AR6003 for Android Platform

Wilson Loi/ Date [20110324]

2011 FAE Training

There is Here.™

Agenda

- WIFI Driver on Android
  - Android Architecture about WLAN
  - The Differences for WIFI Driver between Android and Linux
  - How to Compile WIFI Driver for Android Environment
  - How to Verify WIFI Driver Manually
  - WIFI Driver Architecture
  - Android WIFI Flows

Agenda

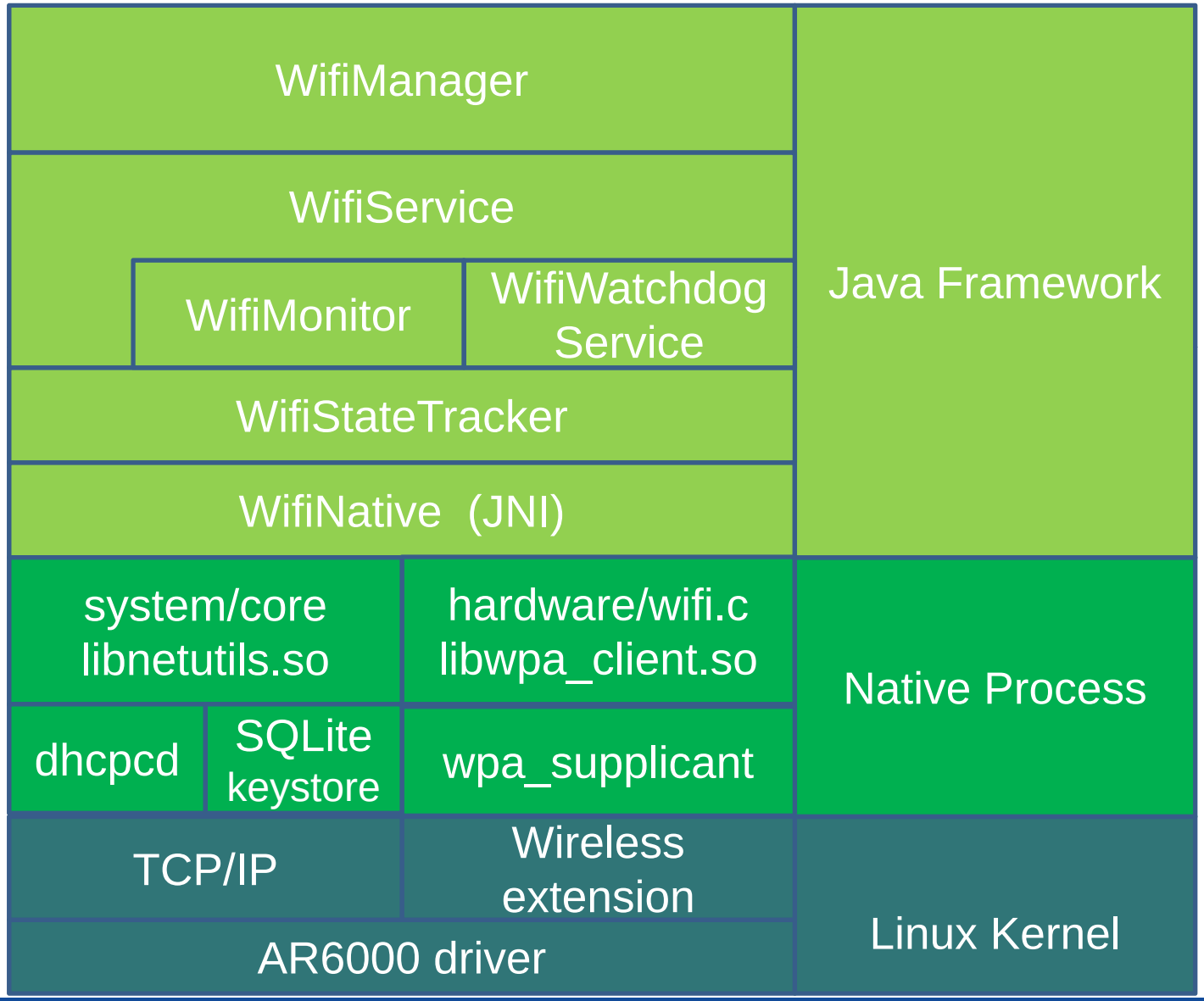
ATHEROS

- Linux MMC Stack
  - Working Model
  - Bus Driver: MMC, SD, SDIO
  - The Interface between Bus Driver and Host Controller Driver
  - The Interface Between Bus Driver and Function Driver
  - Bus Driver Card Polling Mechanism
  - SDIO Card Initialization
  - Interrupt Process

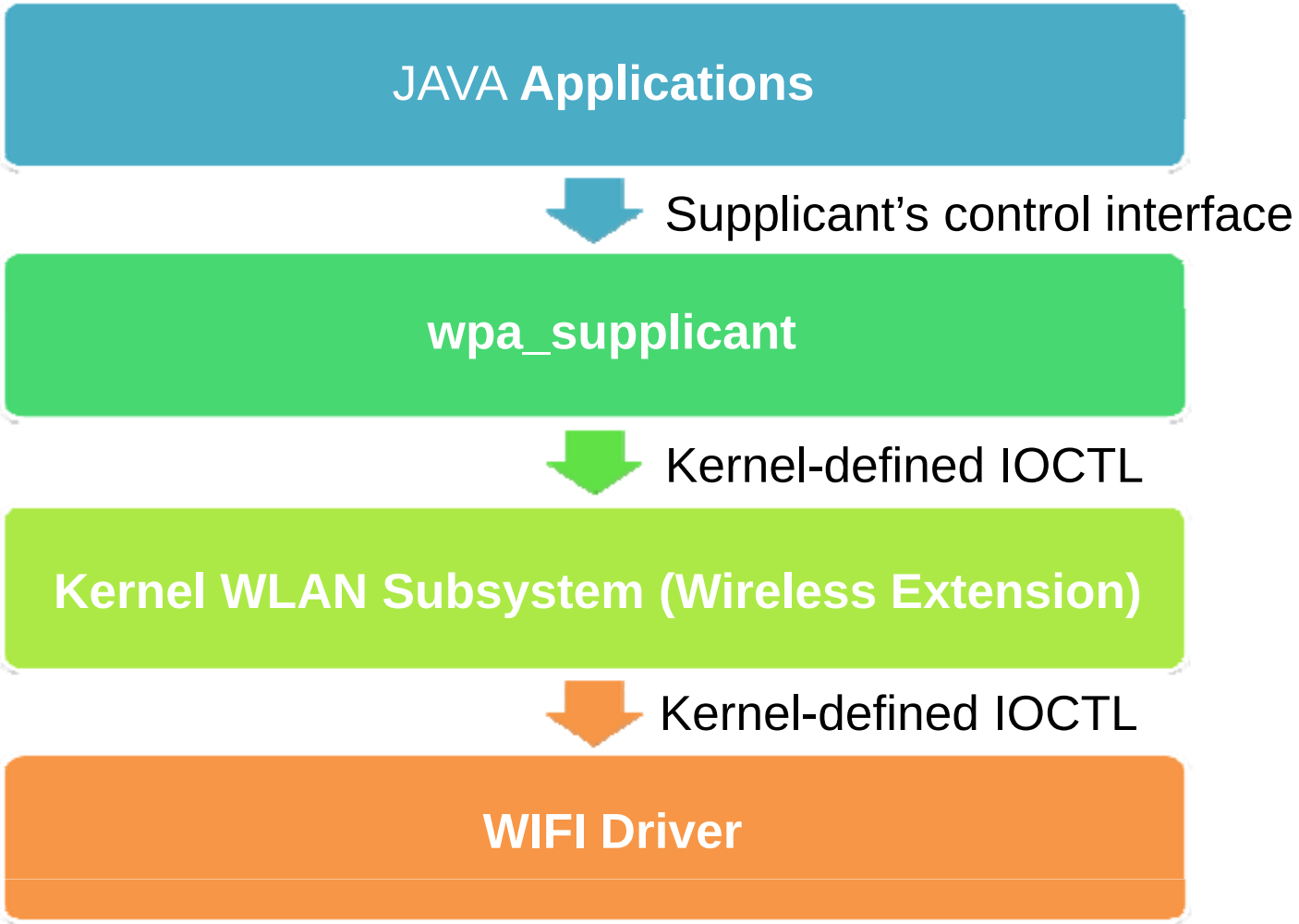
Agenda

- Integration for Android
  - Chip detection, WoW, etc.
  - wpa_supplicant
  - Hotspot AP mode
  - bluetooth.c for BT-COEX
  - Bluetooth clock sharing
  - Google CTS program

