Wireless Fidelity with bwfm(4)

Patrick Wildt

September 22, 2019
Who am I?

- OpenBSD developer
- ARM64-subtree maintainer
- LLVM-subtree updater
- SBC hoarder
Collection of devices

- Cubox-i
- Macbook
- Raspberry Pi 3
- Z83 Mini-PC
<table>
<thead>
<tr>
<th>Milestones</th>
<th>May 2016</th>
<th>June 2016</th>
<th>September 2017</th>
<th>October 2017</th>
<th>December 2017</th>
<th>January 2018</th>
<th>February 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skeleton Driver</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB Firmware Upload</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WiFi Scan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenBSD commit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host AP Mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCIe backend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDIO backend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Study

1. Find documentation
   - Search the web for datasheets (by chip name)
   - git grep in various OS (chip name, vendor/product ID)
   - Neither code nor datasheet? Quit now.
   - Alternative: reverse engineering

2. Study code and/or documentation to grasp concepts
   - Attention: license concerns!

3. Realize it’s going to be a long project
Full vs Soft (simplified)

**Linux**

- Network Layer
  - nl80211
- Configuration Layer
  - cfg80211
- MAC Layer
  - FullMAC
    - mac80211
  - SoftMAC

**OpenBSD**

- Network Layer
  - net80211
- Software MAC Layer
  - SoftMAC

Patrick Wildt

Wireless Fidelity with bwfm(4)
ISC-licensed brcm80211 drivers (Linux):

<table>
<thead>
<tr>
<th>bcmfmac</th>
<th>brcmsmac</th>
</tr>
</thead>
<tbody>
<tr>
<td>FullMAC</td>
<td>SoftMAC</td>
</tr>
<tr>
<td>35 496 LoC</td>
<td>75 177 LoC</td>
</tr>
</tbody>
</table>

brcmsmac/phy/phy_n.c: 28 624 Lines Of Magic
Jobs

What do we **not** have to do?
- No beacons
- No frequency changes
- No MCS handling

What do we have to do?
- Initiate scan
- Configure SSID
- Configure keys
- Handle events
- Handle network packets
Patrick Wildt

Wireless Fidelity with bwfm(4)
Dongle

1. Started with SDIO
   - But realized testing kernels will take too long
   - Unsure if SDIO layer actually worked

2. Bought a USB device

3. Started with the lower layers

4. Added PCIe/SDIO backend later
Write code that compiles

1. Skeleton-driver
2. Initialize bus access
3. Try to figure out whether the device is alive
   - read chip id
   - read MAC address
   - receive an interrupt
Initiate Scan:

```c
struct bwfm_escan_params *params;
[...]
bwfm_fwvar_var_set_data(sc, "escan", 
    params, params_size);
```

Connect to SSID:

```c
struct bwfm_ext_join_params *params;
[...]
bwfm_fwvar_var_set_data(sc, "join", 
    params, sizeof(*params));
```
BCDC Packets

For Data packets:
- Flags
- Priority
- Flags2
- Data Offset
- Firmware
- Signals
- Ethernet Destination Mac
- Ethernet Source Mac
- Ethertype
- Data Payload

For Events:
- Flags
- Priority
- Flags2
- Data Offset
- Firmware
- Signals
- Ethernet Destination Mac
- Ethernet Source Mac
- Ethertype 0x886c
- Event

... type status reason ...

... event-specific payload
Why?
How?
What now?

SDIO

Configuration
FIFO
Data
Events

Patrick Wildt
Wireless Fidelity with bwfm(4)
SDIO Interrupt

- Shared pin: DAT[1]/IRQ
- Sampled as IRQ during *Interrupt Period*
- Some host controllers have troubles
- Workaround: externally routed GPIO
PCle

- Packet-based
- Multiple Ringbuffers
  - TX Control Ring
  - TX RX-Post Ring
  - (Control, TX, RX) Complete Rings
  - n Flowrings
- Read/write access to backplane
- Write Firmware & NVRAM
- Turn on/off ARM core
- Read dmesg
hndarm_armr addr: 0x18002000, cr4_idx: 0
000000.001 RTE (SDIO–MSG_BUF) 7.35.180.119 (r594535) on BCM4350 r8 @ 37.4/240.8/240.8MHz
000000.001 allocating a max of 255 rxcpid buffers
000000.002 pciemsgbuf0: Broadcom PCIE MSGBUF driver
000000.003 reclaim section 0: Returned 59036 bytes to the heap
000000.131 enable 1: q0 frmcnt 0, wrdcnt 0, q1 frmcnt 0, wrdcnt 0
000000.131 enable 1: q0 frmcnt 0, wrdcnt 0, q1 frmcnt 0, wrdcnt 0
000000.175 wl0: Broadcom BCM4350 802.11 Wireless Controller 7.35.180.119 (r594535)
000000.175 TCAM: 256 used: 255 exceed: 0
000000.176 reclaim section 1: Returned 147512 bytes to the heap
000005.375 wl0: wlc-enable-probe-req: state down, deferring setting of host flags
000005.413 wlc_bmac_switch_macfreq: 4350 need fix for 37.4Mhz
000005.421 wl0: wlc-enable-probe-req: state down, deferring setting of host flags
000005.421 enable 1: q0 frmcnt 0, wrdcnt 0, q1 frmcnt 0, wrdcnt 0
Firmware Features


Version: 7.35.101.6 (r702795)
CRC: 4f3f65c5
Date: Sun 2017-06-04 16:51:38 PDT
Ucode Ver: 963.316
FWID: 01-5e8eb735
Tricky bits

- Flow-control
- Asynchronous control messages
- Asynchronous creation of flowrings
- net80211 Integration
Remote Control Message Injection (CVE-2016-0801): Updated firmware in November 2017

KRACK (October 2017): Updated firmware in June 2018

(based on linux-firmware.git)
KRACK

/*
 * The firmware supplicant can handle the WPA
 * handshake for us, but we honestly want to
 * do this ourselves, so disable the firmware
 * supplicant and let our stack handle it.
 */

bwfm-fwvar-var-set-int(sc, "sup_wpa", 0);
NVRAM

Purpose:
- Provides configuration for the specific package
- Sets up antenna configuration, max dB, etc.

Needed on:
- PCIe (sometimes)
- SDIO (always)
- USB (not yet?)

Provided by:
- Hardware designer (in their git repo)
- EFI BIOS (in an EFI variable)
Current Status

- Works as client
- Properly fast 802.11ac (Wi-Fi 5)
- Implemented on recent Macbooks
- Implemented on raspberry Pis
- Available as *official raspberry Pi USB Dongle* (while supplies last)
- Works as access point often enough
Future

- Better AP support
- Multi-AP support
- Suspend/Resume
- Firmware Signals
- Support for more devices
Questions?