Reliable Linux Wireless - Techniques for Debugging Wireless Module Integrations

STEVE DEROSIER / CAL-SIERRA CONSULTING
• 50 minutes?!
• How to work with WiFi modules with the Linux kernel - especially misbehaving ones!
• Overview of Linux WiFi basic concepts - chips and stack
• Interfacing of devices
• Debugging tools
• HOW TO GET HELP!
Why Steve deRosier?

- Steve deRosier - Principle Consultant at Cal-Sierra Consulting LLC
  - 10 years working with WiFi and the Linux-Wireless community
  - 17 years working with Embedded Linux
  - Contributed to OSS in many projects: binutils, buildroot, linux-wireless, linux-mtd, ALSA, others...
  - Linux-wireless: ath6kl, libertas_tf, o11s, brcmfmac, others...
  - Special knowledge of Laird’s Linux wireless platforms
• Full-MAC vs Soft-MAC
• Firmware
• Interfaces
• Other pins
• BT coexistence
WiFi Chips

- **Full-MAC vs Soft-MAC**
- Firmware
- Interfaces
- Other pins
- BT coexistence

- Chips fall into two major categories: full- and soft-MAC
- Primary difference is where upper-level logic resides
  - Full: in chip’s firmware
  - Soft-MAC: in Linux’s mac80211
- Not better/worse, just different tradeoffs
WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

- Most chips must load firmware
- Even soft-MAC chips have firmware
- Blobs from vendors, typically in /lib/firmware
- Linux-firmware git
- Load as late as possible to avoid delays or boot issues
• Full-MAC vs Soft-MAC
• Firmware
• Interfaces
• Other pins
• BT coexistence

• Virtually every hardware bus interface is represented
  – SDIO
  – USB
  – PCIe
• Most drivers have multiple h/w interfaces
• h/w bus driver handles the bus; use the API but don’t handle it directly
• Usually a bus-neutral abstraction layer for messaging
• Full-MAC vs Soft-MAC
• Firmware
• Interfaces
• Other pins
• BT coexistence
Full-MAC vs Soft-MAC
Firmware
Interfaces
Other pins

BT coexistence

BT integrated in some chips, not in others
Pins for BT, often UART
Pins for coex
Often ignored; needs setup
• Chip and bus
• Bus driver
• WiFi device driver
• mac80211 (Soft-MAC)
• cfg80211
• nl80211
• User-space management
• Data
• Chip and bus
• Bus driver
• WiFi device driver
• mac80211 (Soft-MAC)
• cfg80211
• nl80211
• User-space management
• Data

• Two types: Full and Soft-MAC
• Bus interfaces
  – SDIO
  – USB
  – PCIe
Linux wireless stack (from bottom up)

- Chip and bus
- **Bus driver**
- WiFi device driver
- `mac80211` (Soft-MAC)
- `cfg80211`
- `nl80211`
- User-space management
- Data

- Depends on the bus
- Uses standard Linux bus driver
- Types
  - SDIO
  - USB
  - PCIe
  - others
• Chip and bus
• Bus driver
• **WiFi device driver**
• mac80211 (Soft-MAC)
• cfg80211
• nl80211
• User-space management
• Data

• Two types, Full and Soft MAC
• Soft uses mac80211
• Uses the bus driver to talk to the hardware
• Usually a bus-independent abstraction layer to support multiple buses (eg ath6kl_core + ath6kl_sdio || ath6kl_usb)
• Chip and bus
• Bus driver
• WiFi device driver
  • mac80211 (Soft-MAC)
  • cfg80211
  • nl80211
• User-space management
• Data

• Linux WiFi MAC driver
• Used by Soft-MAC devices
• Use:
  – ieee80211_ops
  – ieee80211_alloc_hw
  – ieee80211_register_hw
Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Main configuration API used
- All drivers (Full and SoftMAC) use it
- Replaced wext
• Chip and bus
• Bus driver
• WiFi device driver
• mac80211 (Soft-MAC)
• cfg80211
• nl80211
• User-space management
• Data

