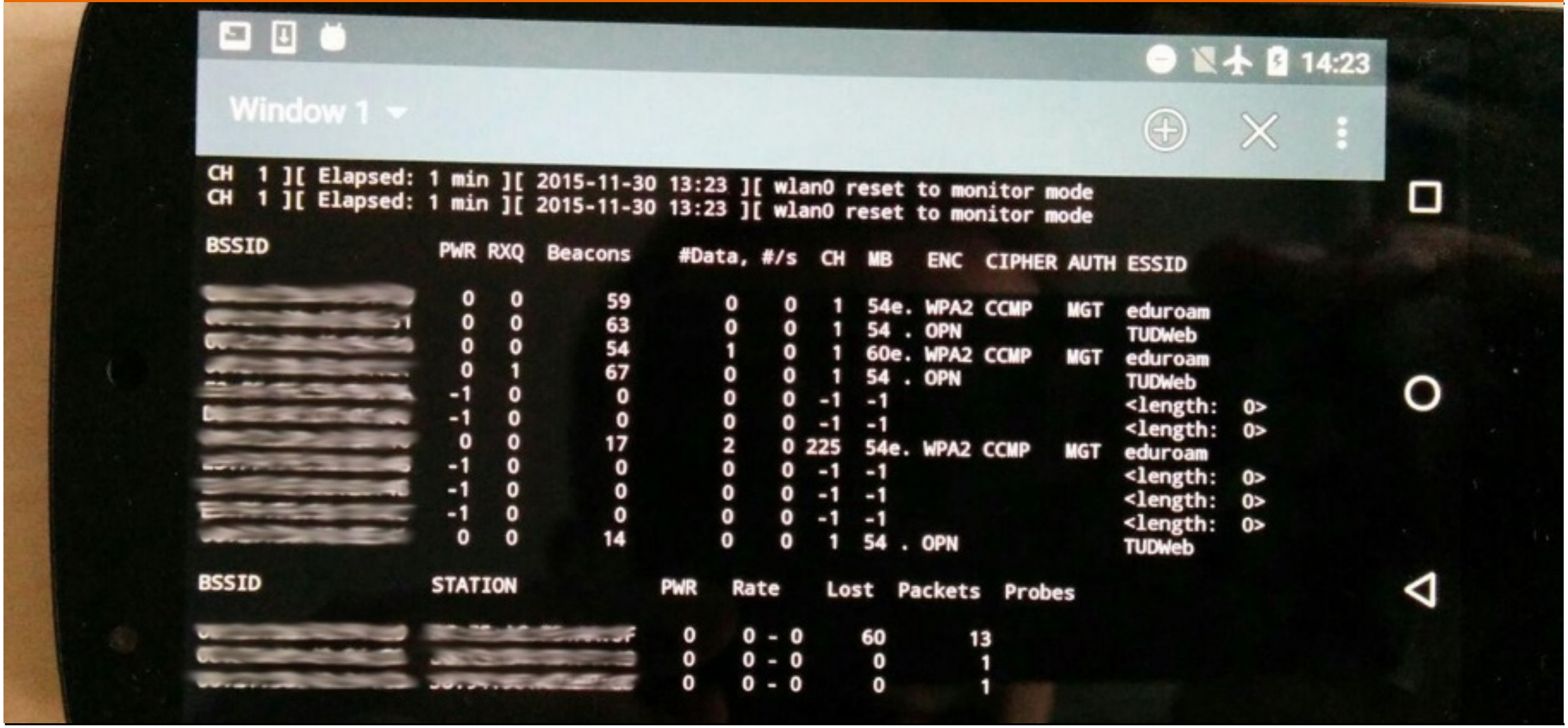
NEXMON

An Open Source Firmware for Broadcom FullMAC WiFi chips

Matthias Schulz

TECHNISCHE UNIVERSITÄT DARMSTADT

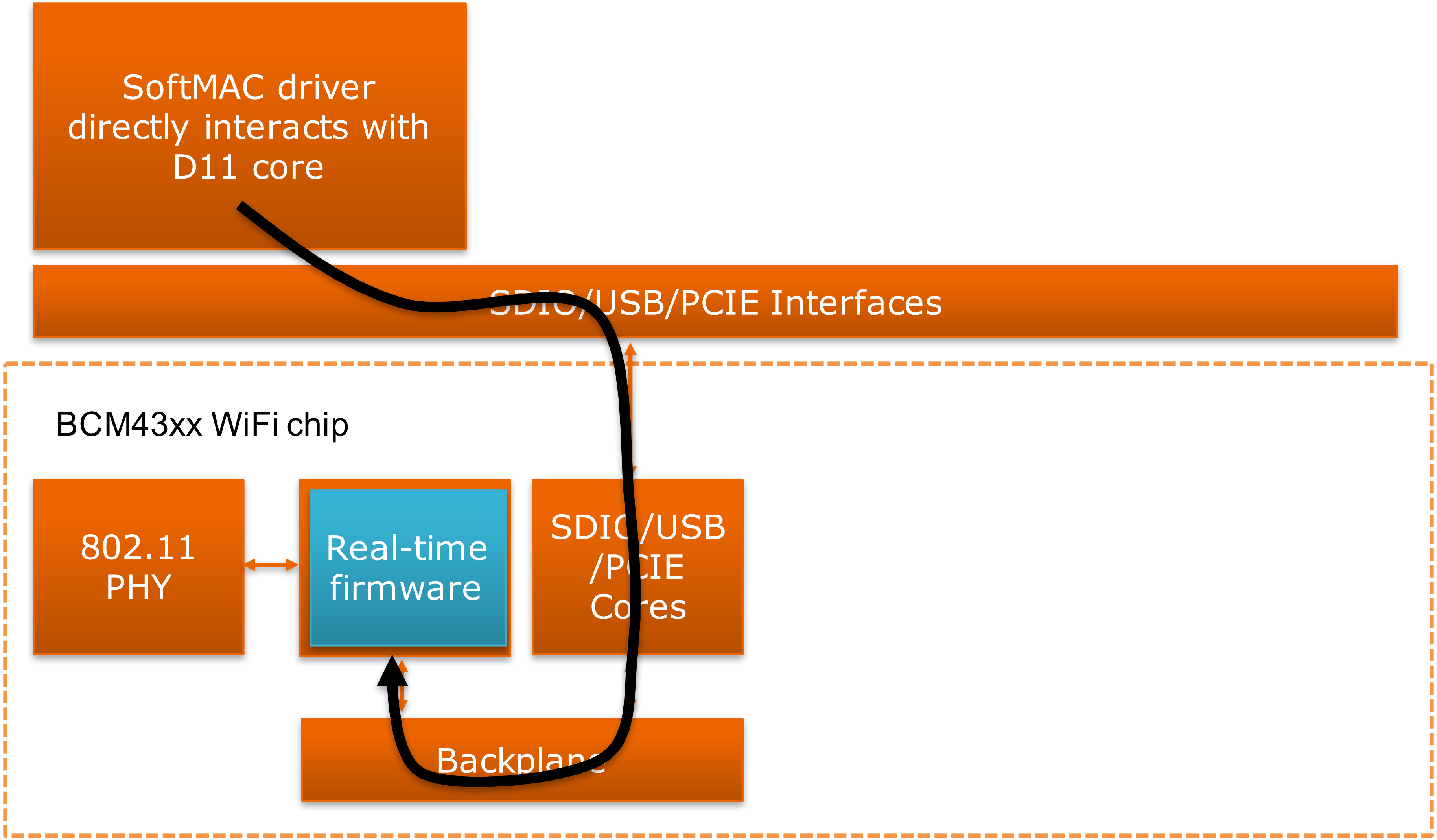


```
CH  1 ][ Elapsed: 1 min ][ 2015-11-30 13:23 ][ wlan0 reset to monitor mode
CH  1 ][ Elapsed: 1 min ][ 2015-11-30 13:23 ][ wlan0 reset to monitor mode

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

[redacted]           0   0       59        0    0   1  54e. WPA2 CCMP   MGT  eduroam
[redacted]           0   0       63        0    0   1  54 . OPN              TUDWeb
[redacted]           0   0       54        1    0   1  60e. WPA2 CCMP   MGT  eduroam
[redacted]           0   1       67        0    0   1  54 . OPN              TUDWeb
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]           0   0       17        2    0 225  54e. WPA2 CCMP   MGT  eduroam
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]           0   0       14        0    0   1  54 . OPN              TUDWeb

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

[redacted]         [redacted]           0    0 - 0    60       13
[redacted]         [redacted]           0    0 - 0     0        1
[redacted]         [redacted]           0    0 - 0     0        1
```