WLAN Architecture in Android



WLAN Architecture in Android



The differences for WIFI Driver between Android and Linux

ATHEROS

- Android has it own user-space library.
  - Build our wlan tools, wmiconfig, bmiloader, eeprom in Android SDK tree with Android.mk file
  - Or we can compile WIFI driver's application tools by static link.
- Android doesn't support a comprehensive script.
  - Add test.c patch into android sh to support [ ] function in script
    - We still have problem to reload the firmware after resume
  - Implement all the script function into driver, like WinCE
    - Pass driver module parameters by android property API
- Focus on Linux MMC stack
  - Most customer request to use linux mmc stack

Modify <WIFI Driver>/host/localmake.linux.inc

- ATH_ANDROID_ENV := yes
  - Build for Android system
- ATH_SOFTMAC_FILE_USED :=yes
  - Use MAC address which is stored in /system/wifi/softmac
- ATH_HTC_RAW_INT_ENV := yes
  - TCMD support
- ATH_AR6K_OTA_TEST_MODE := no
  - Don't enable power saving for OTA mode
- ATH_CROSS_COMPILE_TYPE := $(ANDROID_SDK)/…/toolchain/…/arm-eabi-
- ATH_BUS_TYPE:=SDIO
- ATH_BUS_SUBTYPE:= linux_sdio
- ATH_LINUXPATH := <Your kernel source path>
  - Android/out/target/product/xxxx/obj/KERNEL_OBJ
- Type 'make' to make the driver
  - host/.output/$(ATH_BUILD)-$(ATH_BUS_TYPE)/image/ar6000.ko

How to Verify WIFI Driver Manually

- Put firmware and driver module in the folder
  - /system/wifi
    - ar6000.ko
    - athwlan.bin.z77
    - data.patch.hw2_0.bin
- Load WIFI module
  - insmod /system/wifi/ar6000.ko

How to Compile WIFI Driver within Android SDK

ATHEROS

1. Copy AR6kSDK.xxx/android into your android/
   - android/system/wlan/atheros/AR6kSDK.xxxx
   - android/external/ath_supplicant-0.6.10
   - android/external/hostapd-0.6.8
2. Patch your SDK with corresponding patch file
3. Ensure Settings of Android and AR600x
   - The defines in olca/host/Android.mk file
   - Android/device/<vendor>/<platform>/BoardConfig.mk
     - BOARD_WLAN_ATHEROS_SDK:=system/wlan/atheros/AR6kSdk.xxx
     - WPA_SUPPLICANT_VERISON := VER_0_6_ATHEROS
4. Build the Android sdk
   - make -j4 PRODUCT-<platform>-eng

How to Verify WIFI Driver Manually

- Kernel debug messages will show “WMI is Ready” if WIFI bring-up successfully
- We can verify WIFI basic functions by a Linux application package “Wireless Tools” now
- Scan APs by iwlist
  - iwlist –i <interface> scan
- Connect to an AP by iwconfig
  - iwconfig –i <interface> essid <ESSID>

WIFI Driver Architecture: Some Important Files

ATHEROS

- host/include/ar6000_api.h
  - ar6000_drv.c header file
- host/os/linux/include/osapi_linux.h
  - Includes OS API definitions (osapi_linux.h)
- host/os/linux/include/wlan_config.h
  - Configuration for wifi driver
- host/os/linux/include/athdrv_linux.h
  - IOCTL assignment and descriptions

WIFI Driver Architecture: Some Important Files

ATHEROS

- host/os/linux/include/ar6000_drv.h
  - Various data structures
- host/os/linux/include/ieee80211_ioctl.h
  - Various IOCTL data structures
- host/os/linux/include/osapi_linux.h
  - OS-dependent-function re-mapping (printk, locks, etc)
- host/os/linux/ar6000_drv.c
  - Main driver functions (start/stop, TX/RX)

WIFI Driver Architecture: Some Important Files

ATHEROS

- host/os/linux/ar6000_android.c
  - Main driver functions for Android
- host/os/linux/ar6000_pm.c
  - Power management for wifi driver
- host/os/linux/ioctl.c
  - IOCTL-related functions
- host/os/linux/wireless.c
  - The functions for the kernel “wireless extension”

WIFI Driver Architecture: Some Important Files

- ar6000_drv.c
  - Startup/Shutdown functions
    - Initialization/destruction of WLAN states
    - Buffer allocation and buffer free
  - TX/RX interfaces
    - Buffer management (enqueue and dequeue)
    - QoS: IP TOS to WLAN Priority
- wireless_ext.c / ioctl.c
  - Driver ←→ Application Interface
    - IOCTLs (application to driver)
    - Event notifications (driver to application)
- ar6000_pm.c
  - Power management for wifi on/off, suspend/resume
- ar6000_android.c
  - Android related helper functions

WIFI Driver Architecture: Startup/Shutdown functions

- ar6000_init_module ()
  - Initialize HTC/Dataset/GPIO states
  - Register ar6000_avail_ev() and ar6000_unavail_ev()
  - Register WIFI driver to kernel network subsystem via module_init()
- ar6000_cleanup_module ()
  - Clean up HTC states
  - Unregister WIFI driver via module_exit()

WIFI Driver Architecture: Startup/Shutdown functions

- ar6000_avail_ev()
  - HTC calls this when Target is ready
  - Registers device driver function calls
    - netdev->hard_start_xmit ←→ ar6000_data_tx()
    - netdev->open ←→ ar6000_open()
    - Netdev->stop ←→ ar6000_close()
    - netdev->do_ioctl ←→ ar6000_ioctl()
    - netdev->get_stats ←→ ar6000_get_stats()

- ar6000_unavail_ev()
  - Clean up HTC and remove allocated devices

WIFI Driver Architecture: Startup/Shutdown functions

ATHEROS

- ar6000_open ()
  - Execute BMI sequence
  - Register events to HTC (send, receive, has data, etc.) for all endpoints
  - Provide HTC with Rx buffers
  - Start HTC (enable target and interrupts)

- ar6000_close()
  - Block further incoming buffers to Tx from upper layer
  - Disconnect STA from AP and clear WLAN states
  - Stop HTC

WIFI Driver Architecture: TX/RX Interfaces

ATHEROS

- ar6000_data_tx() – called from network layer
  - For transmitting data frames
  - Prepare WMI and HTC headers
  - En-queue to the corresponding endpoint
    - If QoS is enabled, use IP TOS for mapping
- ar6000_control_tx() – called from WMI
  - For transmitting control messages
- ar6000_tx_complete()
  - Called by HTC event (scheduled by interrupt)

WIFI Driver Architecture: TX/RX Interfaces

ATHEROS

- ar6000_rx()
  - Called by HTC event (scheduled by interrupt)
  - Remove WMI and HTC headers
  - 802.3 conversion
- ar6000_data_rx_data_refill()
  - Prepare WMI and HTC headers
  - En-queue buffers to HTC queue

WIFI Driver Architecture: Android related

- android_ioctl_siwpriv()
  - Do Android custom DRIVER command (RSSI, ComboScan)
- android_ar6k_check_wow_status()
  - Hold wake lock if we have wow pattern before suspend
- android_readwrite_file()
  - Load firmware
  - Load software mac address
  - Write driver debug message to logcat
- android_send_reload_event()
  - Send HANG event to supplicant to recovery error

WIFI Driver Architecture: Power related

- ar6000_suspend_ev()/ar6000_resume_ev()
  - Suspend/Resume function
- ar6000_set_wlan_state()
  - Setup Wifi On/Off state
- ar6000_set_bt_hw_state()
  - Setup Bluetooth state for clock sharing
- ar6000_setup_cut_power_state()
  - Setup state of cut power mode for WiFi
- ar6000_setup_deep_sleep_state()
  - Setup state of deep sleep for WiFi
- ar6000_wow_suspend()/ ar6000_wow_resume()
  - Setup WoW for Wifi

Android WIFI Flows: Background

ATHEROS

- Android uses WIFI through the wpa_supplicant interface which use wireless extensions to control our driver
- WIFI driver is only loaded after enabling WLAN from Android GUI
- wpa_supplicant is used as the backend for WLAN control and security connections (WEP/WPA)
- Others, like TCP/IP packet flow and so on, are based on standard Linux implementation.

