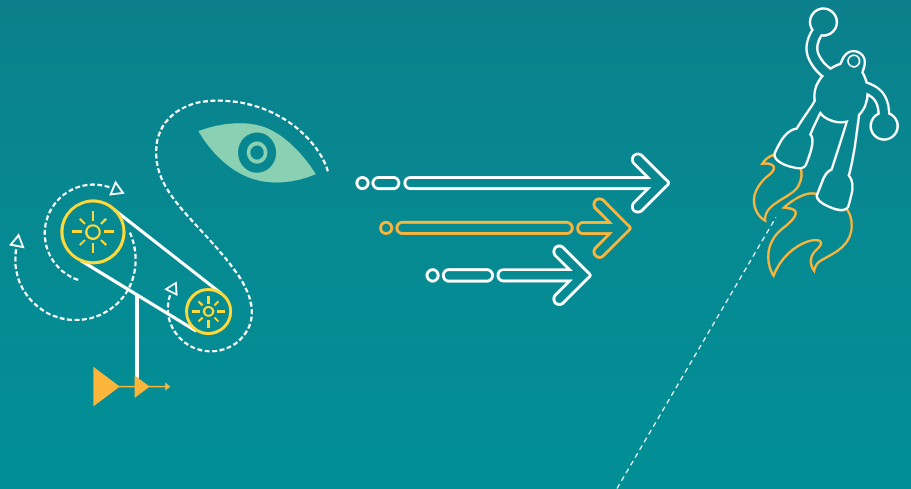
Feature Description

MDM+QCA61x4(A)/QCA9377 Mobile AP

Qualcomm Atheros, Inc.

80-Y7674-41 Rev. N



Qualcomm Atheros, Inc.
1700 Technology Drive
San Jose, CA 95110
U.S.A.

Revision history

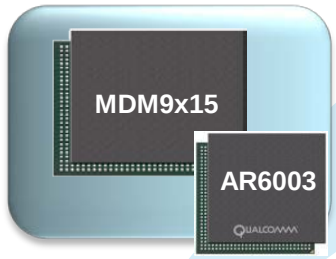
Revision	Date	Description
A	November 2013	Initial release
B	December 2013	Added host driver architecture
C	January 2014	Removed slides 10 through 12; updates to slides 15 and 17
D	February 2014	Updated slides
E	April 2014	Updated the introduction for more functions
F	April 2014	Fixed some minor issues
G	May 2014	Updated Green AP feature with the latest status. Add the introduction for QCMobileAP Automatic Shutdown feature and Automatic Channel Selection feature.
H	May 2014	Added the introduction for Configurable SCC/MCC Support for QCMobileAP-STA mode feature
J	August 2014	Updated with MDM9x40/45 information
K	March 2015	Updated title, added slides for ACS and OBSS, and Antenna Switch for Power Consumption
L	May 2015	Updated with MDM9x50 information, and added introduction for Thermal Throttling feature.
M	September 2015	Added QCA9377+MDM9x07 with SDIO • Added slides 7, 9, 11, 53, and 58-66. • Updated slides 29 and 30
N	October 2015	• Added software feature configuration index and merged with ACL document.

Table of contents

Section 1

MDM Mobile AP overview

MDM-WLAN roadmap (1 of 2)



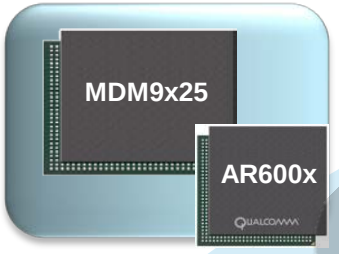
2012

Solution to scale/tier product offerings

AR6003G → 1x1 SB

AR6003X → 1x1 DB

- 5 GHz Wi-Fi support
- Mature SW stack



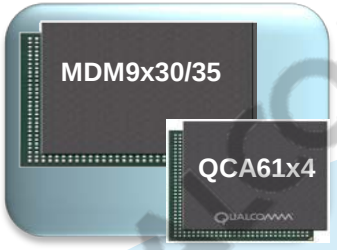
2013

Multi-tier offering:

AR6003G → 1x1 SB

AR6004 → 2x2 DB

- 2x2 MIMO
- Using HSIC
- LTE COEX (channel avoidance)
- MCC in AP-AP and AP-STA modes



2014

Advanced tiers:

QCA6164 → 1x1 11ac

QCA6174 → 2x2 11ac

- 802.11ac
- Using low-power PCIe
- DFS master support in QCMobileAP mode
- WCI: 2-wire advanced LTE COEX

2014

Advanced tiers:

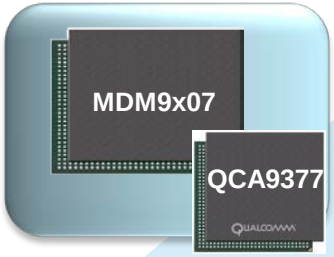
QCA6174A → 2x2 11ac

- All QCA61x4 features
- WLAN-LTE datapath offload
- Additional RAM/ROM for the future upgrade for more SW features.

2015

MDM-WLAN roadmap (2 of 2)

2016



Advanced tiers:

QCA9377 → 1x1 11ac DB SDIO

- Most QCA61x4 features
- Lower cost

QCA61x4(A) low-power 11ac solution

Features and packages

- QCA6164 is a 1x1 11ac solution.
  QCA6174(A) is a 2x2 11ac solution.
- Single regulated 3.3 V supply operation with optional support for Vbatt through xPFET
- Integrated RF front end, single-ended design
- Wafer-scale package to reduce PCB area
- WLNSP (0.4 mm pitch) to minimize PCB area
- Low-latency, high-performance interface (PCIe) support with advanced low-power state (L1-sub state) for standby power reduction
- Offloading for minimal host utilization at 11ac speeds

Target/QCA61x4(A) schedule	
QCA61x4 ES	Oct 2013
9x30/35 LE 1.0 ES	Jan 2014
QCA61x4 CS	Mar 2014
9x30/35 LE 1.0 CS	July 2014
QCA6174A CS	Aug 2014

Uses integrated PA/LNA and BPF (non-LTE CoEX support). Choice of filter, xpA/xlNA, and other components may vary by BOM. Does not include chip ASP.

WLAN technology	802.11a/b/g/n/ac w/advanced features QCA6164: 1x1 QCA6174(A): 2X2
Best-in-class coexistence	2-wire LTE coex with MDM
Interfaces (supporting low power)	WLAN: PCIe+L1SS/SDIO3.0
PCB footprint (WLNSP) internal PA/LNA, SW filter, COB	~110 mm^2 (QCA6174) ~ 90 mm^2 (QCA6164)
WLAN channel bandwidths	20/40/80 MHz
WLAN TCIP/IP throughput 80 MHz 11ac	QCA6174 (2x2) PCIe-L1SS: ~300 Mbps QCA6164 (1x1) PCIe-L1SS: ~300 Mbps
WLAN power consumption (target, at V_{BATT})	**QCA6174(A)** **Rx (2X2):** MCS7 @ 2.4 GHz 83 mA MCS9 VHT80 @ 5 GHz 198 mA **Tx (2X2):** 11b 11 Mbps 2.4 GHz 688 mA 11g 54 Mbps 2.4 GHz 568 mA 11a 54 Mbps 5 GHz 608 mA MCS7 HT20 5 GHz 590 mA MCS8 HT40 5 GHz 590 mA MCS9 HT80 5 GHz 596 mA **DTIM 1** (1 chain) 1–1.2 mA ; **DTIM 3** (1 chain) ~0.6 mA **QCA6164** **Rx:** MCS7 2.4 GHz 41 mA MCS9 VHT80 5 GHz 125 mA **Tx:** 11b 11 Mbps 2.4 GHz 357 mA 11g 54 Mbps 2.4 GHz 294 mA 11a 54 Mbps 5 GHz 328 mA MCS7 HT20 5 GHz 325 mA MCS8 HT40 5 GHz 335 mA MCS9 HT80 5 GHz 342 mA **DTIM 1** (1 chain) 1–1.2 mA ; **DTIM 3** (1chain) ~0.6 mA
WLAN max Pout (at the chip)	Up to +22 dBm (11b, 2.4 GHz) Up to +20 dBm (OFDM, 2.4 GHz) Up to +17 dBm (OFDM, 5 GHz) Up to +10.5 dBm (11ac)
Power supply	Regulated 3.3 V and DC2B (xPFET)

QCA9377 low-power 11ac solution for MDM9x07 (1 of 2)

Features and packages

- QCA9377-3 supports a low-power SDIO 3.0 interface for WLAN and a UART/PCM interface for Bluetooth.
- Highly integrated WLAN system-on-chip (SoC) for 5 GHz 802.11ac, or 2.4 GHz/5 GHz 802.11n WLAN applications.
- Supports BT 4.1 + HS, BLE, & ANT+; also compatible with BT 1.x and BT 2.x + Enhanced Data Rate.
- Supports a single-ended RF port for a cleaner and lower-cost design.
- Supports 20 MHz/40 MHz at 2.4 GHz and 20 MHz, 40 MHz, or 80 MHz at 5 GHz.
- Supports multiuser MIMO.
- Supports BT-WLAN coexistence and ISM-LTE coexistence.
- Operates on one 3.3-volt power supply and an I/O supply of 1.8 V or 3.3 V.
- Both WLAN and Bluetooth power management use advanced power-saving techniques such as:
  - Gating clocks to idle or inactive blocks
  - Voltage scaling to specific blocks in certain states
    - Fast-start and settling circuits to reduce Tx
  - Active duty cycles
  - Processor frequency scaling
  - Other techniques to optimize power consumption across all operating states

WLAN technology	1x1 802.11a/b/g/n/ac with advanced features
Best-in-class coexistence	Wi-Fi/BT coex engine; LTE-BT coex
Interfaces (supporting low power)	SDIO3.0 4b SDR 104
Antenna configuration	Single Wi-Fi/BT antenna option or separate Wi-Fi/BT antenna
WLAN channel bandwidths	20/40/80 MHz
WLAN power consumption (excluding interface, at VBATT)	**Rx**: MCS7 2.4 GHz 69 mA MCS9 VHT80 5 GHz 125 mA **Tx**: 11b 11 Mbps 2.4 GHz 358 mA 11g 54 Mbps 2.4 GHz 333 mA MCS7 HT20 5 GHz 422 mA MCS8 VHT40 5 GHz 422 mA MCS9 VHT80 5 GHz 421 mA
WLAN max Pout (at the chip)	Up to +22 dBm (11b, 2.4 GHz) Up to +20 dBm (OFDM, 2.4 GHz) Up to +17 dBm (OFDM, 5 GHz) Up to +10.5 dBm (VT80, MCS9 5 GHz)
Power supply	Regulated 3.3 V and DC2B (xPFET)
Reference document	*QCA9377-3 Dual-band 1X1 802.11AC + Bluetooth 4.1 Device Specification* (80-WL431-1)

Uses integrated PA/LNA and BPF (non-LTE CoEX support). Choice of filter, xpA/xlNA, and other components may vary by BOM. Does not include chip ASP.

QCA9377 low-power 11ac solution for MDM9x07 (2 of 2)

Features and packages, cont'd

- Additional features include:
  - Low-density parity check (LDPC)
  - 1.5 kB of on-chip, one-time-programmable (OTP) memory. Eliminates the need for an external flash and further reduces external component count and BOM cost.
  - A 48 MHz reference clock.
  - Bluetooth support for class 1 and class 2 power-level transmissions without requiring an external PA.
  - An internal PA and internal LNA to support the datasheet specifications. Can support an external PA and external LNA with central logic.
  - Available in a low-profile 4.34 mm × 5.48 mm WLNSP package with 0.4 mm pitch.