• Userspace interface to cfg80211
• Along with cfg80211 replaces wext
• No more ioctls
Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Basic tools like `ip`, `iw` and so on
- wpa_supplicant
- Higher level tools like NetworkManager, conman, etc
Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- From the data-application side, no different than Ethernet or other systems
- Open UDP or TCP sockets
Interfacing
• So you’ve got a new device, what do you do?
  – Plug it in, get the device IDs from the relevant bus tools
  – Determine what driver it matches
    • hotplug loads it or modprobe if necessary
  – If none, maybe it’s similar to another chip?
    • Add the bus device IDs to the existing driver
    • Write from scratch?!? (call me!)
  – Get the basics working first, then move on to the other features
```plaintext
Device IDs

• `lspci -nn` : “01:00.0 Network controller [0280]: Qualcomm Atheros AR93xx Wireless Network Adapter [168c:0030] (rev 01) ”
• sdio, use sysfs:
  # cd /sys/bus/sdio/devices/mmc0:0001:1
  # cat vendor
  0x0271
  # cat device
  0x0301
```
• Find and add to code as necessary
  Example: ath6kl, AR6003 0x0271:0x301

```c
#define MANUFACTURER_ID_AR6003_BASE 0x300
#define MANUFACTURER_ID_AR6004_BASE 0x400

/* SDIO manufacturer ID and Codes */
#define MANUFACTURER_ID_AH6KL_BASE_MASK 0xFFF00
#define MANUFACTURER_CODE 0x271 /* Atheros */
```

drivers/net/wireless/ath/ath6kl/hif.h

```c
static const struct sdio_device_id ath6kl_sdio_devices[] = {
    {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6003_BASE | 0x8))},
    {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6003_BASE | 0x1))},
    {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x8))},
    {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x1))},
    {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x2))},
};
```

drivers/net/wireless/ath/ath6kl/sdio.c
Good result

```
[ 15.240000] mmc0: queuing unknown CIS tuple 0x01 (3 bytes)
[ 15.250000] mmc0: queuing unknown CIS tuple 0x1a (5 bytes)
[ 15.250000] mmc0: queuing unknown CIS tuple 0x1b (8 bytes)
[ 15.260000] mmc0: queuing unknown CIS tuple 0x14 (0 bytes)
[ 15.260000] mmc0: queuing unknown CIS tuple 0x80 (1 bytes)
[ 15.260000] mmc0: queuing unknown CIS tuple 0x81 (1 bytes)
[ 15.260000] mmc0: queuing unknown CIS tuple 0x82 (1 bytes)
[ 15.260000] mmc0: new high speed SDIO card at address 0001
[ 15.320000] ath6kl_sdio mmc0:0001:1: Direct firmware load for ath6k/AR6003/hw2.1.1/fw-5.bin failed with error -2
[ 15.360000] ath6kl: initializing atheros generic netlink family
[ 15.580000] random: nonblocking pool is initialized
[ 16.130000] ath6kl: ar6003 hw 2.1.1 sdio fw 3.4.0.0094 api 4
```
• Is it plugged in correctly?
• Is it powered (sometimes a power enable pin)?
• Does it announce on bus?
• Does it fail to load firmware?
• Does it come up and look OK, but fails on `ifconfig wlan0 up`?
• Does it work OK, but won’t connect to an open AP?
• Does it connect to open AP but other features won’t work?
• Kill NetworkManager (or equivalent)
A hunch is creativity trying to tell you something
- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark
• *kconfig debug options*
• *dmesg*
• *module parameters*
• *sysfs*
• *debugfs*
• *coredump*
• *fttrace*
• *Wireshark*

• By default, debug config options are disabled
• *brcmfmac*: `CONFIG_BRCMDBG`
• *ath6kl*: `ATH6KL_DEBUG`, `ATH6KL_TRACING`
Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Lots of debug printing to kernel log.
- Need to ++ debug print level and enable specific messages.
- ath6kl:
  ```
  `echo 0x00140000 > /sys/module/ath6kl_core/parameters/debug_mask`
  ```
• kconfig debug options
• dmesg
• module parameters
• sysfs
• debugfs
• coredump
• ftrace
• Wireshark

• Different drivers have special module parameters. brcmfmac:
  – debug - Sets debug level. The levels are defined in debug.h. Maintainer asks for kernel logs with debug level set to 0x1416, which shows driver-firmware interactions.
  – feature_disable - Override feature detection to avoid its use in driver.
  – ignore_probe_fail - Allow post-mortem debugging if firmware crash happens during probe. Check the forensics file in debugfs.
Debugging tools