Android WIFI Flows: File Architecture

ATHEROS

- Application Framework:
  - Folder: base/wifi/java/android/net/wifi
  - Files: WifiManager, WifiMonitor, WifiConfiguration, etc
- Service Framework:
  - Folder: base/services/java/com/android/server/
  - Files: WifiService, WifiWatchdogService, etc
- JNI (Java Native Interface)
  - jni/android_net_wifi_Wifi.cpp

Android WIFI Flows: File Architecture

ATHEROS

- HAL (Hardware Abstract Layer)
  - hardware/libhardware/wifi/wifi.c
  - wpa_supplicant
- WIFI driver: ar6000.ko

Android WIFI Flows: WIFI Turn On Flow

ATHEROS

- Application Framework: WifiManager
  - setWifiEnabled()
- Service: WifiService
  - setWifiEnabled()
  - handleMessage() with MESSAGE_ENABLE_WIFI
- JNI: android_net_wifi_Wifi.cpp :
  - JNINativeMethod: loadDriver
  - android_net_wifi_loadDriver()
- HAL: hardware/libhardware/wifi/wifi.c
  - wifi_load_driver()
- Kernel: insmod ar6000.ko

Android WIFI Flows: WIFI Scan Flow

- Application Framework: WifiManager
  - startScan()
- Service:
  - WifiService::startScan()
  - WifiNative::scanCommand()
- JNI: android_net_wifi_Wifi.cpp
  - scanCommand ←→ android_net_wifi_scanCommand
  - scanResultsCommand ←→ android_net_wifi_scanResultsCommand

Android WIFI Flows: WIFI Scan Flow

- HAL: hardware/libhardware/wifi/wifi.c
  - wifi_send_command : SCAN / SCAN_RESULTS
  - wpa_supplicant/driver_wext.c
  - wpa_driver_wext_scan() : SIOCSIWSCAN
- Kernel: WIFI driver handles the commands of wireless extension interface

Add-ons funcitons in android wpa_supplicant

ATHEROS

Invoke by driver_cmd callback of wpa_supplicant driver_ops which will call android_ioctl_siwpriv() in ar6000_android.c eventually.

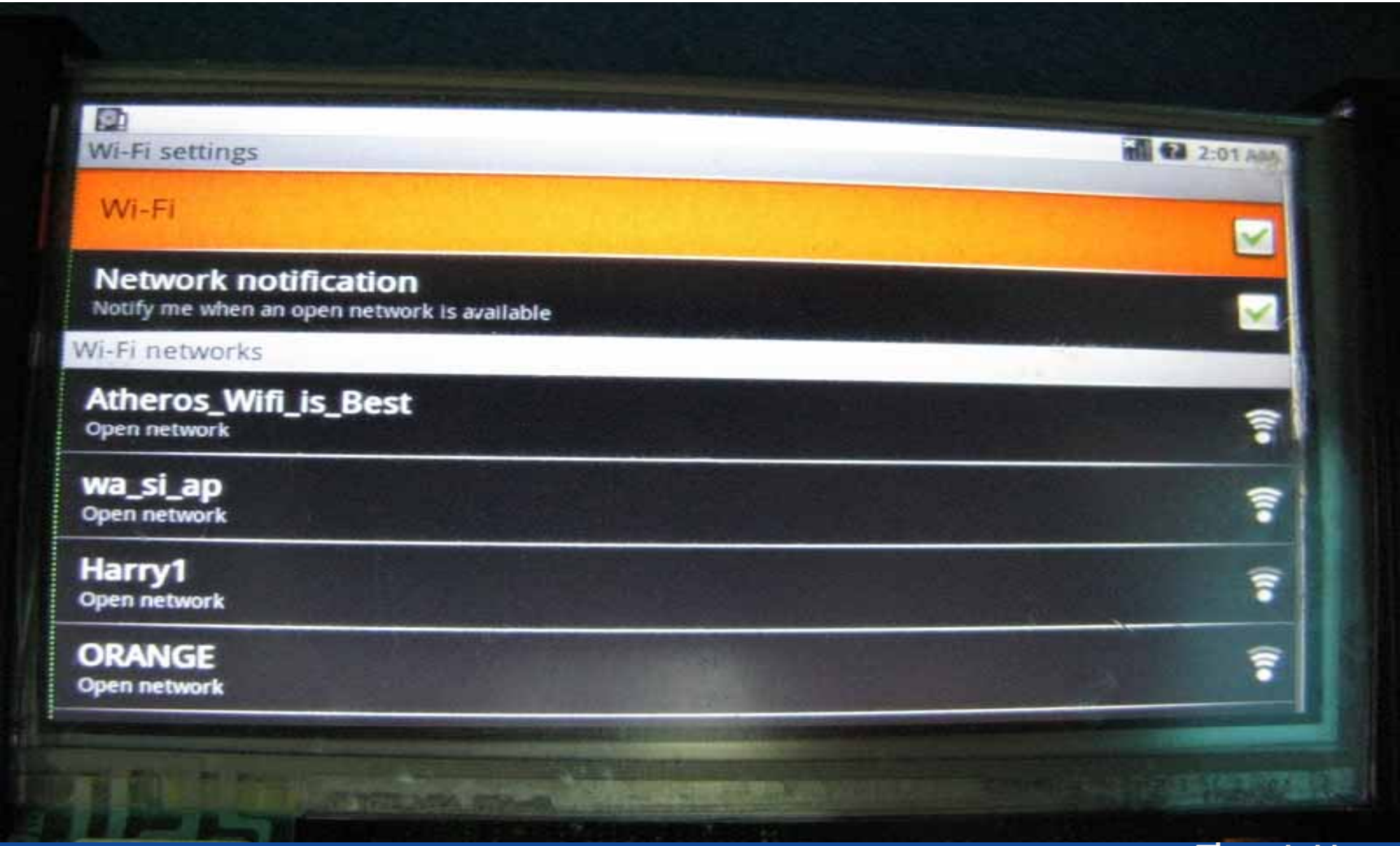
- RSSI
  - Get the rssi
- LINKSPEED
  - Get the current tx/rx rate
- MACADDR
  - Get the mac address of wlan card
- START
  - Enable the wlan based on wifi sleep policy
- STOP
  - Disable the wlan based on wifi sleep policy
- POWERMODE
  - Enable/Disable power saving, mainly for dhcp request
- SCAN-CHANNELS
  - Setup the regular domain
- CSCAN (combo scan)
  - Setup scan Parameters
    - Channels, dwell, passive settings
- SETSUSPENDOPT
  - Suspend mode options
- BTCOEXMODE
  - Btcoex for dhcp obtaining
- RXFILTER-START
  - Multicast filter