QCA61x4 vs. QCA61x4A

QCA 61x4	QCA 61x4A
CS: March 2014	CS: August 2014
RAM: 768 KB	RAM: 1216 KB
ROM: 352 KB	ROM: 448 KB

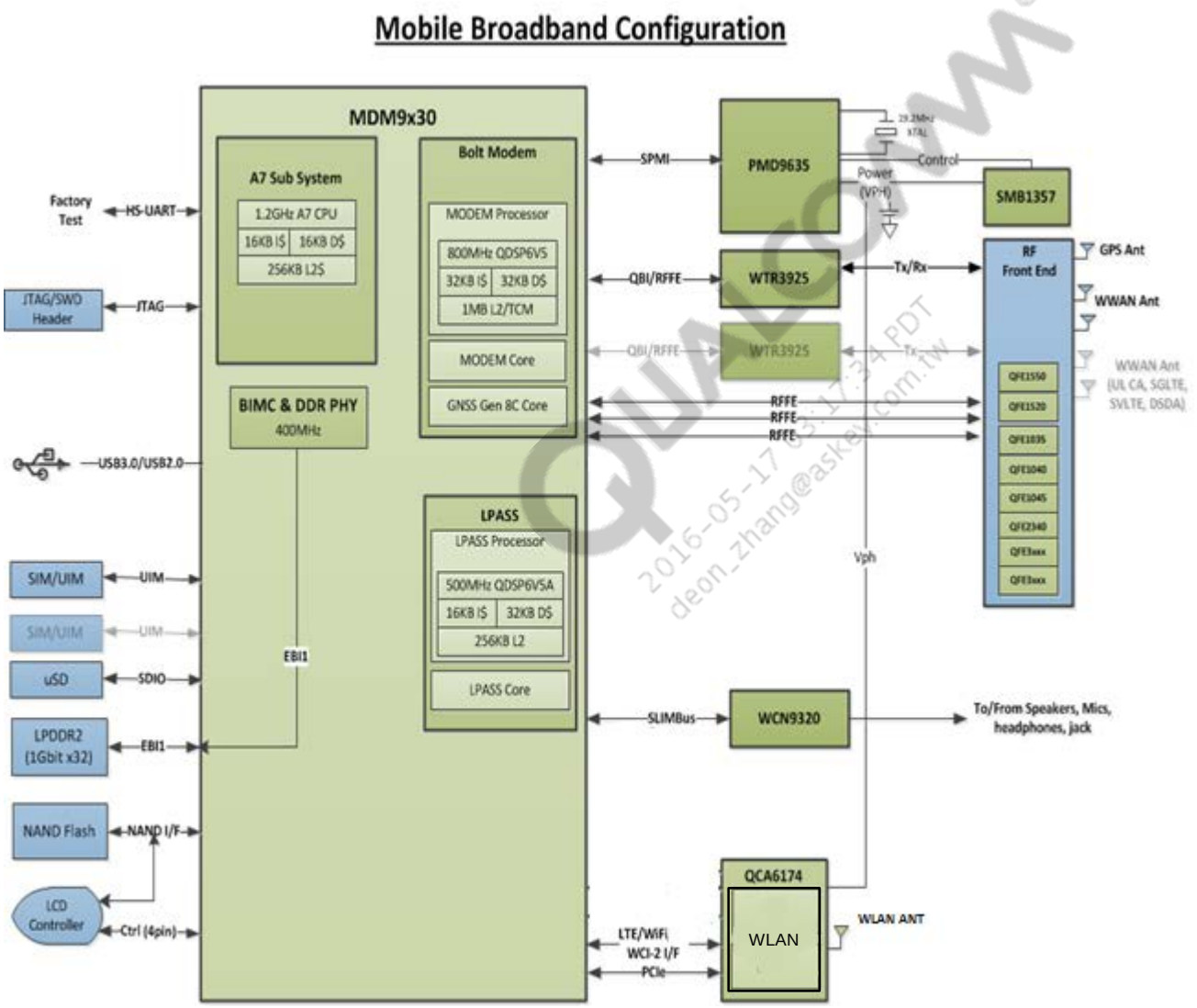
- QCA61x4A has *same* package size as QCA61x4
  - Replaced an IP (target fixed consumer market) with RAM/ROM addition of RAM/ROM.
  - All hardware configurations and modules developed with QCA61x4 are applicable to QCA61x4A with a BOM change.
- Addition of RAM/ROM to allow for upgrade on software features in the future.
- QCA61x4 performance and RF updates are applicable to QCA61x4A.

QCA9377 vs. QCA61x4(A)

QCA 9377+MDM9x07	QCA 61x4A+MDM9x30/40/50
SDIO3.0 interface	PCIe interface
1x1	1x1/2x2
IPA not supported	IPA supported

- QCA9377 has the same low-profile package size as QCA61x4.
- QCA9377 has a different interface: SDIO vs. PCIe
- QCA9377 uses a similar driver/firmware architecture.

System-level block diagram



- All 9x15/9x25 features are carried over.
- PCIe-L1SS interface (low idle power and latency).
- HW acceleration (IPA) to support high end-to-end throughput
- LTE/Mobile router concurrency

Note: IPA was not applicable for MDM9x07 with SDIO 3.0 interface)

HW acceleration: IP Accelerator (IPA)

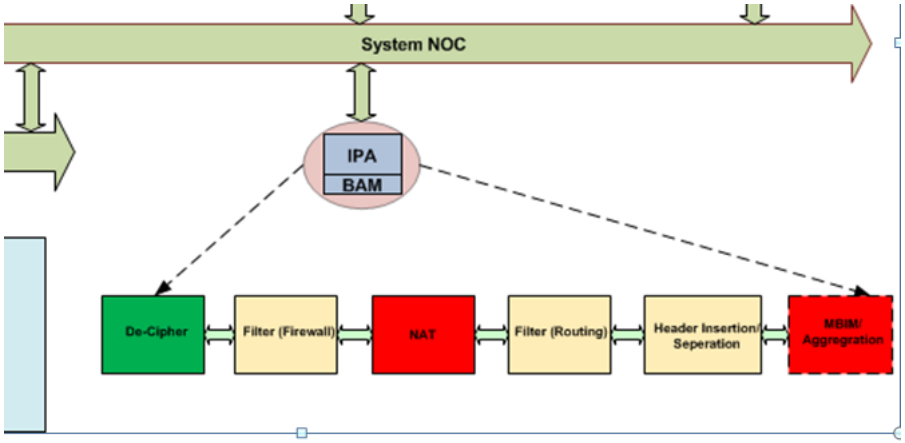
- Increasing WWAN data rates (LTE Cat6 and above) translates to a commensurate increase in MIPS needed to move the data between WWAN and WLAN/PCIe interfaces. IPA helps to offload the burden.

- Source and destination NAT
  - Support NAT on UL and DL datapaths
  - Ability to auto-generate NAT rules based on a configured port range

- IP routing
  - Ability to route packets between any source and any destination based on the rules
  - Header insertion/deletion

- Support addition to IP packets of 802.3/LLC and proprietary headers

- Deciphering
  - Support SNOW 3G and AES decryption for LTE DL Data

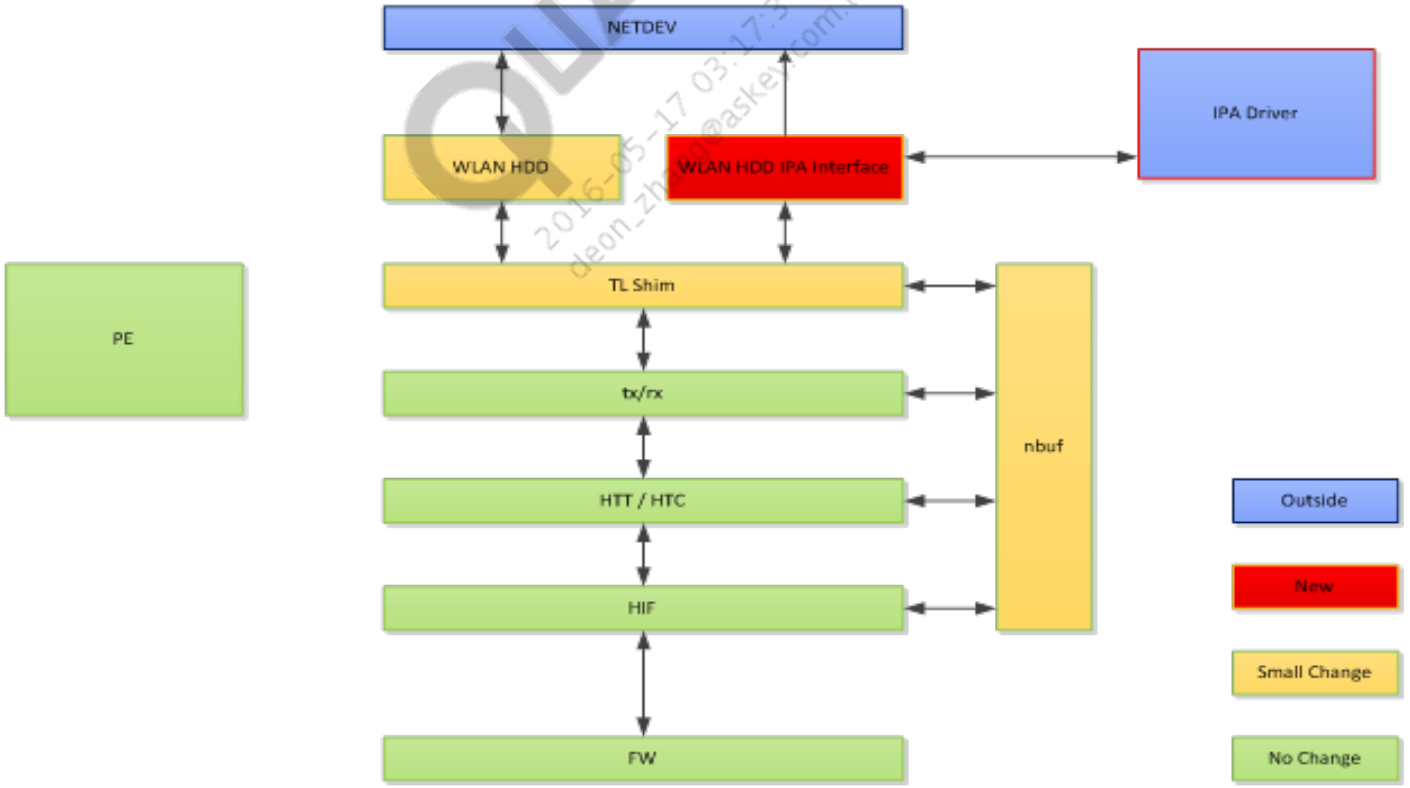


Note: MDM9x07+QCA9377 do not support the IPA feature.

HW acceleration: IPA improvement (1 of 2)

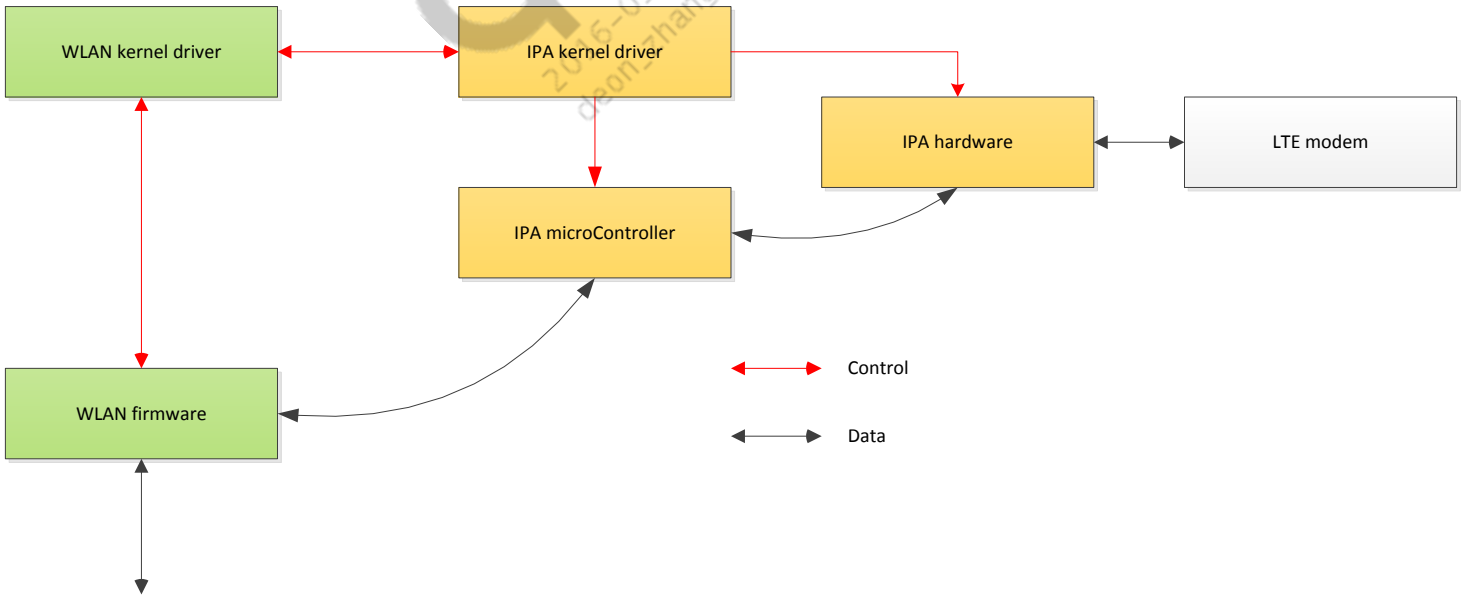
- For QCMobileAP mode + LTE backhaul, most packets are routed between the modem and the WLAN. The destinations of the packets are WWAN over MODEM and STAs connected to QCMobileAP. The MDM device is not the major packet destination most of the time.