Our Goal

	mac80211 SoftMAC Implementation e.g. brcmsmac	"Ethernet" FullMAC Implementation e.g. BCMDHD	Open MAC Implementation e.g. NEXMON
Time critical MAC parts (timings, acknowledgments)	Closed source **real-time firmware**	Closed source **real-time firmware**	Open source **real-time firmware** (future work)
Management Tasks (association, encryption)	Open source **driver**	Closed source **firmware**, **driver** only sees Ethernet frames	Partially open source **firmware/driver**
Extendibility/ Hackability	Access to WiFi frames, non-real-time MAC layer modifications	No frame injection, monitor mode relies on firmware implementation	Hacker decides where MAC layer parts are implemented

Structure of Broadcom WiFi Chips



SoftMAC driver directly interacts with D11 core

SDIO/USB/PCIE Interfaces

BCM43xx WiFi chip

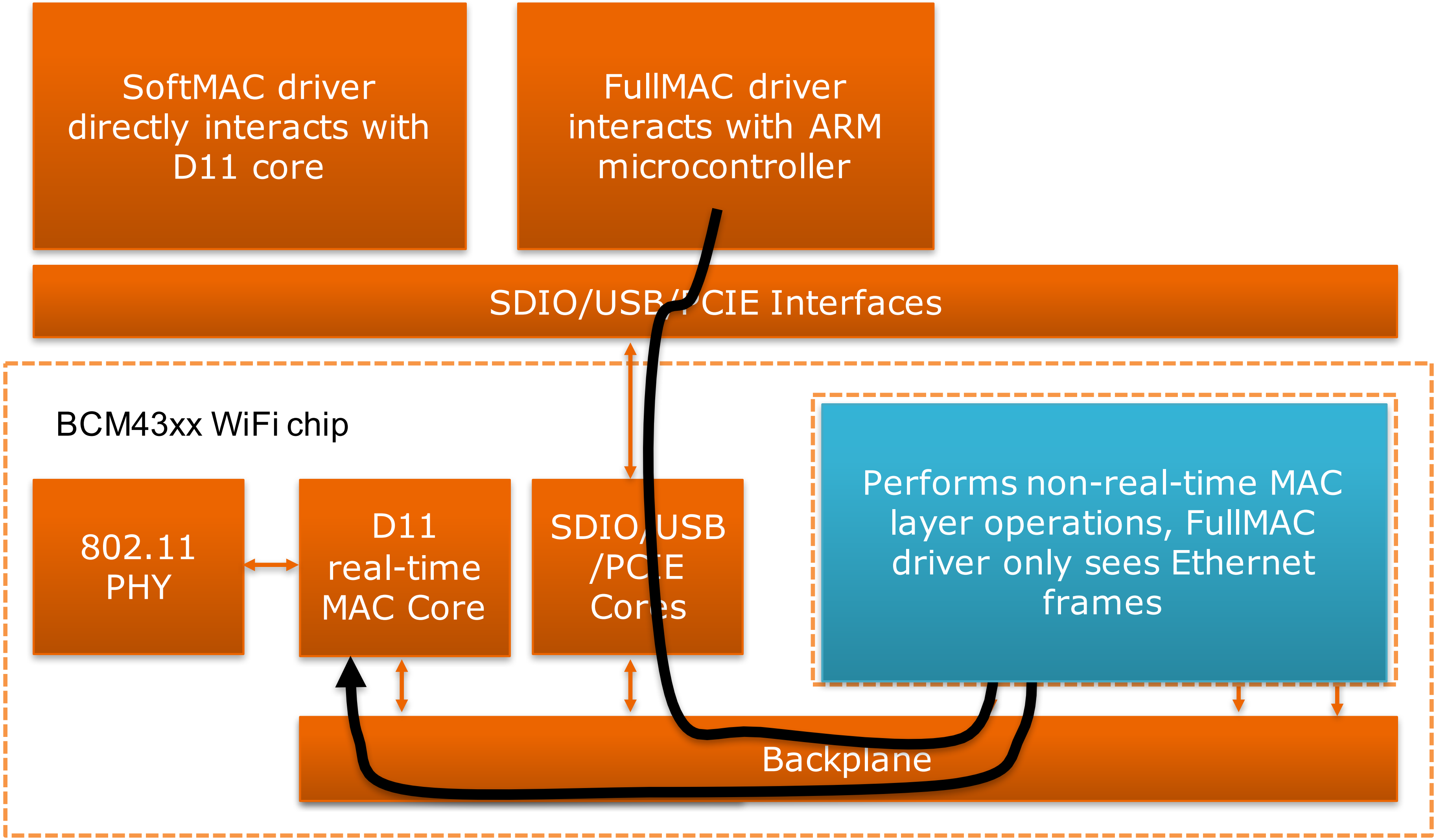
802.11 PHY

Real-time firmware

SDIO/USB /PCIE Cores

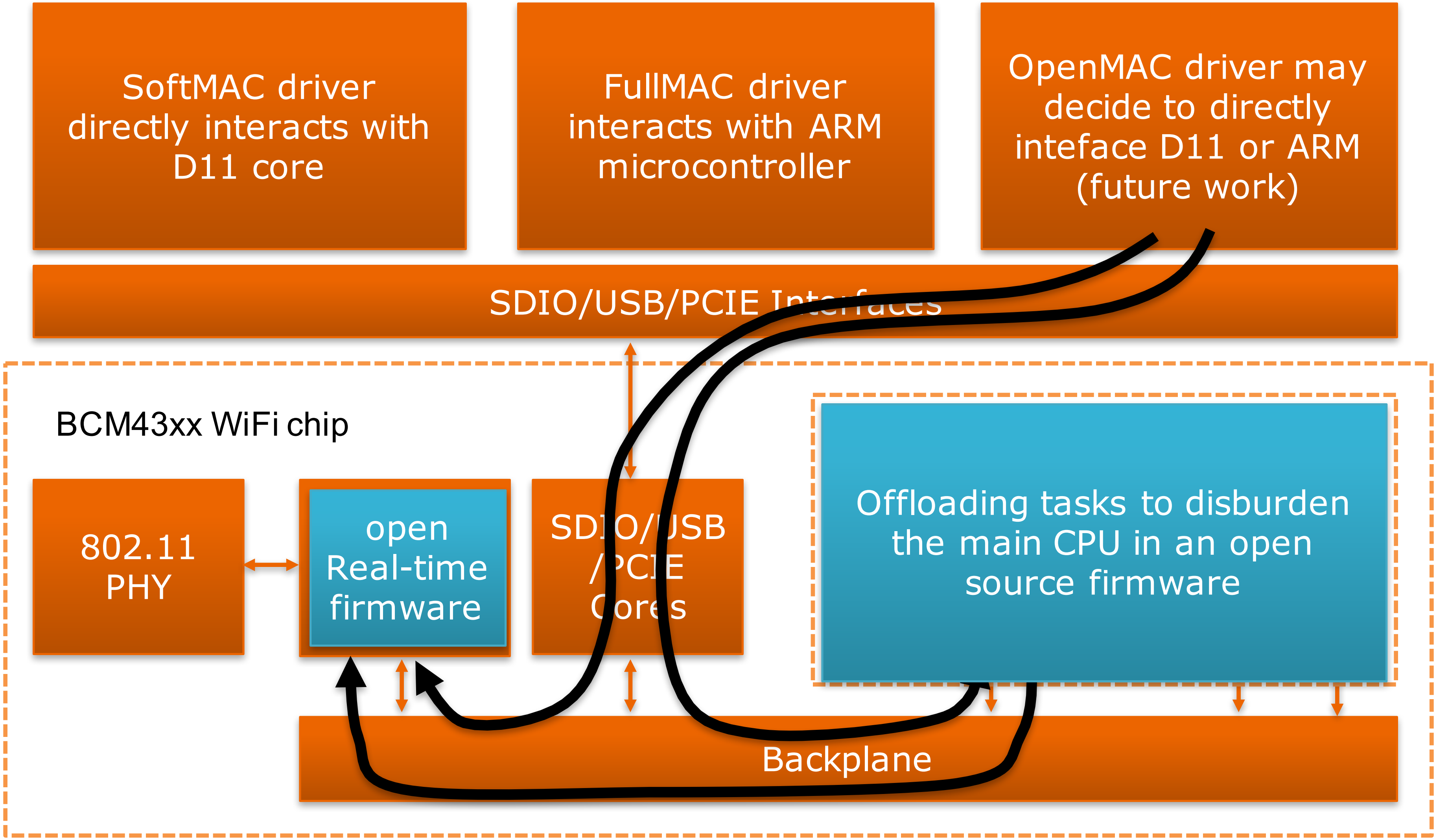
Backplane

Structure of Broadcom WiFi Chips