Wifi Facilities in Android

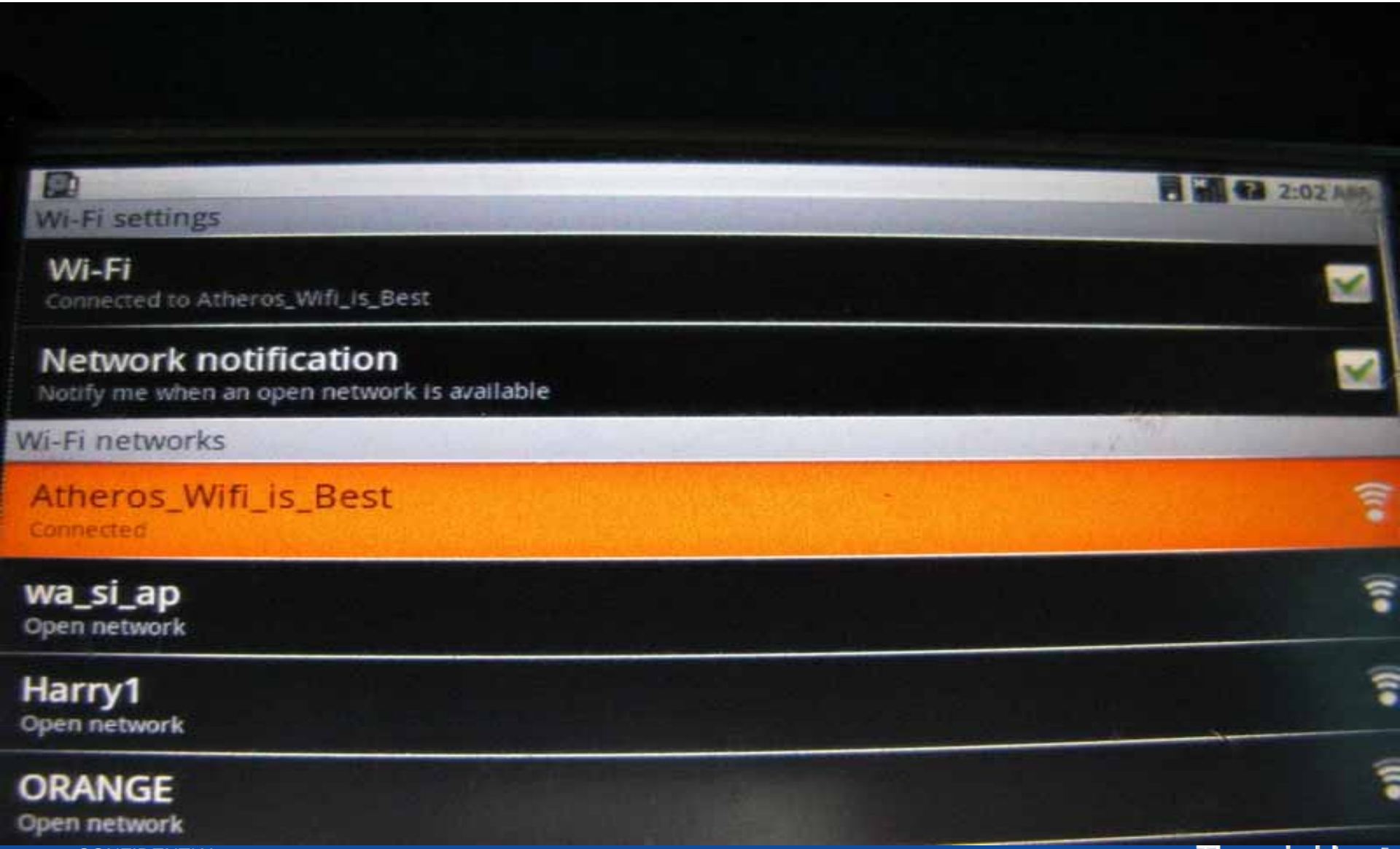
ATHEROS

- Services
  - Native process which can be launched by Java and CPP
  - Setup in /init.rc
    - wpa_supplicant
    - dhcpcd
    - abtfilt
    - hostapd
    - wlan_tool
      - Programmable script
      - Wrapper above services
      - Wifi on/off
      - Configuration after boot up
- Configuration
  - Android property system
    - Network interface name
  - SQLITE3 database
    - Wifi parameters used for Java code
      - Settings.Secure.getInt(cr, Settings.Secure.WIFI_NUM_ALLOWED_CHANNELS)
  - wpa_supplicant.conf
    - Network profiles
  - Certificate
    - Keystore system

Android WIFI Flows: AP Scan List



Android WIFI Flows: AP connected



Android evolution

	Donut	Éclair	Froyo	Gingerbread	Honycomb
Date	15 Sep 09	12 Jan 10	20 May 10	06 Dec 10	22 Feb 11
Kernel	2.6.29	2.6.29	2.6.32	2.6.35	2.6.35
MMC Stack	Suspend unsupported		Cut-power suspend	WoW/Deep Sleep	Combo card 8 bits mode
Supplicant	0.5.11	0.5.11	0.6.10	0.6.10+ CSCAN	0.6.10+ CSCAN
Hotspot	No	No	Yes	Yes	Yes
NAT	No	No	Yes	Yes	Yes
DRIVER CMD	RSSI LINKSPEED MACADDR START/STOP POWERMODE	+COUNTRY		+GETBAND +GETPOWER +SET-SUSPENDOPT	+CSCAN (parameters)
DRIVER EVENT	START STOP	+HANGED		Enhance HANGED	

Questions

- Which Android version support combo scan?

A. Froyo **B.** Donut **C.** Gingerbread

- Which file do I need to modify to build AR600x driver with Android SDK?

A. BoardConfig.mk **B.** init.rc **C.** bluetooth.c

- Which modules do Android call the WEXT IOCTL?

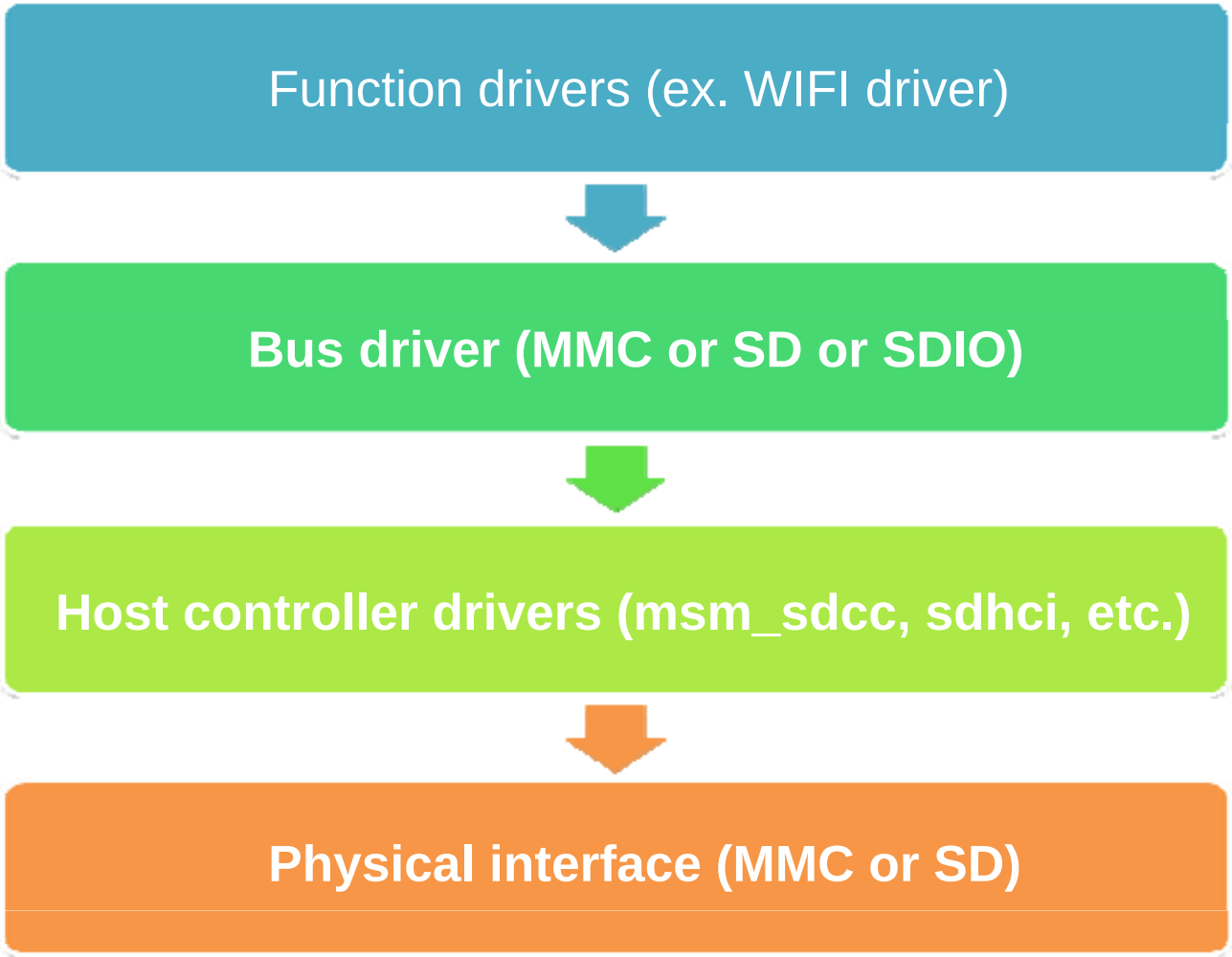
A. WifiManager **B.** wpa_supplicant **C.** JNI