- On MDM9x30/35, the IPA feature helps avoid the MDM kernel networking layer involving the routing process if the destinations of the packets are not an MDM device.
  - Throughput between WWAN and QCMobileAP is not limited by MDM application processor ability.
  - Power usage is more efficient.



HW acceleration: IPA improvement (2 of 2)

- Following MDM9x40/45, the IPA feature has been improved as follows:
  - On MDM9x30/35 the WLAN driver is the packet routing module between WLAN firmware and IPA hardware. IPA hardware routes a packet between WLAN and the modem. In this case, even MDM is not the destination of the packet. All of the packet still goes through the WLAN driver and then through the IPA module. This procedure leads to an unnecessary MDM application processor wakeup and thus some extra power consumption.
  - To save power consumption on the MDM AP, the datapath is offloaded to the IPA microcontroller on the MDM. The IPA microcontroller communicates directly with the IPA hardware and WLAN firmware can communicate with the IPA microcontroller. Thus, packet communication is always between the WLAN firmware and the IPA microcontroller and the MDM AP can be out of the QCMobileAP datapath.

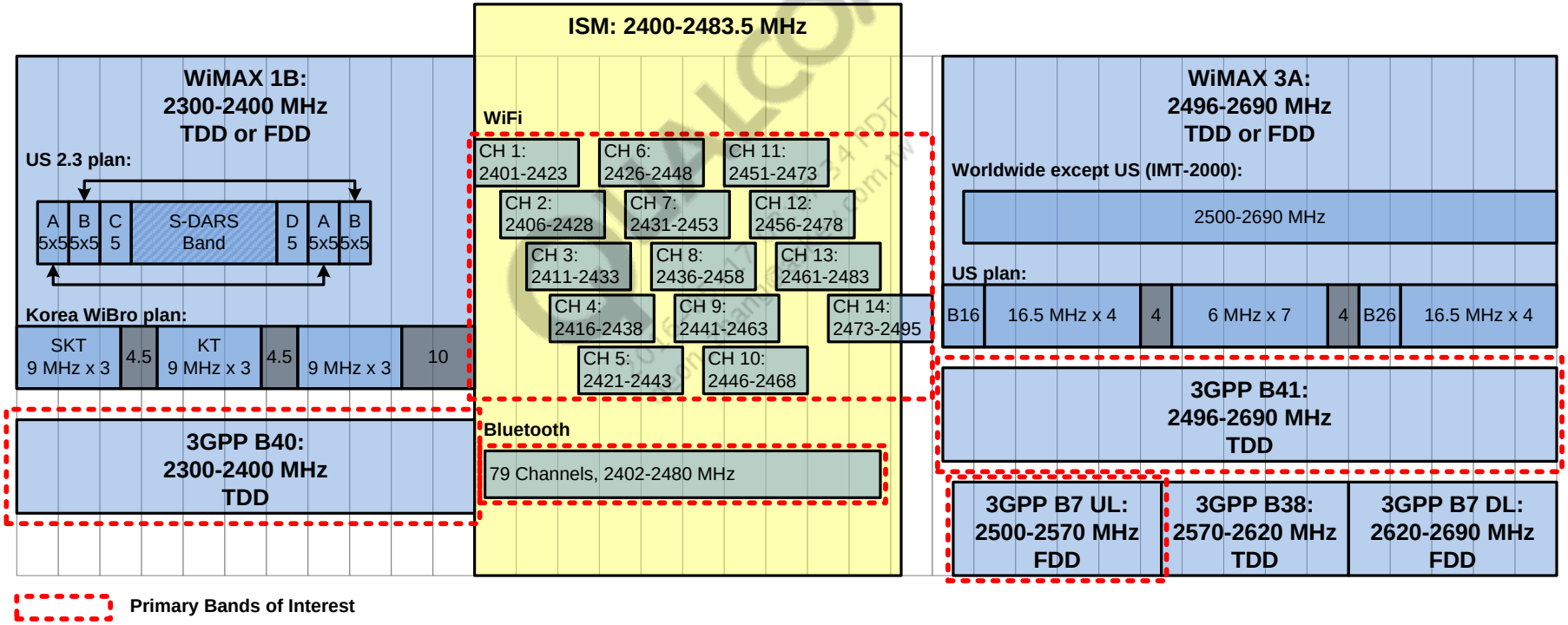


Section 2

QCA61x4(A)/QCA9377 feature set

LTE-ISM coexistence scenario

- WLAN and Bluetooth share the same ISM band.
- LTE shares bands 38/40/41 for TDD and bands 7/20 for FDD uplink. This can create severe interference due to the adjacent and limited bands.



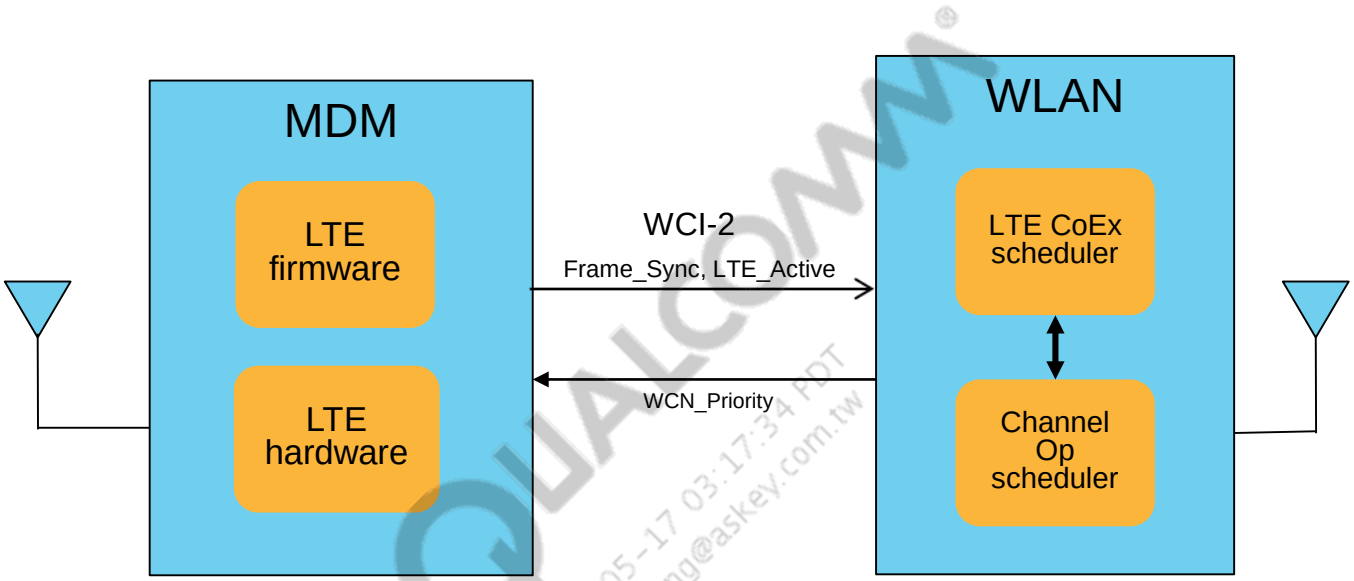
LTE band and RF impact

Band	Impact
7	• When LTE is using the closest channel to the ISM band, WLAN can be desensed across the ISM band, that is, WLAN victim case • WLAN may impact LTE on some channels, that is, LTE victim cases
40	• When LTE uses the top 30 MHz, Bluetooth and WLAN are desensed across the ISM band • When LTE uses the lower 70 MHz, the lower 20 MHz of the ISM band has some desense • When Bluetooth or WLAN uses the lower 20 MHz of the ISM band, it desenses LTE across the band • When Bluetooth or WLAN is above 2420 MHz, only the upper 30 MHz of B40 are desensed
38	Can be solved with filtering
41	Coexistence needed for 2500–2570 frequency range

LTE-ISM coexistence features

Feature	Description
2-Wire UART Interface (to MDM) with fast message control path	
Time Division Duplexing (B40, B41)	
WLAN DL sub-frame TDM (through CTS-to-self)	Prevents WLAN Rx from occurring during LTE Tx. WLAN Tx or Rx during LTE DL or unused UL subframes.
WLAN unused UL sub-frame TDM (opportunistic extension of WLAN usage time)	Similar to above, but only unused UL subframes are used when WLAN Tx power is too high for LTE DL.
WLAN autonomous denials	If LTE is receiving, automatically prevents WLAN from transmitting.
Frequency Division Duplexing (B7)	
Channel selection	AP mode channel selection updated according to LTE UL frequency.
Output Power Control	
WLAN power backoff	Dynamically lowers WLAN Tx power to minimize impact on LTE DL performance.

Two-wire LTE-ISM coexistence implementation



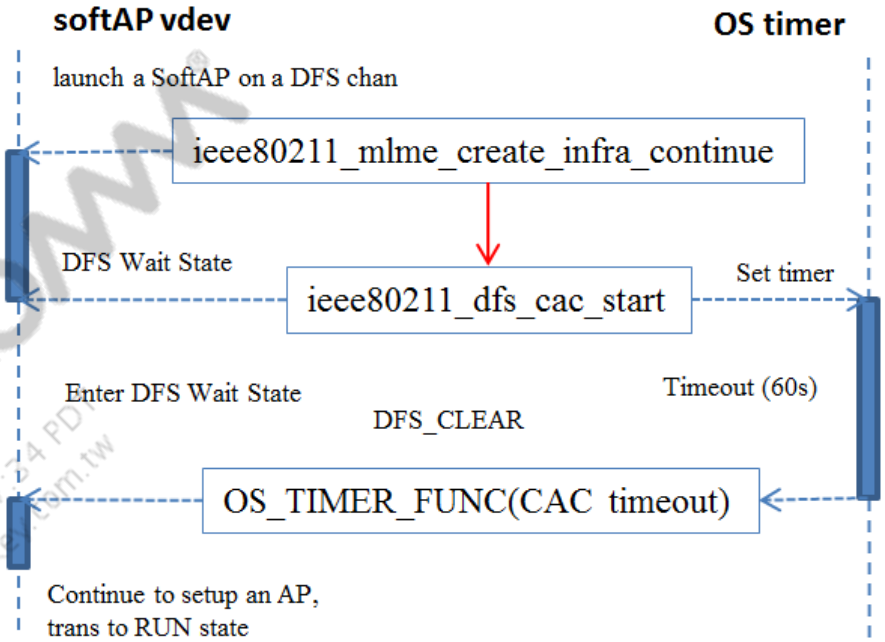
- MDM9x30 and later devices support only two-wire LTE coexistence.
- Maintains a coexistence algorithm similar to the previous three-wire implementation while achieving reduced signal count.
- New message types:
  - Indicate to WLAN the duration of various LTE radio states (for example, LTE radio sleep duration).
  - WLAN indication to LTE radio to reduce transmit power when possible.