SoftMAC driver directly interacts with D11 core
FullMAC driver interacts with ARM microcontroller
SDIO/USB/PCIE Interfaces
BCM43xx WiFi chip
802.11 PHY
D11 real-time MAC Core
SDIO/USB /PCIE Cores
Performs non-real-time MAC layer operations, FullMAC driver only sees Ethernet frames
Backplane

Structure of Broadcom WiFi Chips



SoftMAC driver directly interacts with D11 core

FullMAC driver interacts with ARM microcontroller

OpenMAC driver may decide to directly inteface D11 or ARM (future work)

SDIO/USB/PCIE Interfaces

BCM43xx WiFi chip

802.11 PHY

open Real-time firmware

SDIO/USB /PCIE Cores

Offloading tasks to disburden the main CPU in an open source firmware

Backplane

Current Development Status

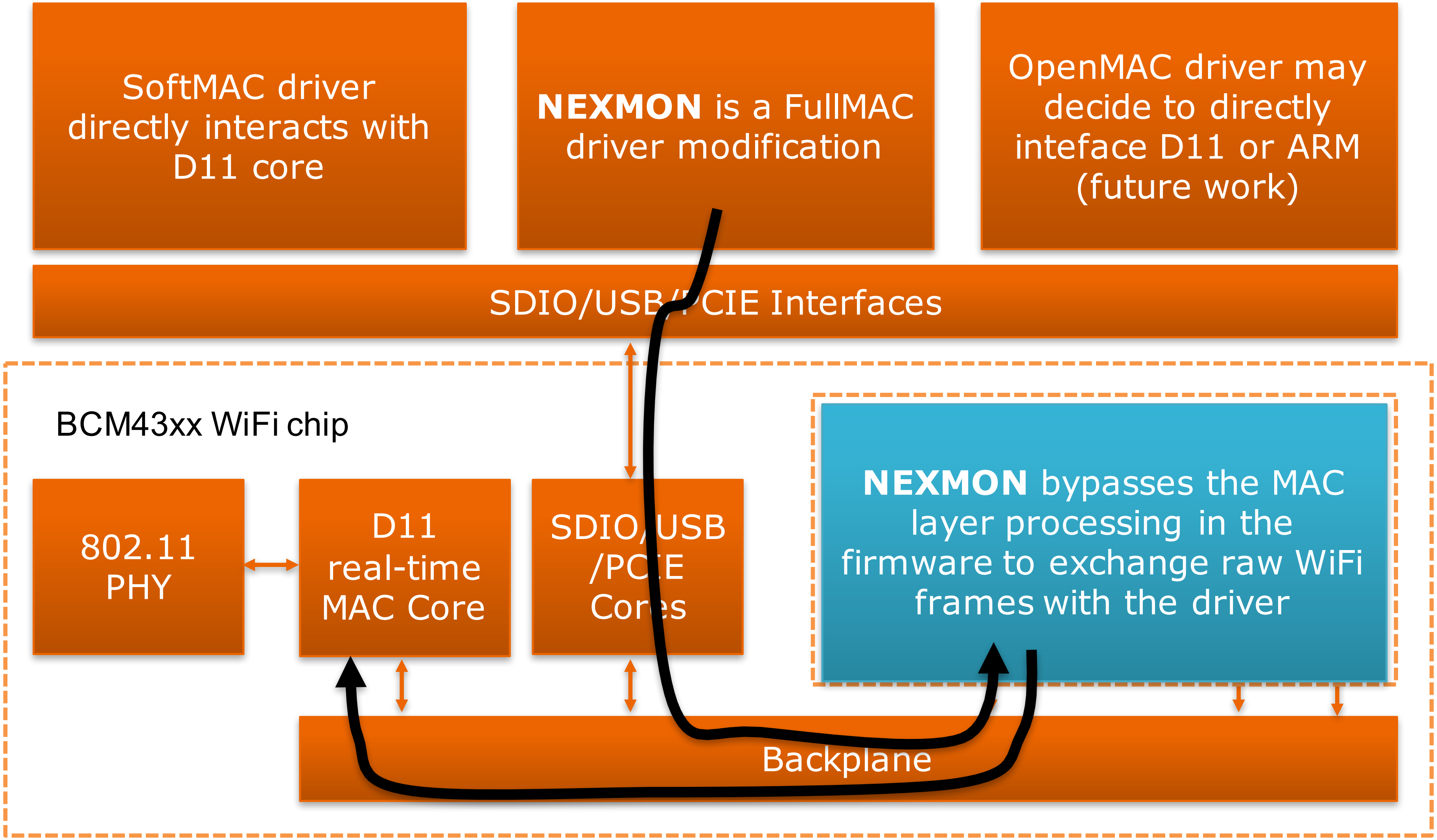
The

nexmon

project

Bringing Monitor Mode and
Frame Injection to FullMAC Chips

Structure of Broadcom WiFi Chips



SoftMAC driver directly interacts with D11 core

NEXMON is a FullMAC driver modification

OpenMAC driver may decide to directly inteface D11 or ARM (future work)