Agenda

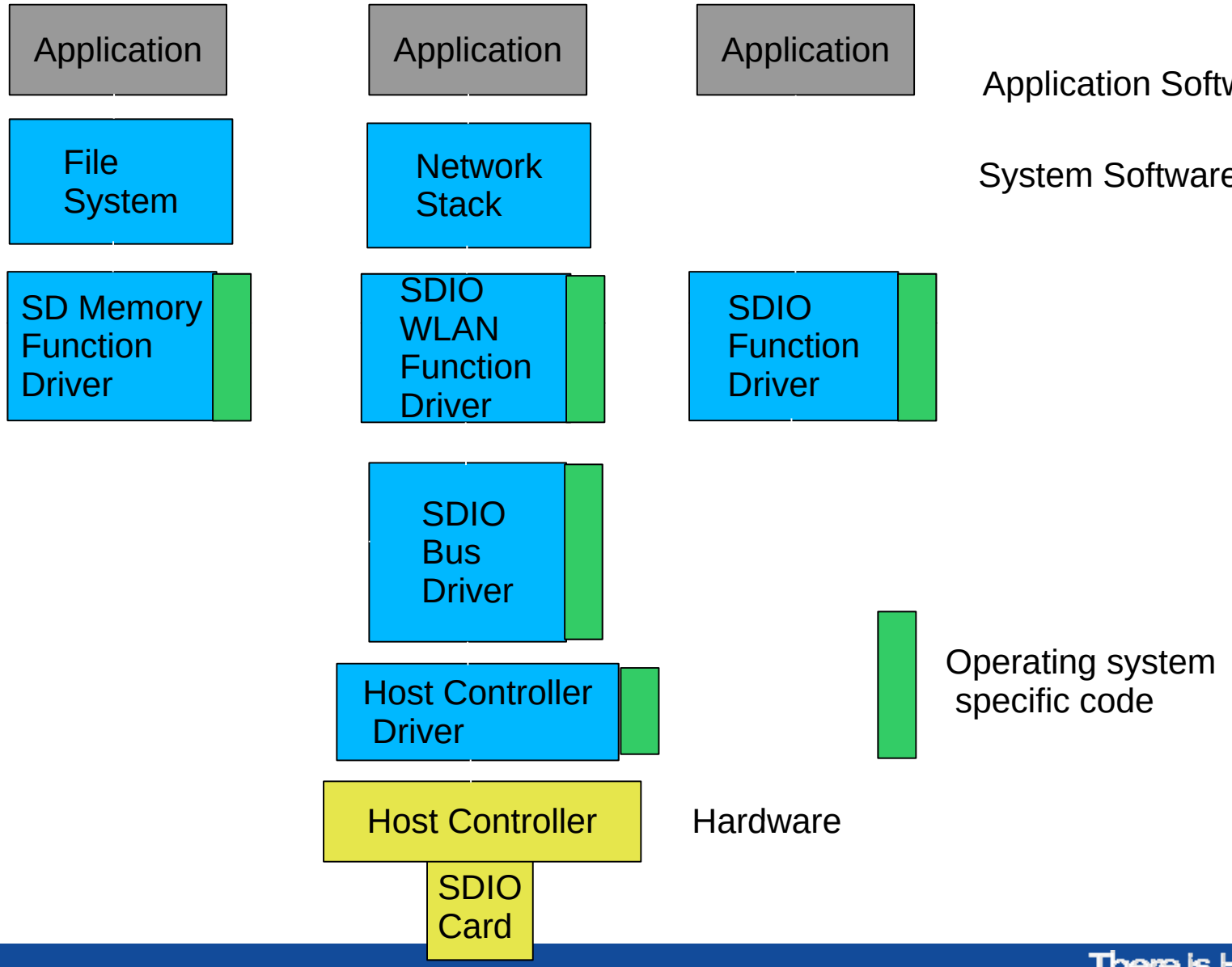
ATHEROS

- Linux MMC Stack
  - Working Model
  - Bus Driver: MMC, SD, SDIO
  - The Interface between Bus Driver and Host Controller Driver
  - The Interface Between Bus Driver and Function Driver
  - Bus Driver Card Polling Mechanism
  - SDIO Card Initialization
  - Interrupt Process

Working Model



Working Model

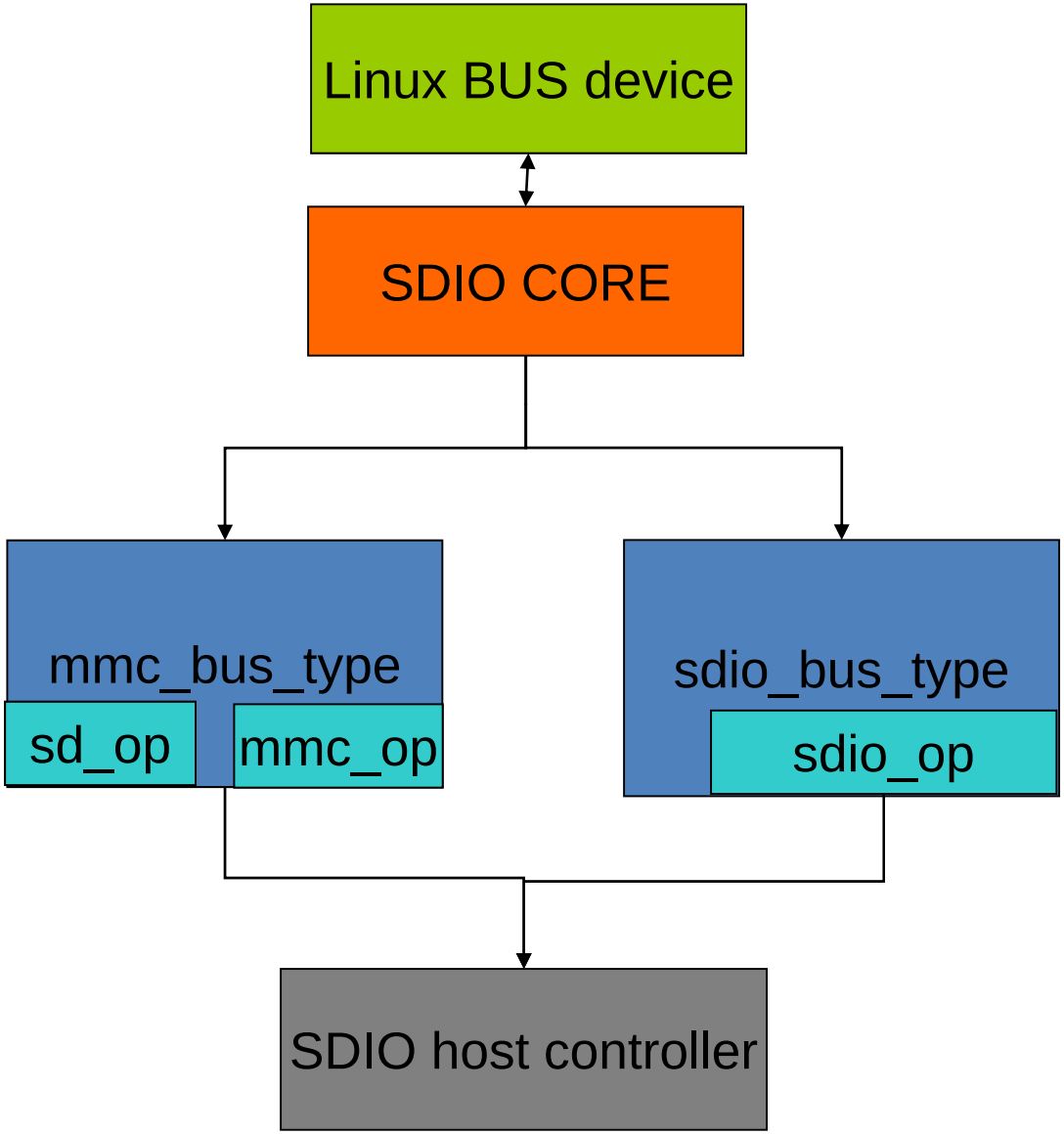


Bus Driver: MMC, SD, SDIO

ATHEROS

- 1 bus driver kernel module
  - core.ko
- 2 kinds of bus type structures (for match, probe, remove, etc.)
  - Memory (MMC/SD): mmc_bus_type
  - IO (SDIO): sdio_bus_type
- 3 kinds of bus operating structures (for remove, detect, suspend, resume)
  - MMC: mmc_ops
  - SD: mmc_sd_ops
  - SDIO: mmc_sdio_ops