DFS (1 of 2)

- QCA61x4(A) can detect radar in the DFS channel under QCMobileAP mode

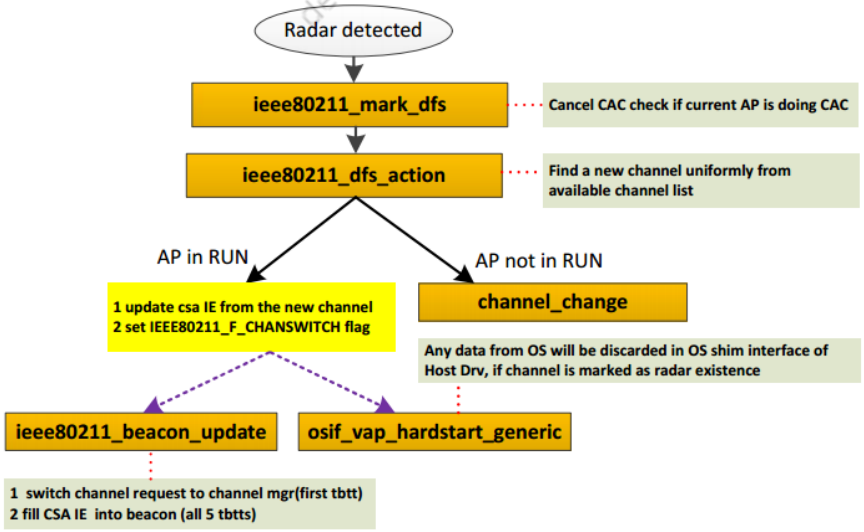
- Channel Availability Check (CAC)
  - CAC is a pure software logic that requires the AP to wait 60 seconds (10 minutes for ETSI weather radar) before it fully enters the RUN state (starts beaconing).
  - If radar is detected during this period, the AP switches to a new channel. If it is still a DFS channel, another CAC period is required.

- No Occupancy List (NOL)
  - Channels marked occupied by radar should not be used for any reason during a period of NOL time. This means no work is allowed on this channel during this time.
  - After the NOL timeout, the channel is a valid candidate if a new channel is selected.
  - Current NOL time is 30 minutes.



DFS (2 of 2)

- When radar is detected on a DFS channel, QCA61x4(A) fills the *Channel Select Announcement IE* in beacon to notify associated STAs.
- If QCA61x4(A) is doing a Channel Availability Check (CAC), that is cancelled because radar has been found.
- Select a new channel from the available channel list uniformly (consider NOL).
- If the AP is not in the RUN state the AP can switch channels immediately.
- DFS channel support under QCMobileAP mode can be enabled and disabled by the following parameter in WCNSS_qcom_cfg.ini*:*
  - # DFS master capability
  - gEnableDFSMasterCap=1



802.11w (PMF)

- IEEE 802.11w is the Protected Management Frames (PMF) standard for the IEEE 802.11 family of standards.
- The target increases the security by providing data confidentiality of management frames.
- PMP is a mechanism to protect 802.11 management frames similar to the way that data frames are protected.
- This standard uses the existing security mechanisms rather than creating a new security scheme or a new management frame format.
- The Wi-Fi Alliance required 802.11w (both Station and QCMobileAP modes) by 07/01/2014.

Green AP

- Th Green AP feature saves power when no stations are connected by switching off some receive and transmit chains. At present it keeps only one receive and transmit chain on when it is triggered.

- The feature automatically detects the station status and adjusts chains accordingly.

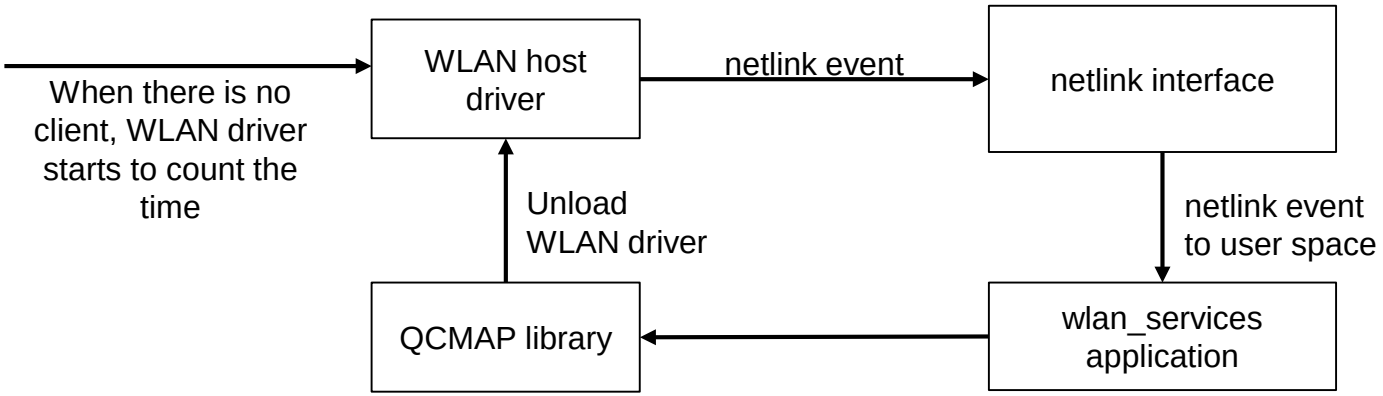
- Green AP is always enabled and it is triggered to save power immediately after mobile AP is turned on but there is no station for 20 ms.The function is enclosed by the macro *FEATURE_GREEN_AP* in the WLAN driver source.

- The limitations of Green AP are:
  - When running in concurrency mode, if there is at least one non-SAP interface, this feature is disabled.
  - When running in concurrency mode, this feature is only enabled when no station is connected to any SAP interface.

Note: This feature was *not* available in QCA9377(1x1) on MDM9x07.

QCMobileAP automatic shutdown

- QCMobileAP automatic shutdown saves power when no stations are connected for the configured timeout value by shutting down QCMobileAP.
- The time to shut down QCMobileAP can be configured by the parameter *gWlanAutoShutdown* in *WCNSS_qcom_cfg.ini* configuration file in */lib/firmware/wlan/qca_cld/*. The value of the parameter is in seconds.
- This feature is supported by the WLAN driver with specific application on the MDM platform.
- The WLAN driver counts the time that there is no client. If there is no client for the configured period, the WLAN driver sends a *WLAN_SVC_WLAN_AUTO_SHUTDOWN_IND* event to the user space application *wlan_services* through the Linux netlink interface.
- The *wlan_services* application then unloads the WLAN driver.
- The working model and event flow are illustrated in the figure.
  - WLAN driver ⟶ WLAN service application ⟶ Unload WLAN driver



Dynamic access control list (1 of 4)

Summary

- The dynamic access control list (ACL) feature is designed to let users dynamically change and modify access control policy and rules without reloading the *hostapd*.
- Dynamic ACL controls access based on MAC address.
- The MDM QCMobile AP solution supports simultaneous *hostapd*-based ACL and dynamic ACL.
- With *hostapd*-based ACL support:
  - The configuration is in the *hostapd* configuration file, *hostapd.conf* with two additional files, for the white and black lists.
  - The configuration is passed to the WLAN driver when *hostapd* is launched.
  - The *hostpad* uses NL80211_ATTR_ACL_POLICY to pass the policy to the WLAN driver with the MAC address list. The policy response may be either NL80211_ACL_POLICY_ACCEPT or NL80211_ACL_POLICY_DENY.
  - The *hostapd* doesn't support dynamically changing the setting after *hostapd* has been working. It is necessary to reload *hostapd* if the setting must be changed.
- With dynamic ACL support:
  - Dynamic ACL is supported by the WLAN driver.
  - The ACL policy and the white and black lists are configured through the *iwpriv* application after the WLAN driver is loaded.
  - A user may dynamically change the settings when the WLAN driver is working.
  - The configuration is lost when the WLAN driver is unloaded.

Dynamic access control list (2 of 4)

hostapd-based ACL details:

- The policy is set in the *hostapd* configuration file, *hostapd.conf*:

  Station MAC address-based authentication

  Note: This kind of access control requires a driver that uses *hostapd* to take care of management frame processing and, as such, this can be used with *driver=hostap* or *driver=nl80211*, but not with *driver=madwifi*.

  0 = accept unless in deny list

  1 = deny unless in accept list

  2 = use external RADIUS server (accept/deny lists are searched first)

  macaddr_acl=0

- The white and black lists are set in the *hostapd* configuration file, *hostapd.conf*:

  Accept (white) and deny (black) lists are read from separate files. Each file contains a list of MAC addresses, one per line. Use absolute path names to make sure the files can be read on SIGHUP configuration reloads.

  accept_mac_file=/etc/hostapd.accept

  deny_mac_file=/etc/hostapd.deny

- The format of the white list and black list files, *hostapd.accept* and *hostapd.deny*, is as follows.

  # MAC List

  68:7F:74:B8:E8:B6

  …

Dynamic access control list (3 of 4)

Dynamic ACL details:

- Four ACL policy types:
  - ACCEPT UNLESS DENIED
    - The DUT accepts the connecting request from a client unless its MAC address is in the black list.
  - DENY UNLESS ACCEPTED
    - The DUT denies the connecting request from a client unless its MAC address is in the white list.
  - USE BOTH ACCEPT AND DENY
    - If a client device is not in both lists, black and white, the DUT denies the client's connecting request and sends an event to notify the user-space that the MAC address is being denied.
  - ALLOW ALL
    - This is the default policy: allow any client to connect to the DUT.

- The macro used in WLAN driver is:

```
typedef enum {
        eSAP_ACCEPT_UNLESS_DENIED = 0,
        eSAP_DENY_UNLESS_ACCEPTED = 1,
        eSAP_SUPPORT_ACCEPT_AND_DENY = 2,
        eSAP_ALLOW_ALL = 3,
} eSapMacAddrACL;
```

- The iwpriv command to configure ACL policy
  - iwpriv <interface name> setAclMode <mode>
  - iwpriv wlan0 setAclMode 0 ← Accept unless denied
  - iwpriv wlan0 setAclMode 1 ← Deny unless accepted
  - iwpriv wlan0 setAclMode 2 ← Support Access + Deny at the same time
  - iwpriv wlan0 setAclMode 3 ← Allow all

Dynamic access control list (4 of 4)

- The *iwpriv* command is used to configure the white list and the black list:
  - iwpriv <interface name> modify_acl <mac addr> <accept list/deny list> <add/delete>
  - iwpriv wlan0 modify_acl 0x68 0x7F 0x74 0xB8 0xE8 0xB6 0 0 ← Add to black list
  - iwpriv wlan0 modify_acl 0x68 0x7F 0x74 0xB8 0xE8 0xB6 0 1 ← Delete from blacklist
  - iwpriv wlan0 modify_acl 0x68 0x7F 0x74 0xB8 0xE8 0xB6 1 0 ← Add to white list
  - iwpriv wlan0 modify_acl 0x68 0x7F 0x74 0xB8 0xE8 0xB6 1 1 ← Delete from white list

- Limitations
  - In the following scenario, the DUT does not automatically disconnect from connected clients when the ACL policy is changed to DENY UNLESS ACCEPTED. It is the current limitation.
    1. A client has been connected to the DUT.
    2. The ACL policy is changed to DENY UNLESS ACCEPTED.
    3. Although the client is not in the white list at this moment, DUT will not automatically disconnect from it.
  - The customers' user-space applications are responsible for disconnecting the connected client.
  - No setting configured by dynamic ACL will be kept after the WLAN driver is unloaded. If it is necessary to keep changes made through dynamic ACL, the user must store those changes in the *hostapd* configuration file and in the white and black list files.

Automatic channel selection (1 of 2)

- The automatic channel selection (ACS) feature automatically selects the best channel with the least interference from other APs.
- When the feature is enabled, QCMobileAP scans the channels first when QCMobileAP is turned on. It collects information from the environment to pick the best channel and then works on that channel.
- ACS picks the best channel by calculating this information:
  - High RSSI value in a channel
  - Number of APs working in a channel
- ACS can be enabled or disabled with the following parameter:
  - On MDM9x30 LE 1.0 – *gApAutoChannelSelection* in *WCNSS_qcom_cfg.ini*
  - On MDM9x30 LE 2.0 and later – *channel=0* in *hostapd.conf*
- The candidate channel is configured to a specific band by the *gAPChannelSelectOperatingBand* parameter in *WCNSS_qcom_cfg.ini*.
- *gAPChannelSelectOperatingBand* can be configured to the following values:
  - 0 – 2.4 GHz
  - 1 – 5 GHz LOW
  - 2 – 5 GHz MID
  - 3 – 5 GHz HIGH
  - 4 – 4.9 GHz
  - 5 – 5 GHz ALL

Automatic channel selection (2 of 2)

- Candidate channels can be configured to a range in the band by the following two parameters in the *WCNSS_qcom_cfg.ini* file:
  - *gAPChannelSelectStartChannel*
  - *gAPChannelSelectEndChannel*

- Candidate channels also can be limited to some specific channels in the band by the *gACSAllowedChannels* parameter in *WCNSS_qcom_cfg.ini*.