SDIO/USB/PCIE Interfaces

BCM43xx WiFi chip

802.11 PHY

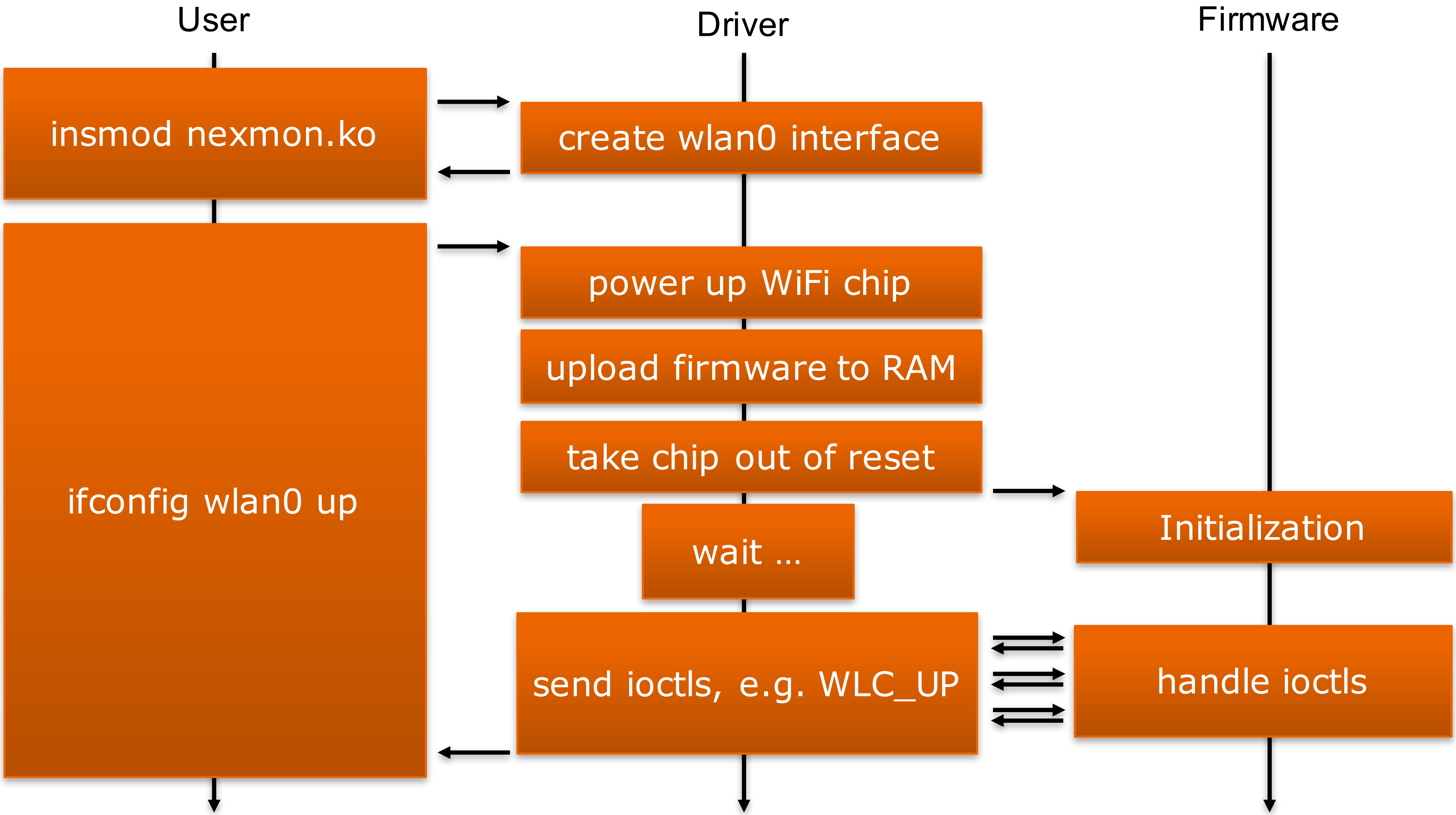
D11 real-time MAC Core

SDIO/USB /PCIE Cores

NEXMON bypasses the MAC layer processing in the firmware to exchange raw WiFi frames with the driver

Backplane

How the Chip Starts Up


User

Driver

Firmware

insmod nexmon.ko

create wlan0 interface

ifconfig wlan0 up

power up WiFi chip

upload firmware to RAM

take chip out of reset

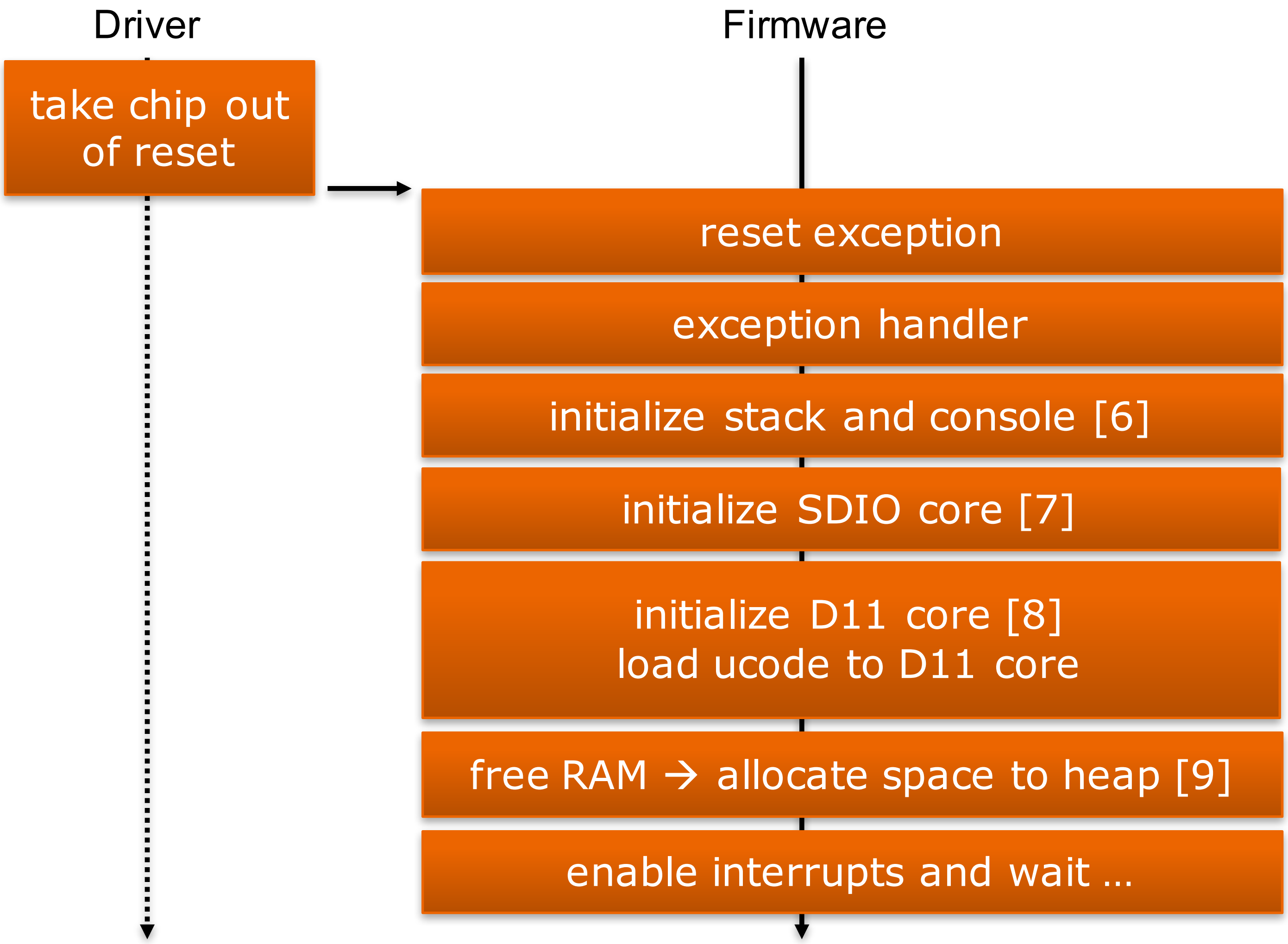
Initialization

wait …

send ioctls, e.g. WLC_UP

handle ioctls

Initialization of the Chip



Driver

Firmware

take chip out of reset

reset exception

exception handler

initialize stack and console [6]