Relationship between each SDIO module

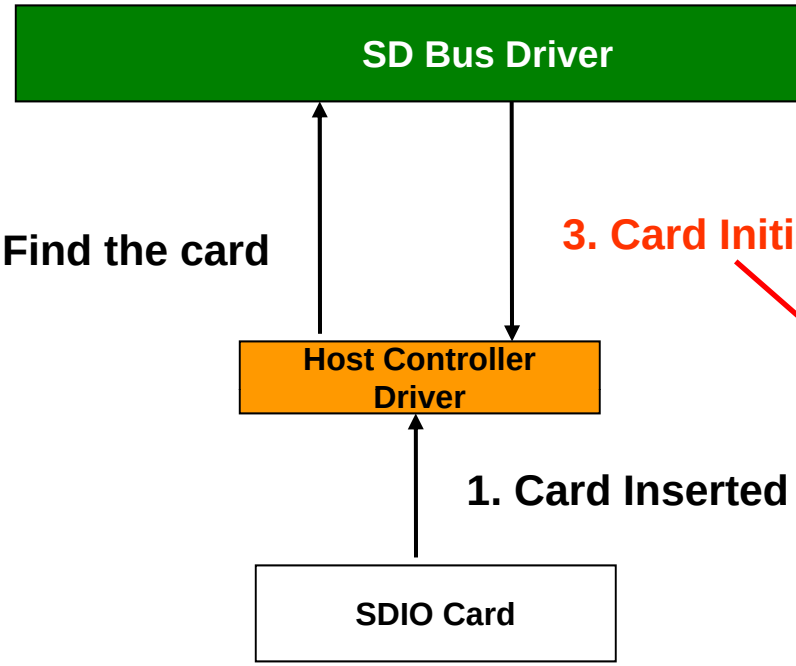


SDIO Card Detection

mmc_rescan() will be called based on host controller platform device

```
mmc_rescan()
    < if host->bus_ops == NULL >
        mmc_go_idle()                  ← CMD0
        mmc_send_if_cond()
        /* sdio? */
        mmc_send_if_cond()             ← CMD5
        < if ok >
            mmc_attach_sdio()
        /* normal SD? */
        mmc_send_app_op_cond()         ← CMD41
        < if ok >
            mmc_attach_sd()
        /* MMC? */
        mmc_send_op_cond()             ← CMD1
        < if ok >
            mmc_attach_mmc()
    < else >
        host->bus_ops->detect()        ← CMD7
```

SDIO Card Initialization



- 1 bit, low clock rate bus mode
- OCR Assignment (CMD5)
- Card Type Detection
- RCA Assignment (CMD3)
- Card Selection (CMD7)
- Start CMD52
- Read the common registers
- Common CIS Query
- Function CIS Query
- Switch to high speed
- Change to the card's maximum speed
- Switch to wider bus
- Initialize all present functions
- Final Bus Mode

The Interface between Bus Driver and Host Controller Driver

ATHEROS

- mmc_alloc_host()
  - Allocate memory for host controller structure
- mmc_add_host()
  - Register the host
  - The host must be ready to serve requests before it
  - Card polling at least 1 time
- mmc_signal_sdio_irq()
  - Inform bus driver of an IO IRQ event
- structure mmc_host_ops
  - request, set_ios, enable_sdio_irq

The Interface between Bus Driver and Function Driver

ATHEROS

- sdio_register_driver()
  - Register a function driver
- structure sdio_driver
  - probe, remove
  - drv.pm.suspend, drv.pm.resume
- sdio_claim_irq()
  - Claim the IRQ

The Interface between Bus Driver and Function Driver

ATHEROS

- sdio_writesb()
  - Write to a FIFO
- sdio_readsb()
  - Read from a FIFO
- sdio_memcpy_toio()
  - Write a chunk of memory
- sdio_memcpy_fromio()
  - Read a chunk of memory

The requirement of claim and release

- Every MMC (SDIO) API need to be locked by mmc claim function
- No asynchronous command is allowed → lower throughput
  - Example
    - sdio_claim_host()
      - sdio_writesb()
      - sdio_memcpy_toio()
    - sdio_release_host()

- Asynchronous support in olca
  - We create a async_task() by kernel_thread
  - We queue all HIF (SDIO) request into linked-list without claim mmc host
  - async_task() will check the pending data in linked-list and call mmc (sdio) command in its context
  - Only async_task() will call sdio_claim_host() and sdio_release_host() in this case

SDIO API Internals

- Implemenation of sdio_mempy_xxxx()

sdio_claim_host(mmc_host)

```
DECLARE_COMPLETION_ONSTACK(complete);
struct scatterlist  sg;
struct mmc_command cmd = { .opcode = SD_IO_RW_EXTENDED };
struct mmc_data    data = { .sg = &sg, .sg_len = 1 };
struct mmc_request  mrq = { .cmd = &cmd,   .data= &data };
data->blksz = blksz;     data->blocks = blocks;     data->flags = flags;
sg_init_one(sg, buf, blksz * blocks);
cmd.flags = MMC_RSP_SPI_R5 | MMC_RSP_R5 | MMC_CMD_ADTC;
cmd.arg = ARGS(rw, fn,  bmode,  incr, addr, blocks_or_bytes);
mmc_set_data_timeout(data, device->func->card);
mrq.done_data = &complete;
mrq,.done = mmc_wait_done;
mmc_host->ops->request(mmc_host, &mrq);
wait_for_completion(&complete);
```

sdio_release_host(mmc_host)

MMC API Internals – Complete Callback

```
void mmc_wait_doen(mmc_request *mrq)
{
   /* DMA tasklet or PIO IRQ context */
   complete(mrq->done_data);
}
```

MMC Scatter-Gather Internals

- DoHifReadWriteScatter() in hif_scatter.c
  - Use mmc core API directly

- SDIO API
  - It always uses 1 buffer for mmc API

```
struct scatterlist  sg;

sg_init_one(&sg, buf, blksz * blocks);

data.sg = &sg;
data.sg_len = 1;
```