- kconfig debug options
- dmesg
- module parameters
  - sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Best place to check for existence, can walk the bus tree
- Shows you bound driver, etc
- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- brcmfmac:
  - revinfo - revision of h/w and firmware
  - features - firmware features
  - msgbuf_stats (pcie only) - stats counters of msgbuf layer
  - counters (sdio) - stats counter of sdio bus layer
  - console_interval (sdio) - period to obtain fw console content
  - forensics (sdio) - dump fw console and trap info
Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Many drivers will dump firmware crashes to sysfs “file”
- brcmfmac: /sys/class/devcoredump
Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
  - ftrace
  - Wireshark
- Kernel’s function tracer
- Must enable
- Useful to dig through call-stacks and figure out what’s going on live
- Also can profile the stack
trace-cmd

```c
trace-cmd
```

```
07:35:13.014 [    0]  0.000 us  usb_kcd_init()
10.59:10:00.000 [    0]  0.000 us  usbd_kcd_init()
26.76:10:00.000 [    0]  0.000 us  usbd_kcd_init()
26.76:10:00.000 [    0]  0.000 us  usbd_kcd_init()
26.76:10:00.000 [    0]  0.000 us  usbd_kcd_init()
... (truncated output)
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
### ftrace

```
<table>
<thead>
<tr>
<th>#</th>
<th>CPU</th>
<th>Time Stamp</th>
<th>Task</th>
<th>PID</th>
<th>Latency</th>
<th>Event</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>10272</td>
<td>0</td>
<td>8305.792540</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.008 us</td>
<td>func_graph_exit</td>
<td>0.028 us</td>
</tr>
<tr>
<td>10273</td>
<td>1</td>
<td>8305.792540</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.012 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10274</td>
<td>1</td>
<td>8305.792543</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.008 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10275</td>
<td>1</td>
<td>8305.792544</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.028 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10276</td>
<td>1</td>
<td>8305.792544</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.028 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10277</td>
<td>1</td>
<td>8305.792545</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.074 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10278</td>
<td>1</td>
<td>8305.792545</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.074 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10279</td>
<td>1</td>
<td>8305.792546</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.053 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10280</td>
<td>1</td>
<td>8305.792546</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.053 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10281</td>
<td>1</td>
<td>8305.792546</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.372 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10282</td>
<td>1</td>
<td>8305.792546</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.372 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10283</td>
<td>1</td>
<td>8305.792547</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.125 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10284</td>
<td>1</td>
<td>8305.792547</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.125 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10285</td>
<td>1</td>
<td>8305.792547</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.055 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10286</td>
<td>1</td>
<td>8305.792547</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.055 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10287</td>
<td>1</td>
<td>8305.792547</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.096 us</td>
<td>func_graph_entry</td>
<td>0.212 us</td>
</tr>
<tr>
<td>10288</td>
<td>1</td>
<td>8305.792548</td>
<td>aireplay-ng</td>
<td>3837</td>
<td>.096 us</td>
<td>func_graph_exit</td>
<td>0.212 us</td>
</tr>
</tbody>
</table>
```
• kconfig debug options
• dmesg
• module parameters
• sysfs
• debugfs
• coredump
• ftrace
• Wireshark

• Device is working, but actual communication fails
• Do the packets go on the air?
• Is there something wrong with what’s going on?
Wireshark
• **linux-wireless list** - need more than “it doesn’t work”:
  – enable debugging features
  – send kernel logs with those debugging on
  – be specific about hardware, device, firmware version, Linux version etc.
  – be specific about what doesn’t work and what you’ve tried
  – expect to include wireless captures
• Linux-wireless wiki: https://wireless.wiki.kernel.org/
• Linux-firmware: https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/
• Linux wireless mailing list: https://wireless.wiki.kernel.org/en/developers/mailinglists
• Help with iwlwifi: https://wireless.wiki.kernel.org/en/users/drivers/iwlwifi/debugging
CAL-SIERRA
CONSULTING

STEVE DEROISIER
steve@cal-sierra.com
+1-916-203-5695