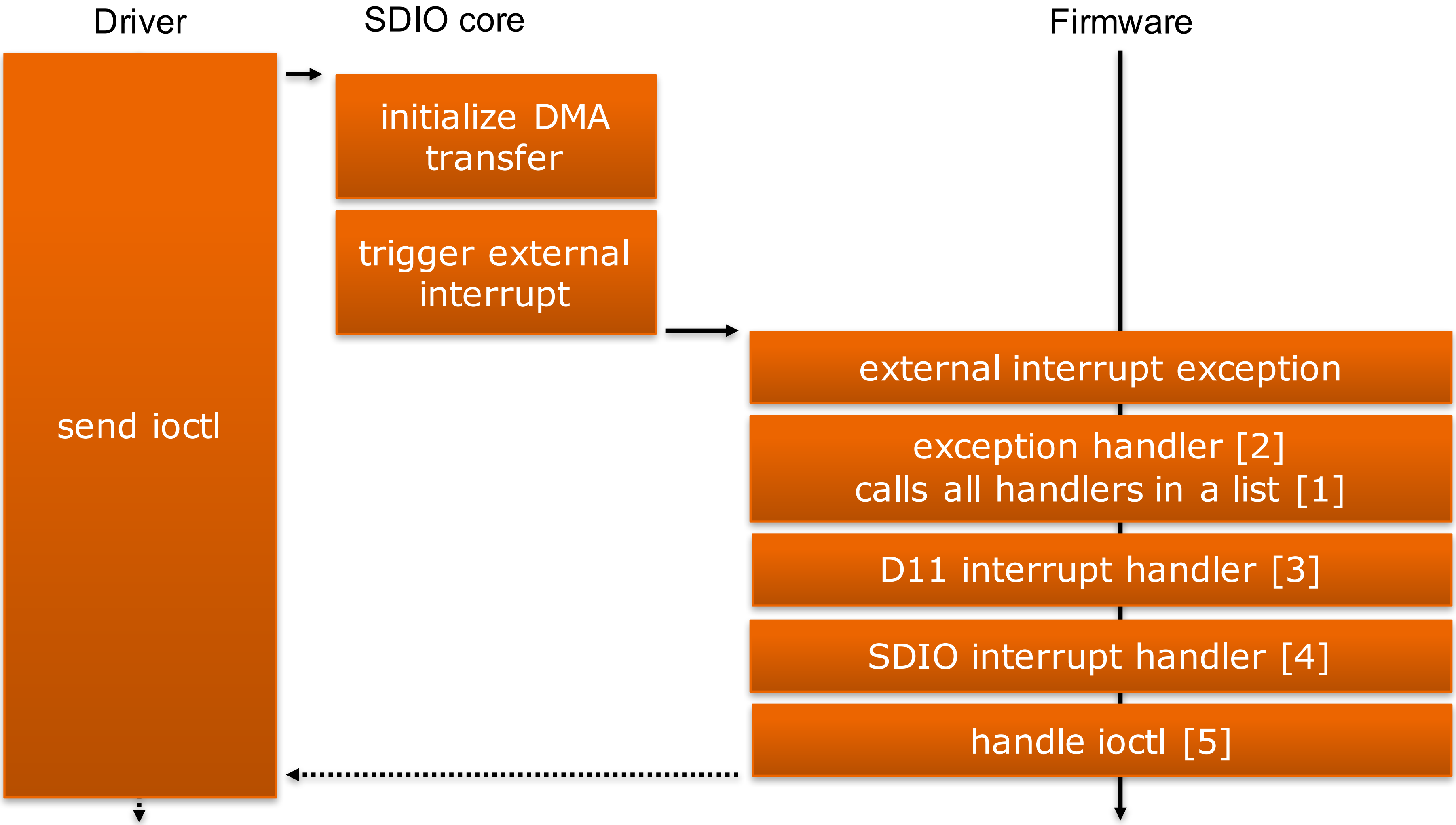
initialize SDIO core [7]

initialize D11 core [8]
load ucode to D11 core

free RAM → allocate space to heap [9]