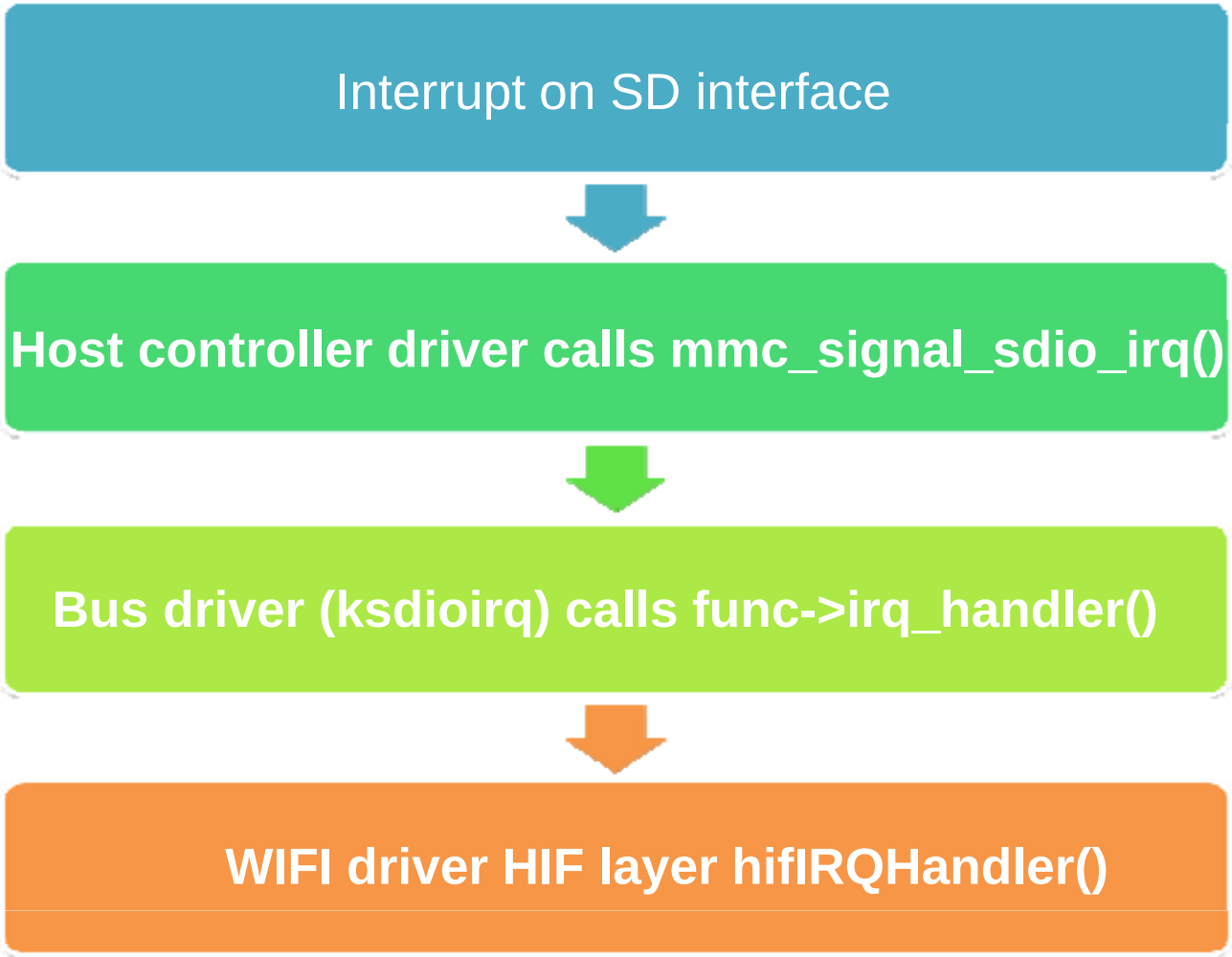
- MMC API
  - We can assign our scatter buffer directly

```
struct scatterlist  sg[Entries];

sg_init_table(&sg, Entries );
for (i = 0 ; i < Entries ; i++, pSg++) {
   sg_set_buf(&sg,  buf[i],  len[i]);
}

data.sg = &sg;
data.sg_len = Entries;
```

Interrupt Process

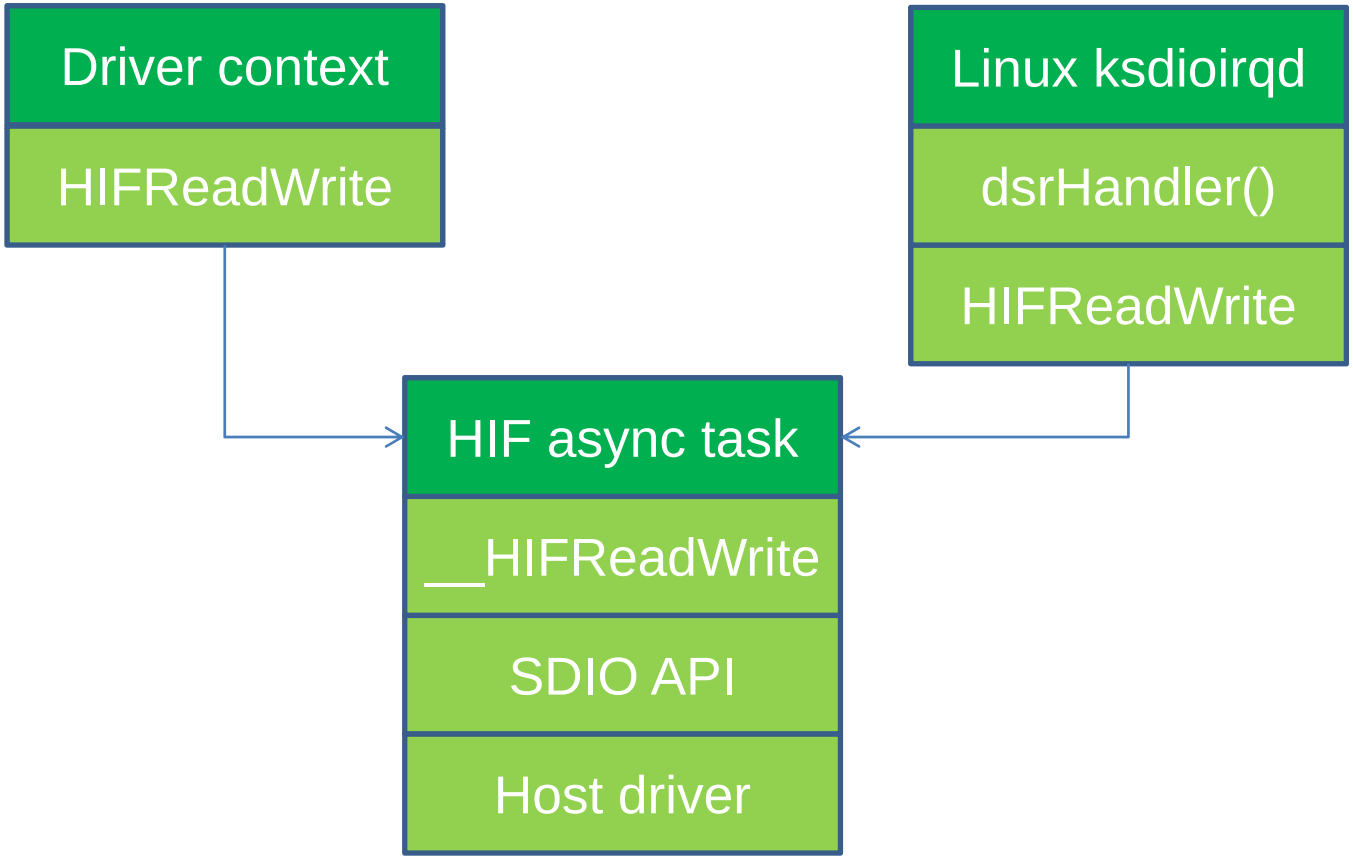
- Bus driver creates a kernel thread for interrupt process:
  - ksdioirqd
- Host controller driver can wake up this kernel thread for the bottom-half process of an IO interrupt event
  - mmc_signal_sdio_irq()
- Host controller driver indicates interrupt support by the below flag
  - mmc->caps |= MMC_CAP_SDIO_IRQ

Interrupt Process



SDIO API integration for HIF layer

- All SDIO read/write operations are done by HIF async_task
- SDIO host will be claimed inside async_task
- Here are the kernel task relationship inside kernel



Linux MMC Suspend and Resume

ATHEROS

- Linux version < 2.6.32 (Android < 2.2)
  - No native MMC suspend and resume method
- Linux version == 2.6.32 (Android 2.2 Froyo)
  - Cut power mode is supported
  - Clock, high speed mode and bus width need to be reset after resume
- Linux version > 2.6.34 (Android > 2.2)
  - Deep sleep mode is support
    - MMC_PM_KEEP_POWER
  - WoW is supported
    - MMC_PM_WAKE_SDIO_IRQ and MMC_PM_KEEP_POWER
    - Bus width will to switch to 1 bit mode after suspend and restore to 4 bits mode after resume