- ACS prefers channels 1, 6, and11 by default when the *gAPChannelSelectOperatingBand* parameter is configured to 2.4 GHz and at least one of the channels
  (1, 6, or 11) is in the best channel group from the scan and calculated result.

- This channel preference can be disabled by setting the *gEnableOverLapCh* parameter to 1 in *WCNSS_qcom_cfg.ini*.

- Since MDM9x40 LE 1.1, the candidate channels are configured by the following parameter in hostapd configuration file:
  - *chanlist=<start_ch>-<end_ch> – Add a dash ('-') to provide a range/*
  - chanlist=<ch1> <ch2> <ch3> – Add a space between values. No need to give list length as first parameter. Channel list may also include ranges.

Configurable SCC/MCC support for QCMobileAP-STA mode (1 of 2)

- When MDM products are used, MCC implementation provides a flexible way to use the channel resources most of the time. However, in a few cases the current MCC/SCC implementation in QCMobileAP-STA mode cannot fully satisfy a customer because it is necessary to configure MCC mode or SCC mode in advance.

- One common case is QCMobileAP and STA under MCC, but they are working in adjacent channels. QCMobileAP under MCC mode has to use *CTS-to-Self* packets to notify its clients during a change to the STA channel. These packets can leak to adjacent channels. The 802.11 devices at STA channel are affected by CTS-to-Self packets. This can lead to poor throughput on the STA channel.

- The *Configurable SCC/MCC Support for QCMobileAP-STA Mode* feature was introduced to overcome this issue and provide a better user experience. This feature lets users choose their preferred behavior when there is an MCC case in the same band:
  - Keep the original MCC behavior.
  - Change to the SCC case by moving QCMobileAP to the STA channel when there is a channel overlapping status.[1]
  - Force the SCC case by moving QCMobileAP to the STA channel when STA and QCMobileAP are working in the same band.

1. The new feature automatically decides whether QCMobileAP and STA are in overlapping channels by calculating their center frequencies and bandwidths.

Configurable SCC/MCC support for QCMobileAP-STA mode (2 of 2)

- The *Configurable SCC/MCC Support for QCMobileAP-STA Mode* feature is controlled by this parameter in the *WCNSS_qcom_cfg.ini* file: *gWlanMccToSccSwitchMode*

- This parameter can be set to three values:
  - 0 – Keep MCC.
  - 1 – Change to SCC when QCMobileAP and STA channel overlap is detected.
  - 2 – Force to SCC if QCMobileAP and STA are at the same band.

- The implementation of this feature in the WLAN driver is enclosed by this macro:

  FEATURE_WLAN_MCC_TO_SCC_SWITCH

- To configure WLAN mode, get *Qualcomm® MobileAP API for MDM9x35 Interface Specification* (80-NH740-46) and read this section: *QCMobileAP (SoftAP) LAN Configuration*, *WlanMode* (currently Section 5.1.2.4).

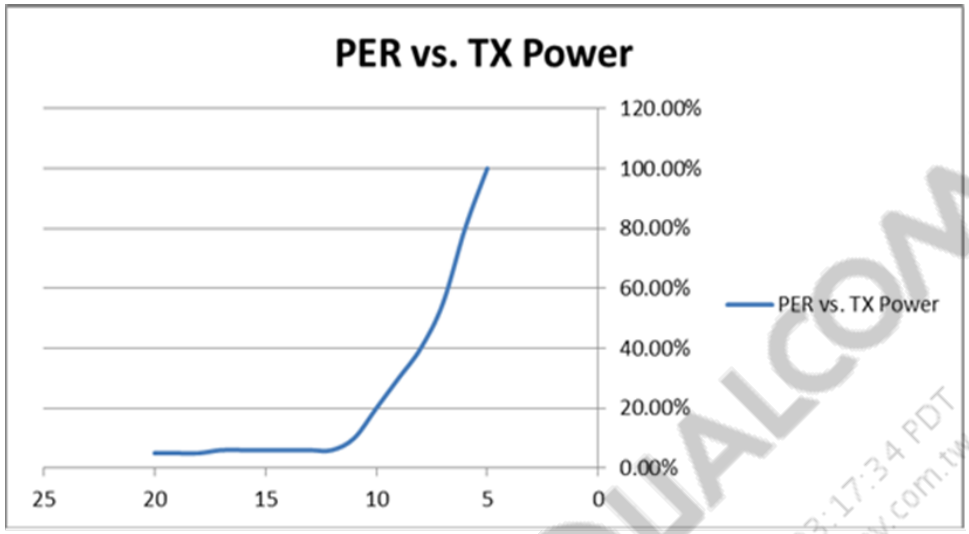
Green Tx (1 of 2)

- Green Tx (GTX) is a proprietary QCA output power control algorithm.
  - Green Tx dynamically adjusts Tx power to ensure the lowest output power is used to maintain the highest PHY rate.
  - The Green Tx algorithm uses PER to determine the lowest output power needed to sustain a link at the highest PHY rate.

- Beneficial use cases of GTX
  - Most mobile routers are short range devices. MIMO APs ensure good signal coverage, which allows Tx power to be throttled down, resulting in significant power savings.
  - When implemented in mobile routers and client devices, Green Tx extends the battery life of both devices as they are typically near each other.

- Basic GTX algorithm
  - Search from the default Tx power from the OTP/HAL PHY rate table.
  - The GTX principal requirement is to not impact throughput. If the device is sitting in the same rate with the last transmit and current PER of this rate is < PER cap (currently 3%) begin to reduce Tx power in steps.
  - If Tx power is reduced while PER persists, continue to reduce power gradually until it fills in the range of PER cap + margin.

Green Tx (2 of 2)



	2G, Single Tone			5G, Single Tone		
	No GreenTx	GreenTx		No GreenTx	GreenTx	
Output power per chain (dBm)	21.4	9.7	1.8	10.6	4.7	0.7
Current consumption per chain(mA)	269	87	69	258	140	128

Transmit power capping

- The user space application uses the private IOCTL command to cap QCMobileAP transmit power in dBm units. The value must be larger than zero (that is, transmit power (dBm) > 0).

- The ultimate capped transmit power is the lowest of the following:
  - User setting from ioctl command
  - Regulatory domain allowed power
  - Calibrated transmit power value

- All frames that include beacons follow the new capped transmit power.

- Private IOCTL command is *setTxPower* <dBm>
  - Example: Capping maximum transmit power to 10 dBm: *iwpriv wlan0 setTxPower 10*

Thermal mitigation (1 of 4)

- Thermal mitigation on WLAN involves monitoring the temperature of the WLAN chip, detecting when a threshold is crossed, and taking necessary actions to reduce the temperature.

- Four levels of temperature threshold ranges are configured by the following *WCNSS_qcom_cfg.ini* file parameters (in */lib/firmware/wlan/qca_cld*.) Based on the threshold range chosen, the WLAN driver controls bus idle time to throttle the throughput and reduce WLAN chip temperature.
  - Level 0 range: gThermalTempMinLevel0, gThermalTempMaxLevel0
  - Level 1 range: gThermalTempMinLevel1, gThermalTempMaxLevel1
  - Level 2 range: gThermalTempMinLevel2, gThermalTempMaxLevel2
  - Level 3 range: gThermalTempMinLevel3, gThermalTempMaxLevel3

- On QCA61x7(A), an on-chip temperature sensor is used to monitor the temperature.

- The architecture of the thermal mitigation feature is as follows:
  - When either a MinLevel or MaxLevel temperature threshold is crossed the WLAN firmware sends a signal to the WLAN driver.
  - When it receives a signal the WLAN driver adjusts the bus idle time to mitigate the temperature increase (or if a thermal MinLevel is crossed, the driver removes the mitigation).

- Bus idle time depends on the level at which the WLAN chip temperature stays:
  - Level 0 range: duty cycle is 100% (Same as the non-throttling case)
  - Level 1 range: duty cycle is 50%
  - Level 2 range: duty cycle is 25%
  - Level 3 range: duty cycle is 12.5%

Thermal mitigation (2 of 4)

- Thermal mitigation is enabled by this parameter in *WCNSS_qcom_cfg.ini:*

  #Enable thermal mitigation

  gThermalMitigationEnable=1

- These *WCNSS_qcom_cfg.ini* parameters establish the range of the four levels:
  - gThermalTempMinLevel0=<min value of level 0 range>
  - gThermalTempMaxLevel0=<max value of level 0 range>
  - gThermalTempMinLevel1=<min value of level 1 range>
  - gThermalTempMaxLevel1=<max value of level 1 range>
  - gThermalTempMinLevel2=<min value of level 2 range>
  - gThermalTempMaxLevel2=<max value of level 2 range>
  - gThermalTempMinLevel3=<min value of level 3 range>
  - gThermalTempMaxLevel3=<max value of level 3 range>

- This *WCNSS_qcom_cfg.ini* parameter sets the bus idle period:

  #Throttle period in ms

  gThrottlePeriod=<customer preference>

- For debugging, use the following iwpriv commands to switch to a specific level immediately when the minimum and maximum values of all level ranges are set to 0 and 255 separately:
  - iwpriv wlan0 setTmLevel 0 – Switch to level 0 immediately
  - iwpriv wlan0 setTmLevel 1 – Switch to level 1 immediately
  - iwpriv wlan0 setTmLevel 2 – Switch to level 2 immediately
  - iwpriv wlan0 setTmLevel 3 – Switch to level 3 immediately

Thermal mitigation (3 of 4)

Limitations

- The thermal mitigation feature only applies to the WLAN Tx datapath. This path generates the most heat. The purpose of this feature is to mitigate temperature increase from the Tx datapath.

- This feature is used to reduce WLAN chip temperature. It is not possible to control either throughput or temperature to a specific value. Thermal mitigation simply controls bus idle time and throttles input to manage overall chip temperature.

- When the Tx datapath is throttled, the factors below are involved in the throughput result. It is impossible to predict throughput when throttling is occurring.
  - Transport protocol layer behavior.
  - Queue length inside the WLAN driver for pause time.
  - How to handle the queued packets when the queue is full.
  - CPU usage status.
  - Throttle period (*gThrottlePeriod*).

- The duty cycle ratio (*setTmLevel*) cannot be mapped to the impact on throughput value.

- When configuring the minimum and maximum values of the four level ranges, overlap the max value of the x level and the min value of the x+1 level (x=0, 1, or 2). This overlap helps the WLAN to switch gracefully between levels. Refer to the first example on the next page.