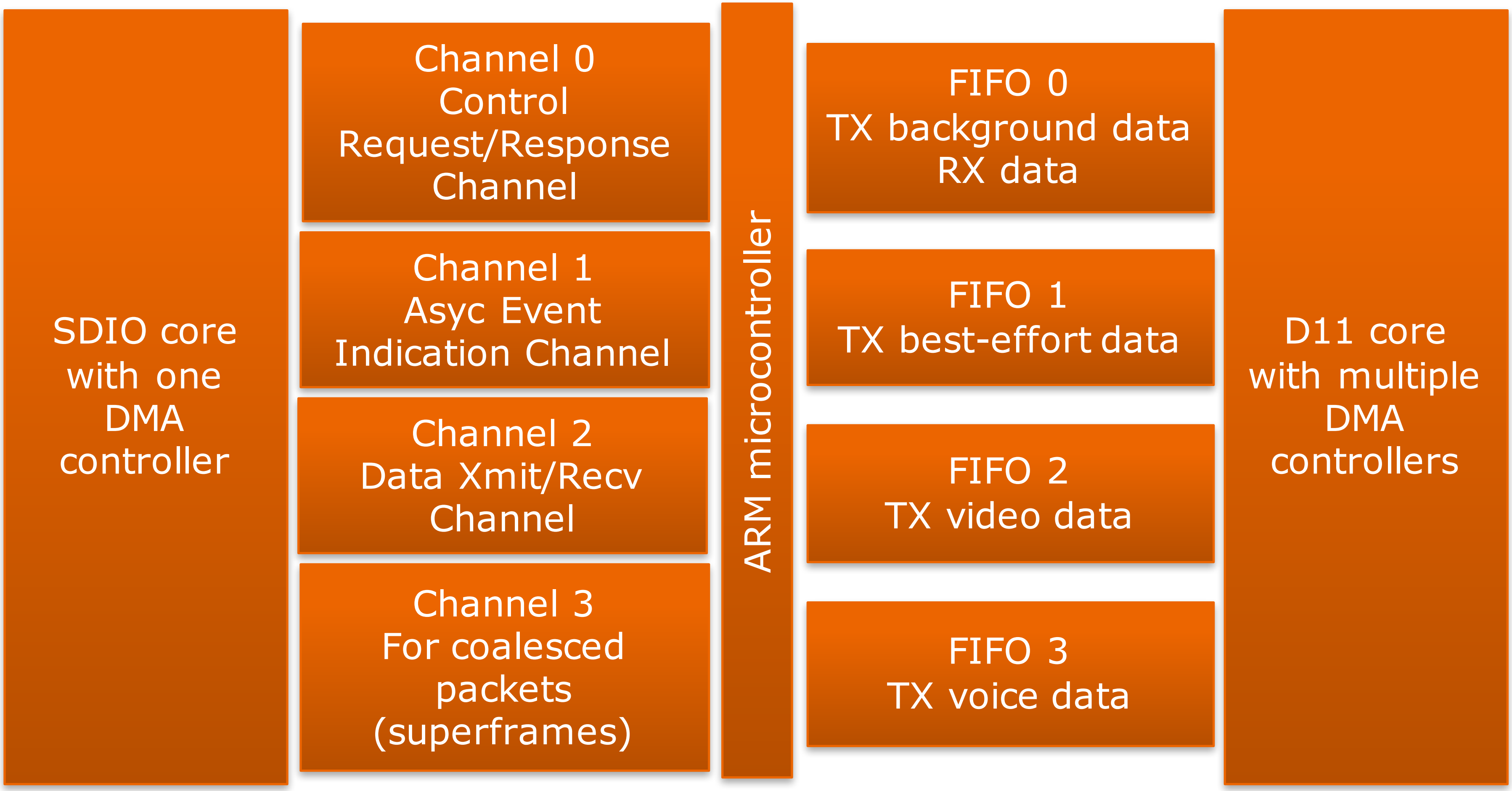
enable interrupts and wait …

How Interrupt Handling Works



Driver
SDIO core
Firmware
send ioctl
initialize DMA transfer
trigger external interrupt
external interrupt exception
exception handler [2]
calls all handlers in a list [1]
D11 interrupt handler [3]
SDIO interrupt handler [4]
handle ioctl [5]

Data Transfers between Cores



SDIO core with one DMA controller

Channel 0
Control Request/Response Channel

Channel 1
Asyc Event Indication Channel

Channel 2
Data Xmit/Recv Channel

Channel 3
For coalesced packets (superframes)

ARM microcontroller

FIFO 0
TX background data
RX data

FIFO 1
TX best-effort data

FIFO 2
TX video data

FIFO 3
TX voice data

D11 core with multiple DMA controllers

BCMON/MONMOB based Monitor Mode Patch for NEXMON

D11 interrupt handler → wlc_bmac_recv

```
int wlc_bmac_recv(struct wlc_hw_info *wlc_hw, unsigned int fifo, int bound, int *processed_frm_cnt)
  {
    struct wlc_pub *pub = wlc_hw->wlc->pub;
    sk_buff *p;
    char is_amsdu = pub->is_amsdu;
    int n = 0, bound_limit;
    if(bound) bound_limit = pub->tunables->rxbnd;
    else bound_limit = -1;
    do {
        p = dma_rx (wlc_hw->di[fifo]);
        if(!p) goto LEAVE;
        if(is_amsdu) is_amsdu = 0;
        dngl_sendpkt(SDIO_INFO_ADDR, p, NEXMON_MONITOR_CHANNEL);
        ++n;
    } while(n < bound_limit);
LEAVE:
    dma_rxfill(wlc_hw->di[fifo]);
    wlc_bmac_mctrl(wlc_hw, (MCTL_PROMISC | MCTL_KEEPCONTROL | MCTL_BCNS_PROMISC),
      (MCTL_PROMISC | MCTL_KEEPCONTROL | MCTL_BCNS_PROMISC));
    *processed_frm_cnt += n;
    return !(n < bound_limit);
}
```

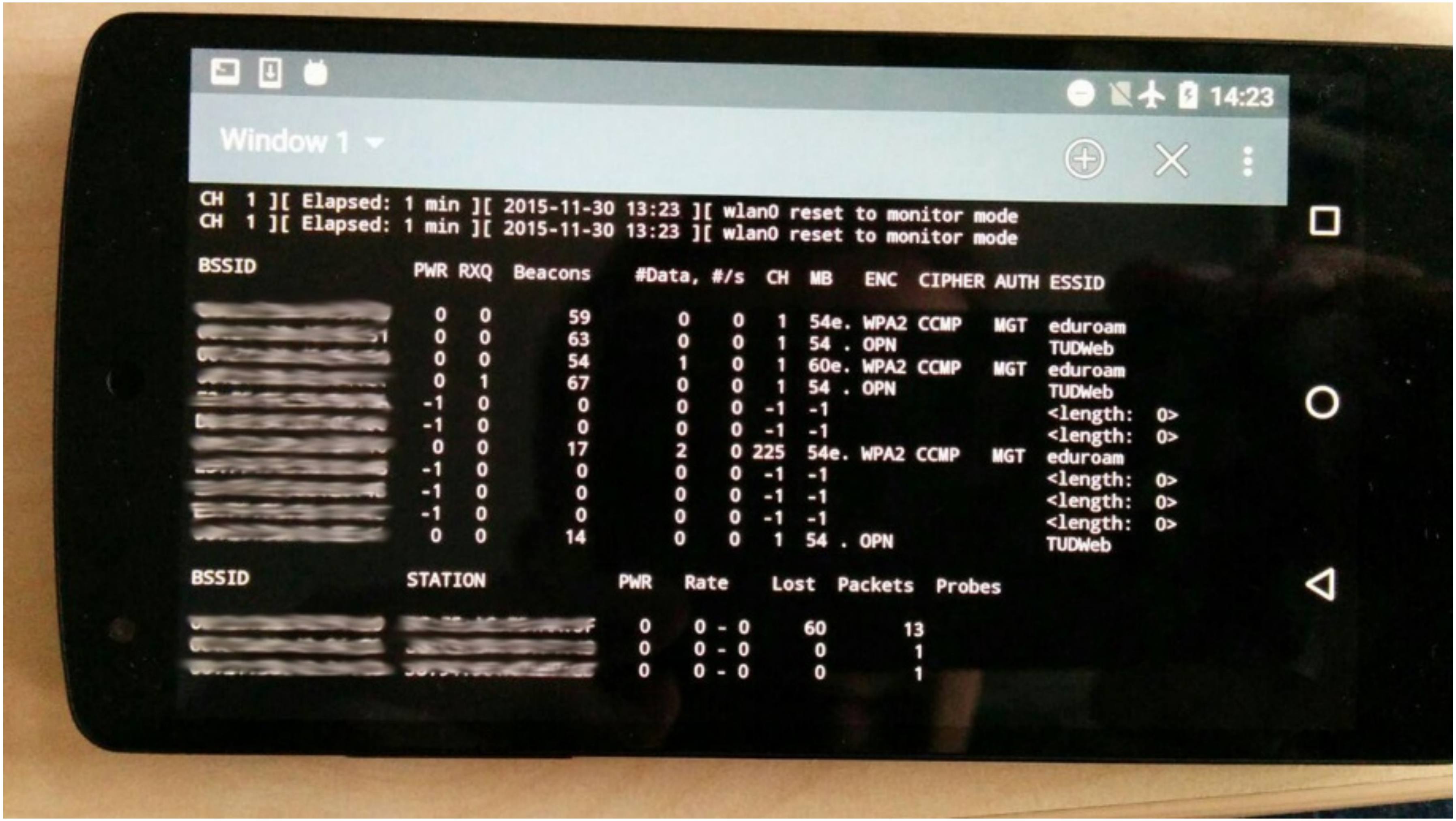
→ Get sk_buff from D11 FIFO

→ Send sk_buff to driver using SDIO

→ Limit number of frames to process

→ Always reactivate promiscuous mode

NEXMON in Action



```
CH  1 ][ Elapsed: 1 min ][ 2015-11-30 13:23 ][ wlan0 reset to monitor mode
CH  1 ][ Elapsed: 1 min ][ 2015-11-30 13:23 ][ wlan0 reset to monitor mode

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

[redacted]           0   0       59        0    0   1  54e. WPA2 CCMP   MGT  eduroam
[redacted]           0   0       63        0    0   1  54 . OPN              TUDWeb
[redacted]           0   0       54        1    0   1  60e. WPA2 CCMP   MGT  eduroam
[redacted]           0   1       67        0    0   1  54 . OPN              TUDWeb
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]           0   0       17        2    0 225  54e. WPA2 CCMP   MGT  eduroam
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]          -1   0        0        0    0  -1  -1                    <length:  0>
[redacted]           0   0       14        0    0   1  54 . OPN              TUDWeb

BSSID              STATION            PWR   Rate    Lost    Packets  Probes

[redacted]         [redacted]           0    0 - 0     60       13
[redacted]         [redacted]           0    0 - 0      0        1
[redacted]         [redacted]           0    0 - 0      0        1
```