Elements in Suspend and Resume

ATHEROS

- PM operations for SDIO

```
struct dev_pm_ops pmops = {
    .suspend = hifDeviceSuspend,
    .resume = hifDeviceResume,
};
struct sdio_driver ar6k_driver = {
    .name = "ar6k_wlan",
    .id_table = ar6k_id_table,
    .probe = hifDeviceInserted,
    .remove=hifDeviceRemoved,
    .drv = { .pm = &pmops },
};
```

- PM flags for mmc host
  - MMC_PM_KEEP_POWER
    - Deep sleep or WoW mode
  - MMC_PM_WAKE_SDIO_IRQ
    - Wake on WoW via SDIO bus

Suspend and Resume Flow

Suspend Flow

- core.c, mmc_host_suspend()→ host->bus_ops->suspend()
  - sdio.c , mmc_sdio_suspend() →func->pmops->suspend()
    - hif.c, hifDeviceSuspend() → deviceSuspendHandler()
    - ar6000_pm.c, ar6000_suspend_ev()
      - host->pm_flags |= MMC_PM_KEEP_POWER →WoW/Deep sleep
      - host->pm_flags = 0 → Cut power mode

Resume Flow

- core.c, mmc_host_resume()→host->bus_ops->resume()
  - sdio.c , mmc_sdio_resume()→func->pmops->resume()
  - hif.c, hifDeviceResume()→deviceResumeHandler ()
    - ar6000_pm.c, ar6000_resume_ev()

Questions

- Which Linux version support WoW suspend mode?

A. 2.6.27 **B.** 2.6.32 **C.** 2.6.34

- Which one below is NOT MMC bus type?

A. SDIO **B.** UART **C.** MMC

- Which one below is NOT supported by Linux mmc stack?

A. WoW **B.** SDIO **C.** Asynchronous request

Agenda

- Integration for Android
  - Chip detection, WoW, etc.
  - wpa_supplicant
  - Hotspot AP mode
  - bluetooth.c for BT-COEX
  - Bluetooth clock sharing
  - Google CTS program

Card detection and WoW

ATHEROS

- Detection mode
  - Polling
    - mmc->caps & MMC_CAP_NEEDS_POLL
  - One-time detection by exporting mmc_detect_change() in /sysfs
  - Callback funciton of platform_device
    - SMDK : int (*__ext_cd_init__)(void (*_notify_func_)(platform_device *, int state))
    - Qualcomm: int (*__register_status_notify__)(void (*_callback_)(int present, void *), void *)
  - HW-IRQ
- Configuration
  - host/os/linux/include/wlan_config.h
    - #define PLAT_WOW_GPIO_PIN 0
    - #define PLAT_WLAN_CHIP_PWD_PIN 0
- Implementation

void plat_setup_power(AR_SOFTC *dev, int on, int detect) {

1. Setup WLAN power with WLAN_CHIP_PWD_PIN
2. Call platform device callback function to do the detection

}

wpa_supplicant

- Android supplicant is NOT wpa_supplicant
  - 0.5.11 when Android < 2.2
  - 0.6.10 after Android >= 2.2
  - supplcaint with customize feature
    - DRIVER cmd since cupcake
    - Two ctrl_iface sockets since cupcake
    - Special certificate Read/Write without using file since Donut
    - keystore for certificate read/write since Éclair
    - New keystore API after Froyo
    - ComboScan since Gingerbread
  - Known issues
    - Unicode display issue
    - Unable to connect if AP does not support PMK for RSN
    - WPA connection issue when probe response and beacon IE are not same
    - GUI will be pending on obtaining screen after WPA rekey
    - Long connection time after adding a new non-existing network
- The known issues need either patch or using ath-supplicant-0.6.10

Configuration of wpa_supplicant

- Fille device/<vendor>/<device>/BoardConfig.mk
- Using ath_supplicant-0.6.10
  - WPA_SUPPLICANT_VERSION := **VER_0_6_ATHEROS**
  - BOARD_WPA_SUPPLICANT_DRIVER := WEXT
- Using Android supplicant
  - WPA_SUPPLICANT_VERSION := **VER_0_6_X**
  - BOARD_WPA_SUPPLICANT_DRIVER := WEXT

Command line for wpa_supplicant

- wpa_cli -i wlan0 -p/data/misc/wifi/
- Scan
  - scan
- Scan resutls
  - scan_results
- Driver command
  - DRIVER RSSI
- Adjust log level
  - log_level 1

Hotspot mode

- Wireless Router mode tethering 3G network
  - OPEN/WPA2-PSK only
- SDK files related to hotspot mode
  - frameworks/base/core/res/res/values/config.xml
  - frameworks/base/services/java/com/android/server/WifiService.java
  - system/netd/SoftapController.cpp
  - system/netd/NetlinkHandler.cpp

Hotspot mode implemenation

- frameworks/base/core/res/res/values/config.xml
  - Change <tether> as wlan1
  - Change <upstream> as 3G or Ethernet port
- android/system/netd/SoftApController.cpp
  - Need to change BRCM proprietary IOCTL into hostapd daemon
- SoftapController::setSoftap()
  - Write new hostapd.conf based on setSoftap() parameters
  - Change station into AP mode by IOCTL
  - Change wlan interface name from wlan0 to wlan1
  - Start hostapd daemon
  - Simulate Netlink event to notify station(wlan0) interface is down while AP(wlan1) is up
- SoftapController::stopSoftap()
  - Stop hostapd daemon
  - Change back to station mode
  - Change wlan interface name back to wlan0
  - Simulate Netlink event to notify AP interface is down while station is up

Testing for Hotspot mode

ATHEROS

- Configure by command line
  - Start wifi driver
    - ndc softap start
  - Start AP mode
    - ndc softap startap
  - Configure AP mode parameters
    - ndc softap set wlan0 wlan1 [ssid] [security] [key] [channel] [preamble] [max stations]

- Configure by database
  - sqlite3 /data/data/com.android.providers.settings/databases/settings.db
    - Update secure set value="<new value>" where name="<ap field name>"
  - Available field name
    - wifi_ap_ssid
    - wifi_ap_security
    - wifi_ap_passwd

BT-COEX

- BT filter
  - Bt filter will configure AR6003 automatically based on DBUS and HCI events
- Patch file
  - android/system/bluetooth/bluedroid/bluetooth.c
    - Start btfilter service in bt_on()
    - Stop btfilter service in bt_off()
- Debug btfilter in host side using logcat
  - logcat abtfilt:* *:s
- HCIUTILS library
  - Add-on library provide by bluetooth vendor when there is not HCI device
  - Main features
    - Event for the beginning and end of BT-Inqury, SCO.
    - EDR and role infomation of the A2DP devices
  - Notes
    - Event for beginning and end of A2DP can be obtained by DBUS

Bluetooth clock sharing

- Share wifi clock source to bluetooth chip
  - Wifi firmware need to be loaded before bluetooth turning on.
- Basic configuration
  - host/os/linux/include/wlan_config.h
    - #define WLAN_CONFIG_BT_SHARING 1
    - #define WLAN_CONFIG_PM_SUSPEND 3 (Cut power if bt is off)
  - host/tools/wlan_tool/wlan_tool
    - CLK_HELPER=1
- Patch file
  - android/system/bluetooth/bluedroid/bluetooth.c
    - Call IOCTL to ar6000 to indicate bluetooth new ON/OFF state

Google CTS program

- Basic rule to pass Google CTS (Compatibility Test Suite) program
  - No program which is running with root permission for a long time.
- Solution
  - Avoid root service for wlan in init.rc
    - E.g. service wpa_supplicant /system/bin/wlan_tool wpa_supplicant
      user wifi
      group wifi system inet net_admin net_raw keystore
  - Root service will not fork any background program
  - Root service will switch back to normal permission after launched.
    - E.g. dhcpcd

Questions

- Which one is necessary for BT-COEX

A. hostapd **B.** abtfilt **C.** wpa_supplicant

- Which one is related to hotspot mode?

A. SoftApController.cpp **B.** WifiManager.java **C.** bluetooth.c

- Which program can control supplicant?

A. wpa_cli **B.** ndc **C.** svc wifi

THANK YOU

Q & A