunächst ein Monitor-VIF für M anlegen ...

```
$ ip link set ⟨Interface⟩ up
$ iw dev ⟨Interface⟩ set channel ⟨Kanal⟩
$ iw dev ⟨Interface⟩ interface add monif type monitor
$ ip link set monif up
```

- ... dann Wireshark starten

```
$ tshark -i mon -w mon.pcap -f \
  "(wlan addr1 ⟨B⟩ and wlan addr2 ⟨A⟩ and subtype null) \
   or (wlan addr1 ⟨A⟩ and subtype ack)"
```

- Die NULL-ACK-Sequenzen sind jetzt in `mon.pcap` gespeichert.

Keine Patches notwendig

Und wie kommen die NULL-ACK-Sequenzen zu Stande?

- NULL-Frames können über Monitor-VIFs eingeschleust werden.
- Im Radiotap-Header wird festgelegt:
  - Bitrate
  - RTS-CTS-Handshake
- Notwendige Kernel-Patches:
  - Bitrate für eingeschleuste Frames[2] zzgl. Korrekturpatch
  - RTS-CTS-Handshake vor eigentlichem Frame ausführen
  - Umgehung treiberspezifischer Rate-Control-Algorithmen
  - ...
- Es wird wieder ein Monitor-VIF für A benötigt.

```
$ ip link set ⟨Interface⟩ up
$ iw dev ⟨Interface⟩ set channel ⟨Kanal⟩
$ iw dev ⟨Interface⟩ interface add injif type monitor
$ ip link set injif up
```

[2] https://patchwork.kernel.org/patch/118564/

Und wie kommen die NULL-ACK-Sequenzen zu Stande?

- Zum Einschleusen von Frame kann libpcap genutzt werden.

```
void inject(char *iface, unsigned int cnt, uint8_t *frame,
            unsigned int len, unsigned long delay)
{
  char errbuf[PCAP_ERRBUF_SIZE];
  pcap_t *pcap;
  int result;
  unsigned int i;

  strcpy(errbuf, "");
  pcap = pcap_open_live(iface, 800, 1, 20, errbuf);

  for(i = 0; i < cnt; i++)
  {
    result = pcap_inject(pcap, frame, len);
    usleep(delay);
  }

  pcap_close(pcap);
}
```

[2] https://patchwork.kernel.org/patch/118564/

Die Auswertung von NULL-ACK-Sequenzen

- Hier kann wieder Wireshark eingesetzt werden (Export als CSV-Format).

```
$ tshark -r mon.pcap -T fields -E header=n -E separator=, \
  -e frame.number            -e frame.len              \
  -e radiotap.mactime        -e radiotap.datarate      \
  -e wlan.fc.type_subtype    -e wlan.ta                \
  -e wlan.sa                 -e wlan.da                \
  -e wlan.bssid              ...
```

- Weiteres Vorgehen
  1. NULL-ACK-Sequenzen identifizieren.
  2. $\widetilde{\Delta}$ berechnen.
  3. Messfehler filtern.