C-based Programming Framework

wrapper.h
contains function prototypes and addresses of existing firmware functions

Makefile generates wrapper.c + wrapper.ld

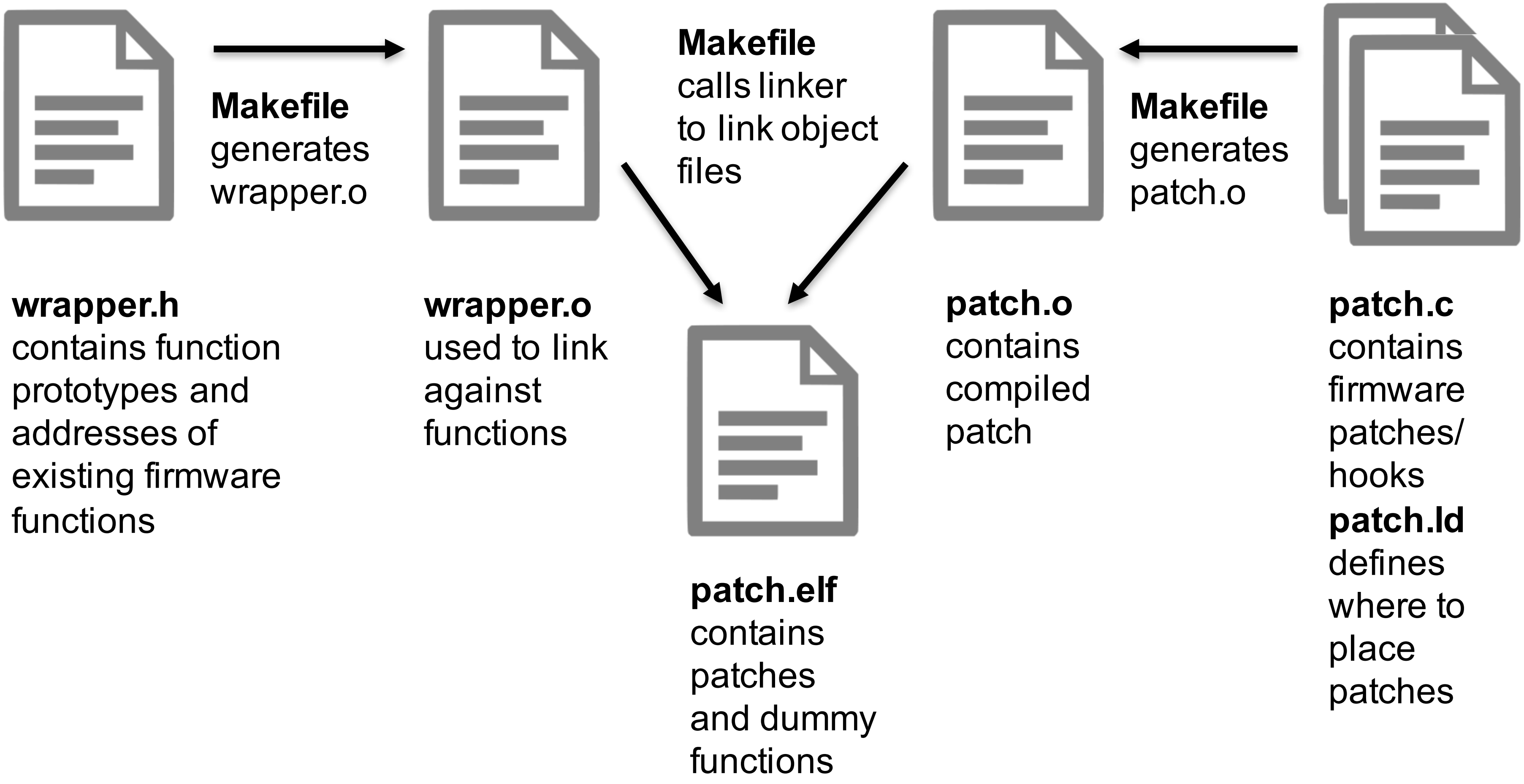
wrapper.c
contains dummy functions

wrapper.ld
places dummy functions at corresponding firmware addresses

Makefile generates wrapper.o and adjusts alignment

wrapper.o
used to link against functions

C-based Programming Framework



wrapper.h contains function prototypes and addresses of existing firmware functions