Thermal mitigation (4 of 4)

- Example 1: test the feature. Monitor WLAN chip temperature to track its status change.
  - gThermalMitigationEnable=1
  - gThermalTempMinLevel0=50
  - gThermalTempMaxLevel0=70
  - gThermalTempMinLevel1=60
  - gThermalTempMaxLevel1=80
  - gThermalTempMinLevel2=70
  - gThermalTempMaxLevel2=90
  - gThermalTempMinLevel3=80
  - gThermalTempMaxLevel3=100
  - gThrottlePeriod=500
- Example 2: test a specific level by using the iwpriv command.
  - gThermalMitigationEnable=1
  - gThermalTempMinLevel0=0
  - gThermalTempMaxLevel0=255
  - gThermalTempMinLevel1=0
  - gThermalTempMaxLevel1=255
  - gThermalTempMinLevel2=0
  - gThermalTempMaxLevel2=255
  - gThermalTempMinLevel3=0
  - gThermalTempMaxLevel3=255
  - gThrottlePeriod=500
  - Switch to level 1 immediately to set a 50% duty cycle. Issue the command *iwpriv wlan0 setTmLevel 1*

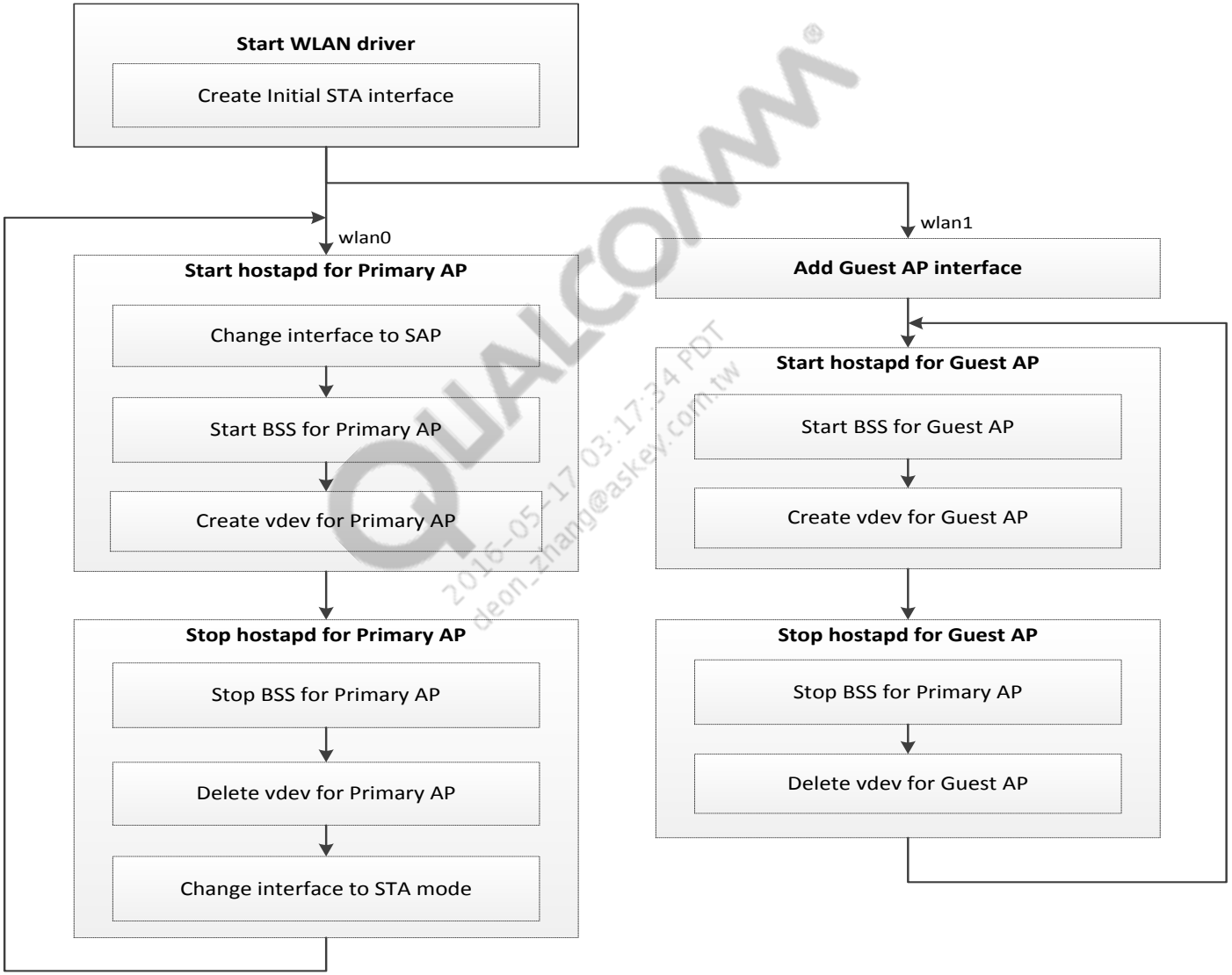
STA+AP concurrency

- MDM9x30 and later devices support STA+AP concurrently.
- The two-device-concurrent operation is achieved by having two virtual interfaces functioning over a single physical interface.
- A single host driver software creates and supports two virtual interfaces.
- The OS IP stack and other applications see the virtual interfaces as individual interfaces.
- STA and AP can work with different security modes.
- STA and AP can operate on the same channel (Single-Channel Concurrency [SCC]) or on different channels (Multi-Channel Concurrency [MCC]) regardless of different radio bands such as 2.4 GHz or 5 GHz.

AP+AP concurrency

- MDM9x30 and later devices support two SAPs concurrently:
  - Primary AP and Guest AP
- The two-device-concurrent operation is achieved by having two virtual interfaces functioning over a single physical interface.
- A single host driver software creates and supports two virtual interfaces.
- The OS IP stack and other applications see the virtual interfaces as individual interfaces.
- Each AP operates with different security modes.
- Multiple virtual devices can operate on the same channel (SCC) or on different channels (MCC) regardless of different radio bands such as 2.4 GHz or 5 GHz.
- If ACS is enabled on the second interface, the second interface is brought up on the same channel as the first interface if the first interface is already up.
- The LTE Coexistence algorithm can override the channel configuration to avoid WLAN interference.

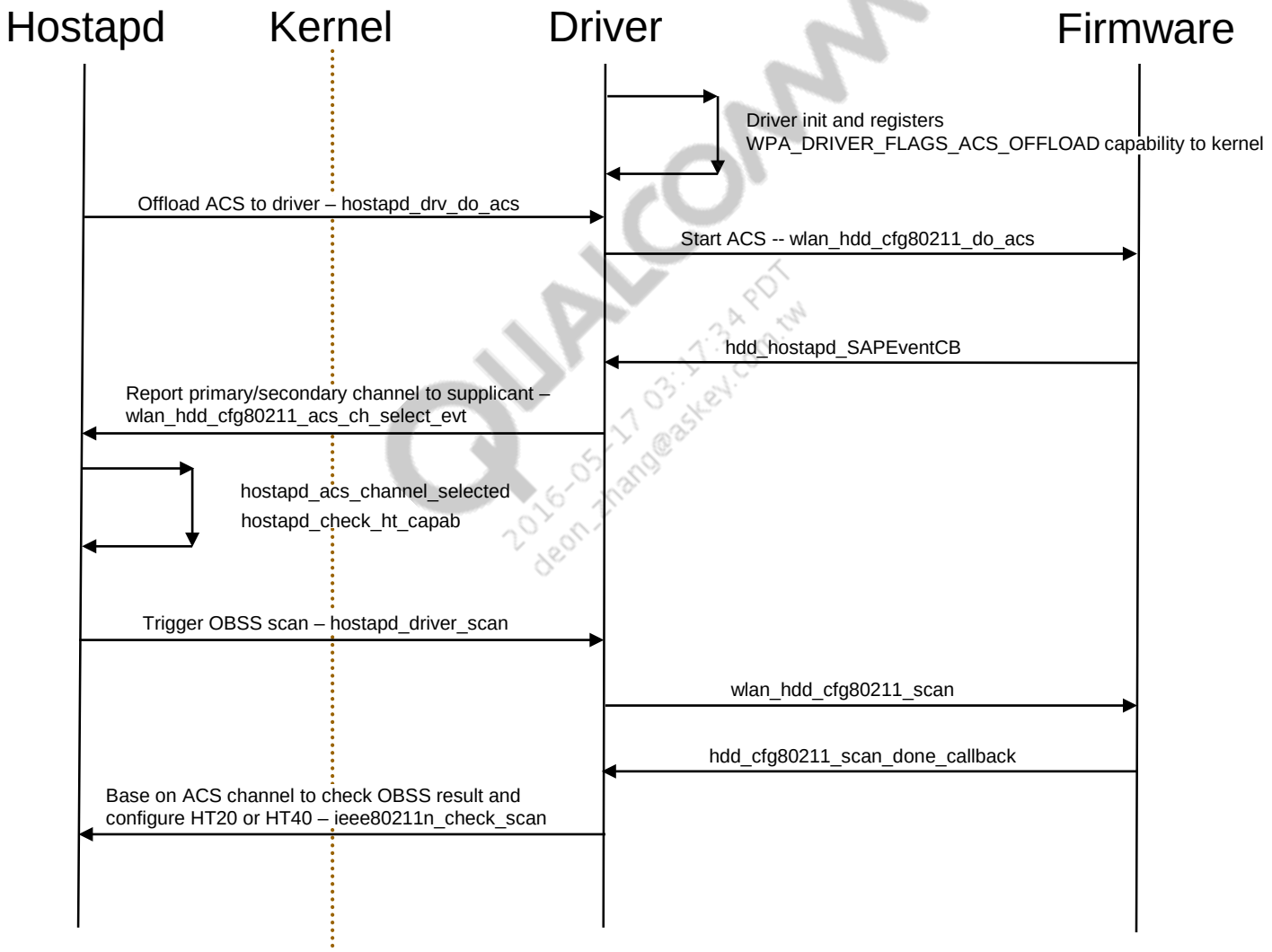
AP+AP lifecycle



ACS and OBSS (1 of 2)

- The ACS feature automatically selects the best channel with the least interference from other APs. See slides 26 and 27 for ACS details.
- The OBSS feature checks any overlapping BSS to decide on HT20 or HT40 mode.
- ACS and OBSS can be enabled by this parameter:
  - set channel = 0 in the *hostapd.conf* file to enable ACS. Set channel = x in the file to disable ACS and assign a channel number.
  - *ht2040_coex_disable* in *hostapd.conf* to disable/enable OBSS scan.
- H*ostapd* offloads the ACS mechanism to the driver.
- The driver notifies *hostapd* about the selected channel.
- *Hostapd* triggers an OBSS scan. Based on the channel number received from the driver for OBSS checking, *hostapd* decides whether to switch to HT20 or HT40 mode.

ACS and OBSS (2 of 2)



Antenna switch for power consumption (1 of 2)

- In some user scenarios, MDM customers might want to dynamically switch the Tx chain from two chains to one chain. This can reduce the power consumption or temperature of the WLAN chip (and vice versa to switch back to the original 2-chain Tx data rate).

- Two features were added after the MDM9x30 LE 2.0 SW branch to support this scenario:
  - Dynamically change Tx chain (under any operating mode)
  - Provide thermal reading in a human-readable format (in °C)

- MDM customers can implement their own thermal throttle algorithms and use these new features to dynamically switch the Tx chain from two chains to one (and vice versa) to lower the power consumption and temperature of the WLAN chip.

- To change the Tx chain, use these commands:

  iwpriv wlan0 set set_txchainmask 1 – Set Tx chainmask to 1x2

  iwpriv wlan0 set set_txchainmask 3 – Set Tx chainmask to 2x2

- To read the WLAN chip temperature use this command:

  iwpriv wlan0 get get_temp

  Note: The reported temperature is from the WLAN chip itself and is not directly related to the external temperature. It should be possible to see the reported temperature value change relative to external temperature, but that is not guaranteed.

Antenna switch for power consumption (2 of 2)

- When changing the Tx chain, the beacon information does not advertise the change of capability. Only the Tx chain used to transmit packets is switching.

- A connected client should not experience any disconnection when the Tx chain is changed.

Note: this feature was NOT applicable for QCA9377(1x1) on MDM9x07

.