Keine Patches notwendig

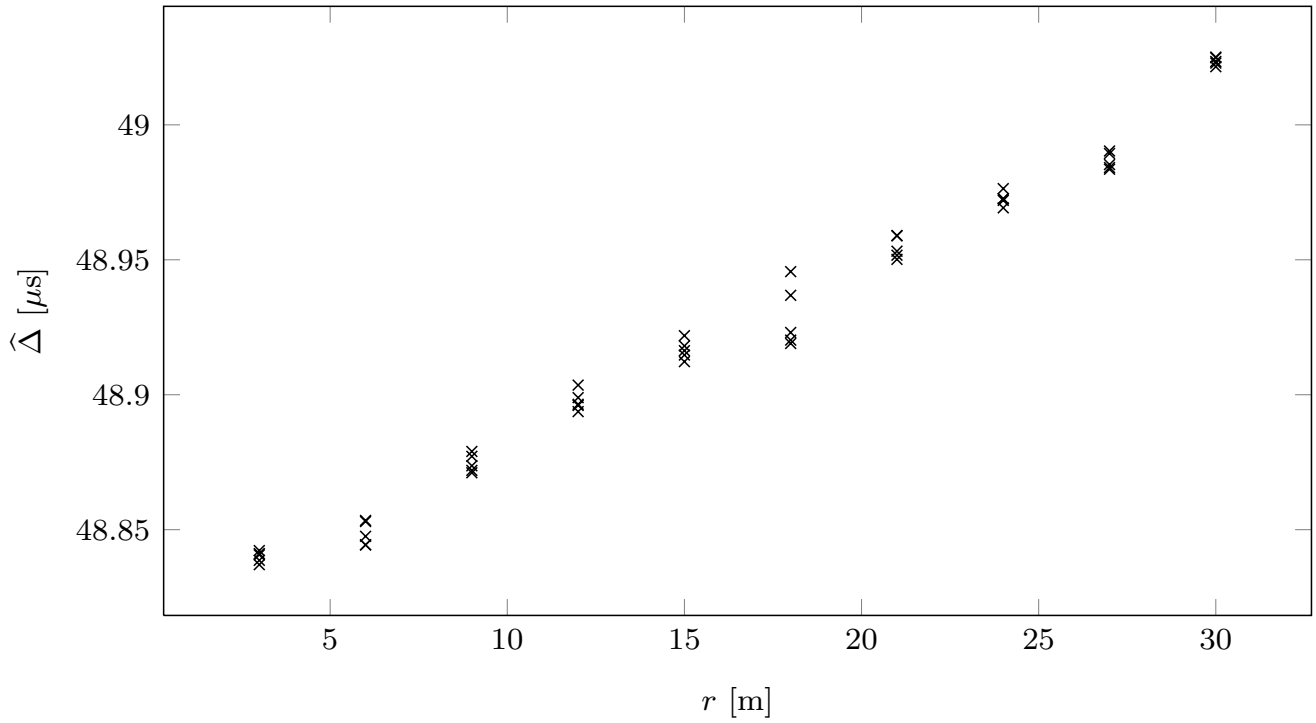
Die Software-seitige Umsetzung

- Aktuell nur Proof-of-Concept-Code.
  - Einfaches C-Programm `nullinject`
  - Lua-Framework zum Erfassen, Erzeugen und Auswerten der NULL-ACK-Sequenzen
- Framework
  - Ansammlung von Hilfsfunktionen
  - Bedienung über interaktive Lua-Shell

```
$ cd messtool
$ lua -i tool.lua
Lua 5.1.4  Copyright (C) 1994-2008 Lua.org, PUC-Rio
> mo = proc.nullack:prepare("wlan2", { "wlan0", "wlan1" }, 13)
> mo:measure("/tmp/messung", "⟨MAC v. B⟩", "⟨BSSID⟩",
    10000, 54, 1000)
10000 packets ⟨A⟩ -> ⟨B⟩ via na_ini (bssid ⟨BSSID⟩)
10000 packets transmitted
> mo:cleanup()
> ^D
$
```

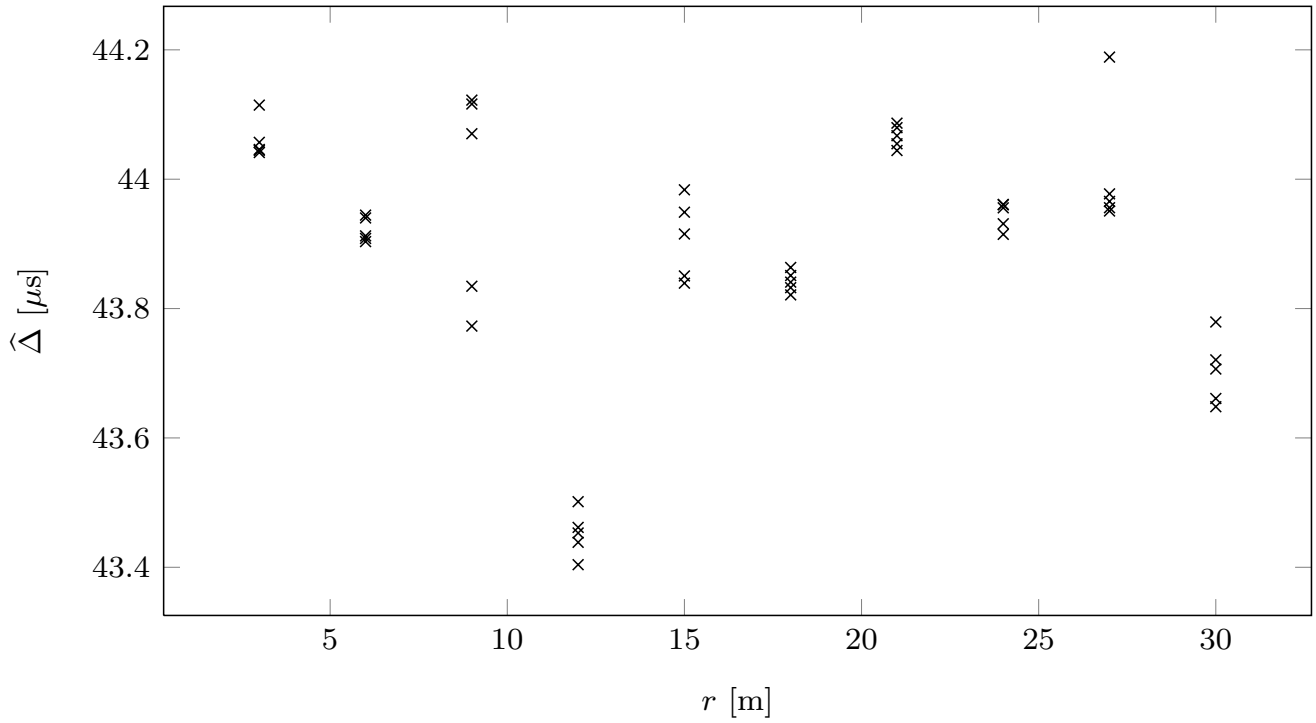
Versuchsergebnisse

54 MBit/s, Atheros-USB-Stick (TL-WN422G), ca. 7000 bis 9000 Einzelmessungen pro Schätzung



Versuchsergebnisse

54 MBit/s, Intel-Karte (IWL6000), ca. 7000 bis 9000 Einzelmessungen pro Schätzung



Ausblick

- Eine Software-basierte Laufzeitmessung ist möglich!
  - Nicht jede Hardware-Plattform ist dafür geeignet.
- Technische Schwierigkeiten erfordern ein zweites WLAN-Gerät
  - Kann evt. durch Firmwaremodifikationen behoben werden.
  - Treiber mit quelloffener Firmware: `carl9170`[3]
- In baulich schwierigen Umgebungen werden WLAN-Signale reflektiert und haben einen längeren „Funkweg“.
- IEEE 802.11y setzt die Laufzeitmessung in Zukunft Hardware-seitig um.

[3] `http://linuxwireless.org/en/users/Drivers/carl9170`