Makefile generates wrapper.o

wrapper.o used to link against functions

Makefile calls linker to link object files

patch.elf contains patches and dummy functions

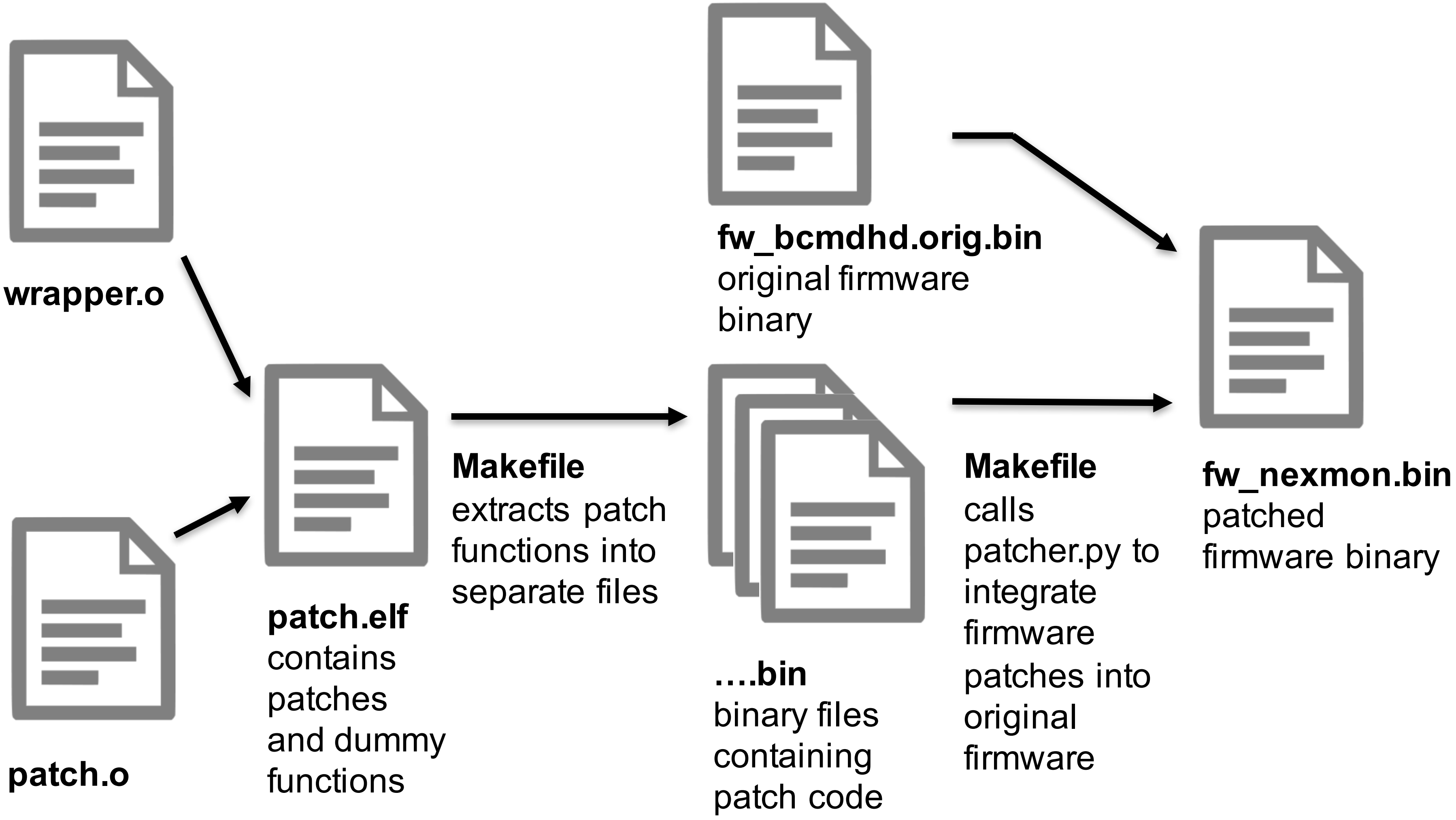
patch.o contains compiled patch

Makefile generates patch.o

patch.c contains firmware patches/ hooks

patch.ld defines where to place patches

C-based Programming Framework



wrapper.o

patch.o

patch.elf
contains
patches
and dummy
functions

Makefile
extracts patch
functions into
separate files

fw_bcmdhd.orig.bin
original firmware
binary

….bin
binary files
containing
patch code

Makefile
calls
patcher.py to
integrate
firmware
patches into
original
firmware

fw_nexmon.bin
patched
firmware binary

Debugging
Our latest Feature

Set hardware breakpoints and watchpoints

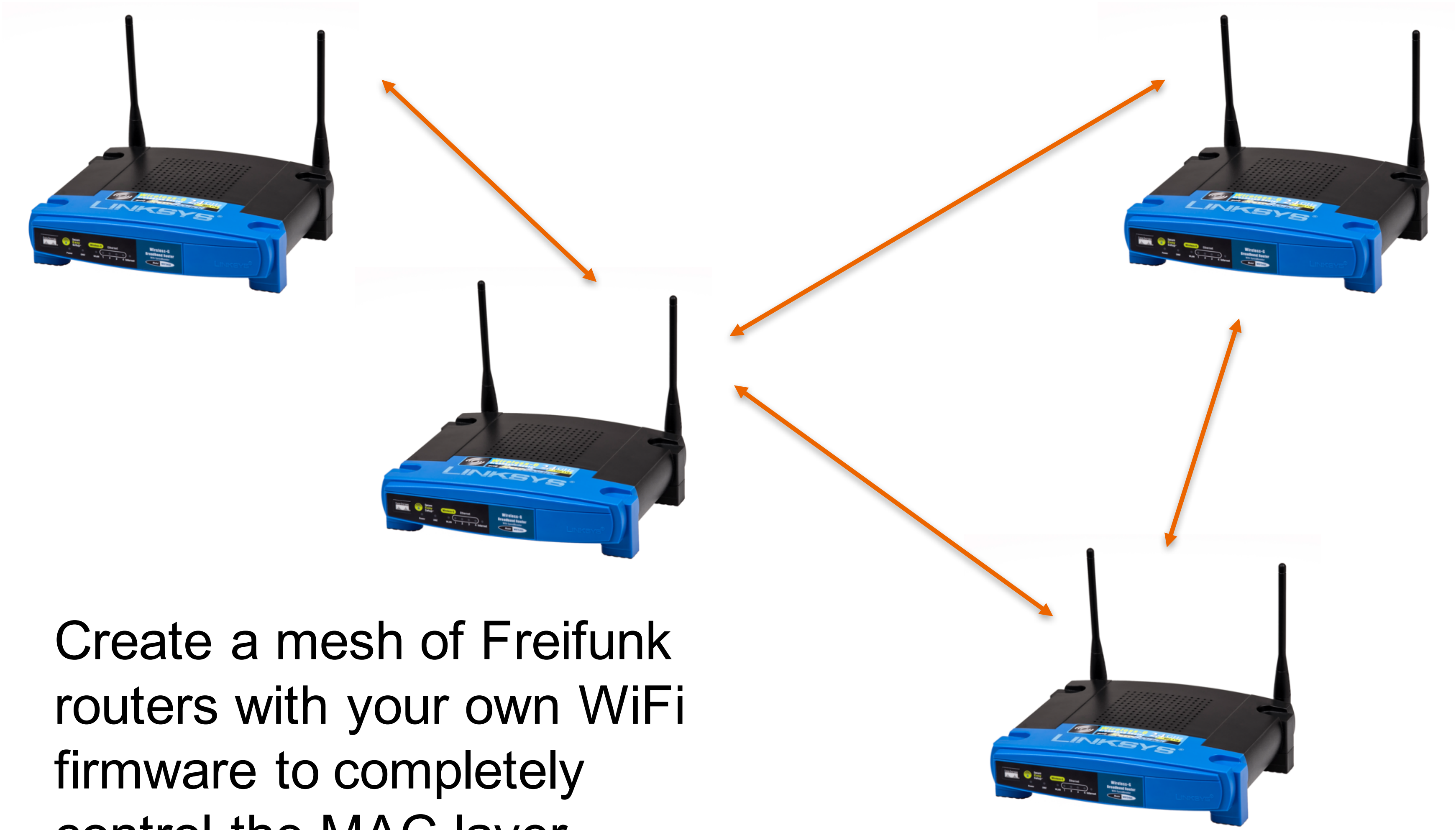
Watchpoint hits trigger data abort exception in monitor debugging mode

Breakpoint hits trigger prefetch abort exception in monitor debugging mode

We changed standard exception handlers to stay in abort mode to allow breakpoints on instruction mismatches, required to reset breakpoints after a hit

We created software debugger to handle debugging events

Possible Freifunk Projects



Create a mesh of Freifunk routers with your own WiFi firmware to completely control the MAC layer.

Interesting Addresses

[1] 0x180E5C pointer to external interrupt handlers

[2] 0x181A88 external interrupt handler

[3] 0x027550 D11 interrupt handler, calls wlc_dpc

[4] 0x01B944 SDIO interrupt handler, calls sdpcmd_dpc

[5] 0x19551C wlc_ioctl: ioctl handler

[6] 0x1EC1E4 initialization of stack and console

[7] 0x1ED6F4 call to SDIO device initialization code

[8] 0x1ED6F4 call to D11 device initialization code

[9] 0x1816E4 function that reclaims memory and allocate it to heap

SECURE MOBILE NETWORKING

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Matthias Schulz
Department of Computer Science

SEEMOO
Mornewegstr. 32
64293 Darmstadt/Germany
mschulz@seemoo.tu-darmstadt.de

Phone +49 6151 16-25478
Fax +49 6151 16-25471
www.seemoo.tu-darmstadt.de