Configurable SW Feature Runtime Configuration Support Status

Feature	**Runtime configuration support status** (without restarting QCMobileAP)
AP mode DFS channel support	Not supported
AP automatic shutdown	Not supported
Access control list (ACL)	Supported
Automatic channel selection (ACS)	Not supported
Configurable SCC/MCC support for AP-STA mode	Not supported
Transmit power capping	Supported
WLAN thermal mitigation	Supported
OBSS scan	Not supported
Dynamic antenna switch between 1x2 and 2x2	Supported
AP WLAN mode (b, g, a, 11bn HT20/40, 11an HT20/40, 11ac VHT20/40/80)	Not supported

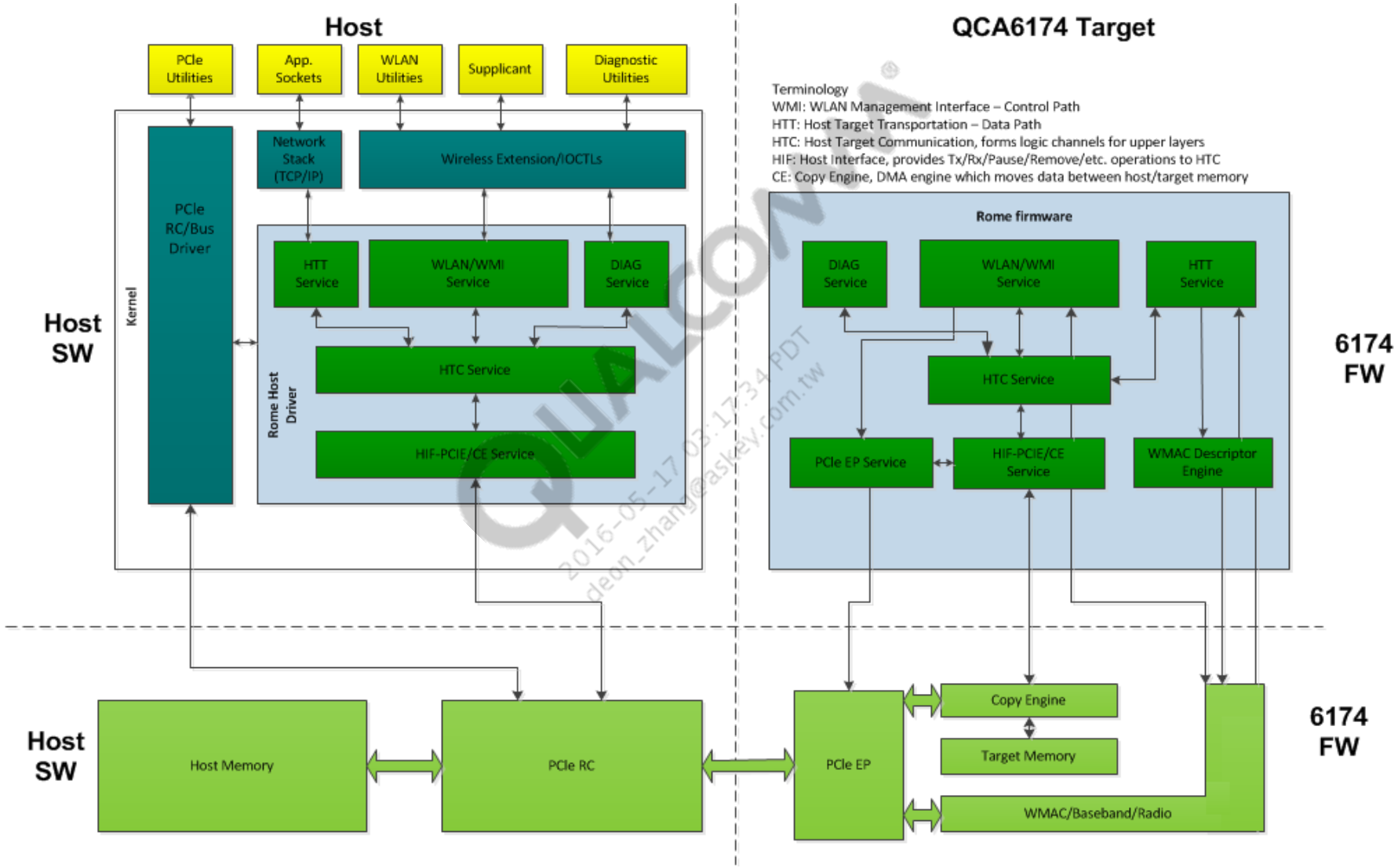
Configurable SW Feature Configuration Index

Feature	Configuration description
AP mode DFS channel support	80-Y7674-41 – “DFS” section
AP automatic shutdown	80-Y7674-41 – “QCMobileAP Automatic Shutdown” section
Access control list (ACL)	80-Y7674-41 – “Dynamic Access Control List” section
Automatic channel selection (ACS)	80-Y7674-41 – “Automatic Channel Selection” section
Configurable SCC/MCC support for AP-STA mode	80-Y7674-41 – “Configurable SCC/MCC support for QCMobileAP-STA mode” section
Transmit power capping	80-Y7674-41 – “Transmit power capping” section
WLAN thermal mitigation	80-Y7674-41 – “Thermal mitigation” section
OBSS scan	80-Y7674-41 – “ACS and OBSS” section
Dynamic antenna switch between 1x2 and 2x2	80-Y7674-41 – “Antenna switch for power consumption” section
AP WLAN mode (b, g, a, 11bn HT20/40, 11an HT20/40, 11ac VHT20/40/80)	80-Y7674-131 – Read the whole document

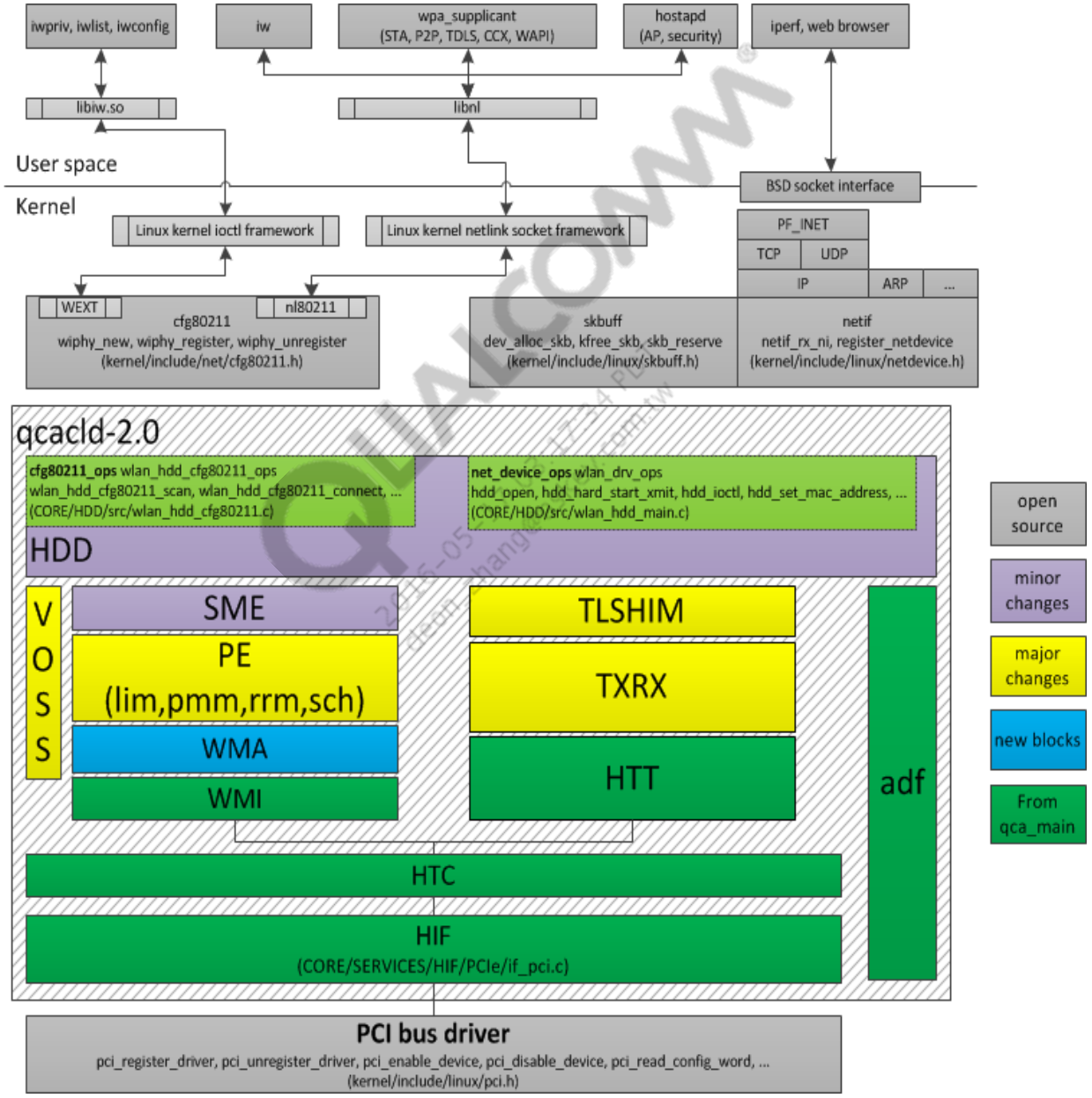
Section 3

QCA61x4(A)/QCA9377 WLAN architecture

QCA61x4(A) hardware/software block diagram

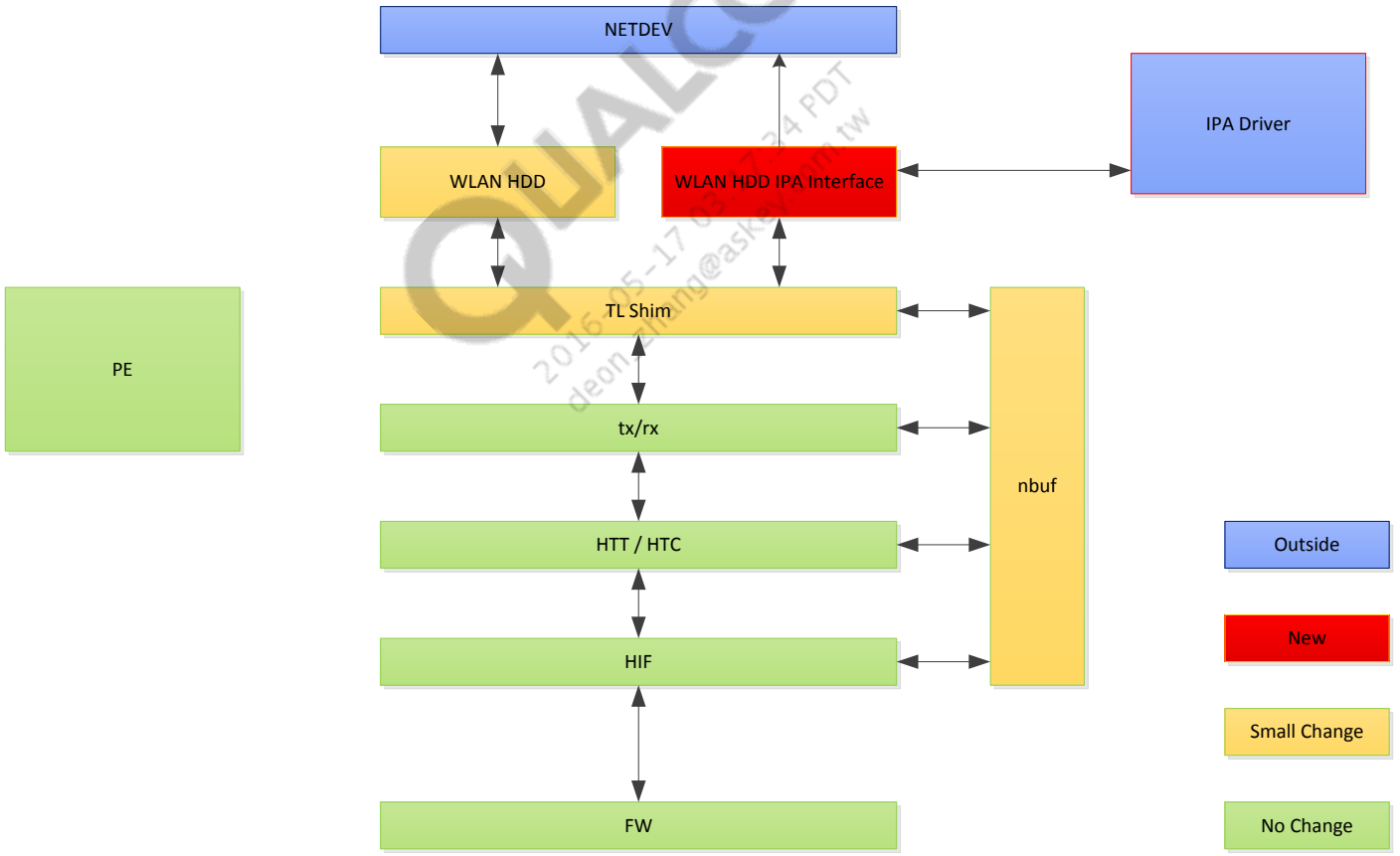


Host driver software block diagram



HDD IPA

- The HDD IPA interface is the module at the HDD layer that creates a communication or datapath between the IPA device driver and the WLAN host driver. The interface performs these functions:
  - Creates a communication interface between the IPA and the WLAN host driver
  - Handles data/communication from IPA to WLAN
  - Handles frames from WLAN lower layer, and decides on the routing path



QCA61x4(A) WLAN driver software modules

Driver source code is located in wlan/qcacld-2.0/CORE

Layer	Abbreviations	Description
ADF	Atheros Driver Framework	OS abstraction for TLSHIM, TXRX, HTT, WMI, HTC, and HIF
HDD	Host Device Driver	Interface with user space for control path (all user configuration comes through this layer) and interface with network stack in datapath. **Northbound**: HDD registers into netdevice and cfg80211 callbacks and implements the associated operations. It also implements the private IOCTLs. **Southbound**: HDD implements control plane operation by using the services of SME and on the data plane by interfacing with the TX_RX module.
HIF	Host Interface	Layer abstract interface difference like USB/SDIO/PCIe.
HTC	Host Target Communication	Mbox for providing host/target flow control, Tx/Rx queue management, and QoS.
HTT	Host Target Transport	Handles host side of Tx/Rx data frame flow to and from target SOC.
PE	Protocol Engine	Implements the core Protocol Engine (that is, MLME.)
SME	Station Management Entry	Control path interface layer between HDD and PE/LIM; exposes APIs to HDD to serve user space requests and interface to the WLAN MAC driver
TLSHIM TXRX	Transport Layer	Provides the abstraction to Tx/Rx data and HDD that helps leverage most of the HDD and entire UMAC from the WCN36x0 driver.
VOSS	Virtual Operating System Services	OS abstraction for HDD, SME, PE, and WMA.
WMA	WMI Adaptation Layer	Provides an adaptation layer to WMI.
WMI	Wireless Module Interface	Control Path Messaging interface between the host and firmware.

cfg80211 callbacks

wlan/qcacld-2.0/CORE/HDD/src/wlan_hdd_cfg80211.c

```
static struct cfg80211_ops wlan_hdd_cfg80211_ops =
{
    .add_virtual_intf = wlan_hdd_add_virtual_intf,
    .del_virtual_intf = wlan_hdd_del_virtual_intf,
    .change_virtual_intf = wlan_hdd_cfg80211_change_iface,
<snip>
    .scan = wlan_hdd_cfg80211_scan,
    .connect = wlan_hdd_cfg80211_connect,
    .disconnect = wlan_hdd_cfg80211_disconnect,
<snip>
    .remain_on_channel = wlan_hdd_cfg80211_remain_on_channel,
    .cancel_remain_on_channel =  wlan_hdd_cfg80211_cancel_remain_on_channel,
<snip>
}
```

net_device_ops callbacks

wlan/qcacld-2.0/CORE/HDD/src/wlan_hdd_main.c

```
static struct net_device_ops wlan_drv_ops = {
    .ndo_open = hdd_open,
    .ndo_stop = hdd_stop,
    .ndo_uninit = hdd_uninit,
    .ndo_start_xmit = hdd_hard_start_xmit,
    .ndo_tx_timeout = hdd_tx_timeout,
    .ndo_get_stats = hdd_stats,
    .ndo_do_ioctl = hdd_ioctl,
    .ndo_set_mac_address = hdd_set_mac_address,
    .ndo_select_queue    = hdd_select_queue,
    .ndo_set_rx_mode = hdd_set_multicast_list,
    .ndo_set_multicast_list = hdd_set_multicast_list,
};
```

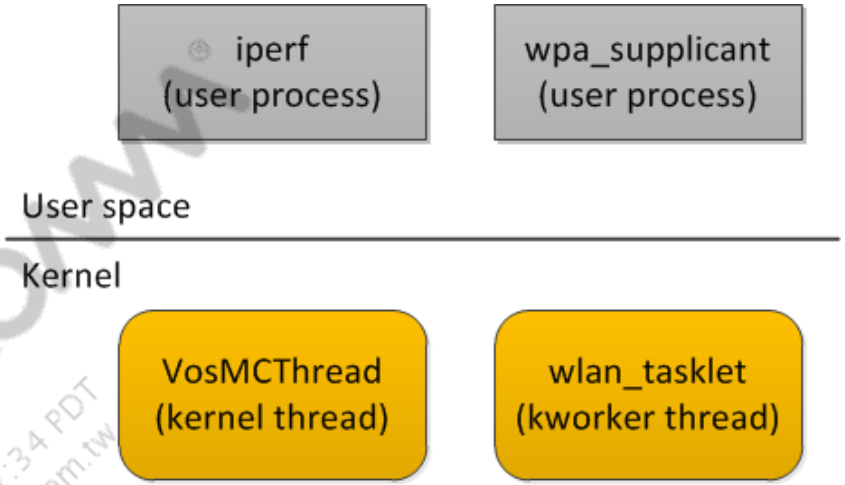
PCI bus driver callbacks

wlan/qcacld-2.0/CORE/SERVICES/HIF/PCIe/if_pci.c

```
static struct pci_device_id hif_pci_id_table[] = {
  { 0x168c, 0x003c, PCI_ANY_ID, PCI_ANY_ID },
  { 0x168c, 0x003e, PCI_ANY_ID, PCI_ANY_ID },
  { 0 }
};
MODULE_DEVICE_TABLE(pci, hif_pci_id_table);
struct pci_driver hif_pci_drv_id = {
  .name       = "hif_pci",
  .id_table   = hif_pci_id_table,
  .probe      = hif_pci_probe,
  .remove     = hif_pci_remove,
  .suspend    = hif_pci_suspend,
  .resume     = hif_pci_resume,
};
```

Thread view

- Tx path runs in network app context (for example, iperf).
- Rx path runs in the HIF Rx tasklet context (wlan_tasklet).
- cfg80211 commands are sent to the driver in wpa_supplicant context .
- VosMCThread − Main control thread for handling the command control path.
- wlan_tasklet − Receives all events and data from the firmware.



Main driver behavior

Initialization

- Rome/Tufello is a discrete solution (compared to the Pronto ISOC solution) so the driver load is independent of the Linux kernel boot.

Control path behavior

- Add/Delete interface, Scan, Connect, Disconnect…

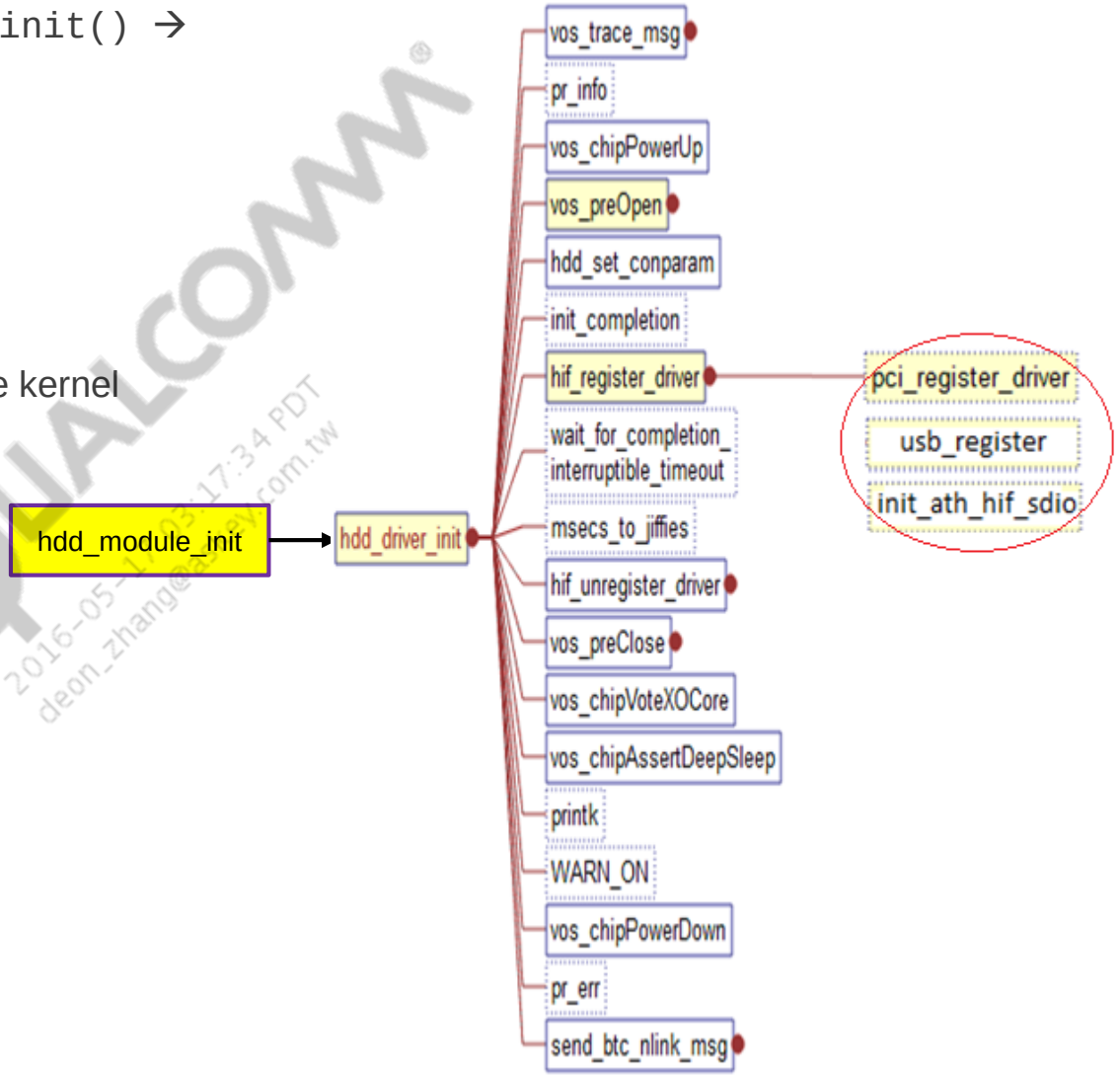
Datapath behavior

- Tx and Rx

Loading driver

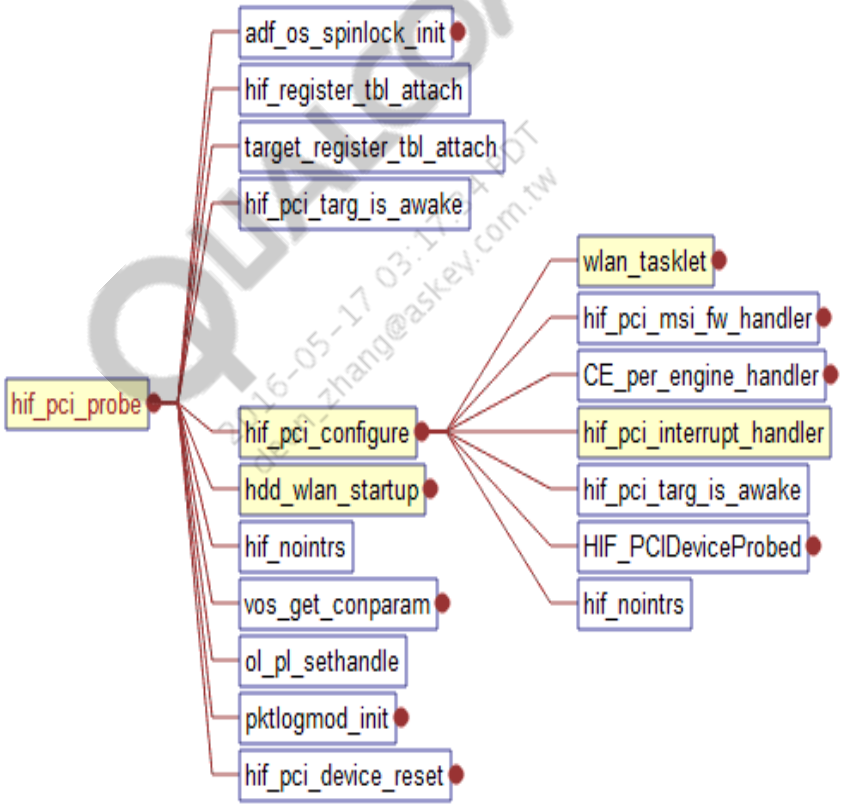
```
hdd_module_init() → hdd_driver_init() →
hif_register_driver() →
pci_register_driver()
int pci_register_driver(struct
pci_driver *drv)
```

This function registers a PCI driver from the kernel



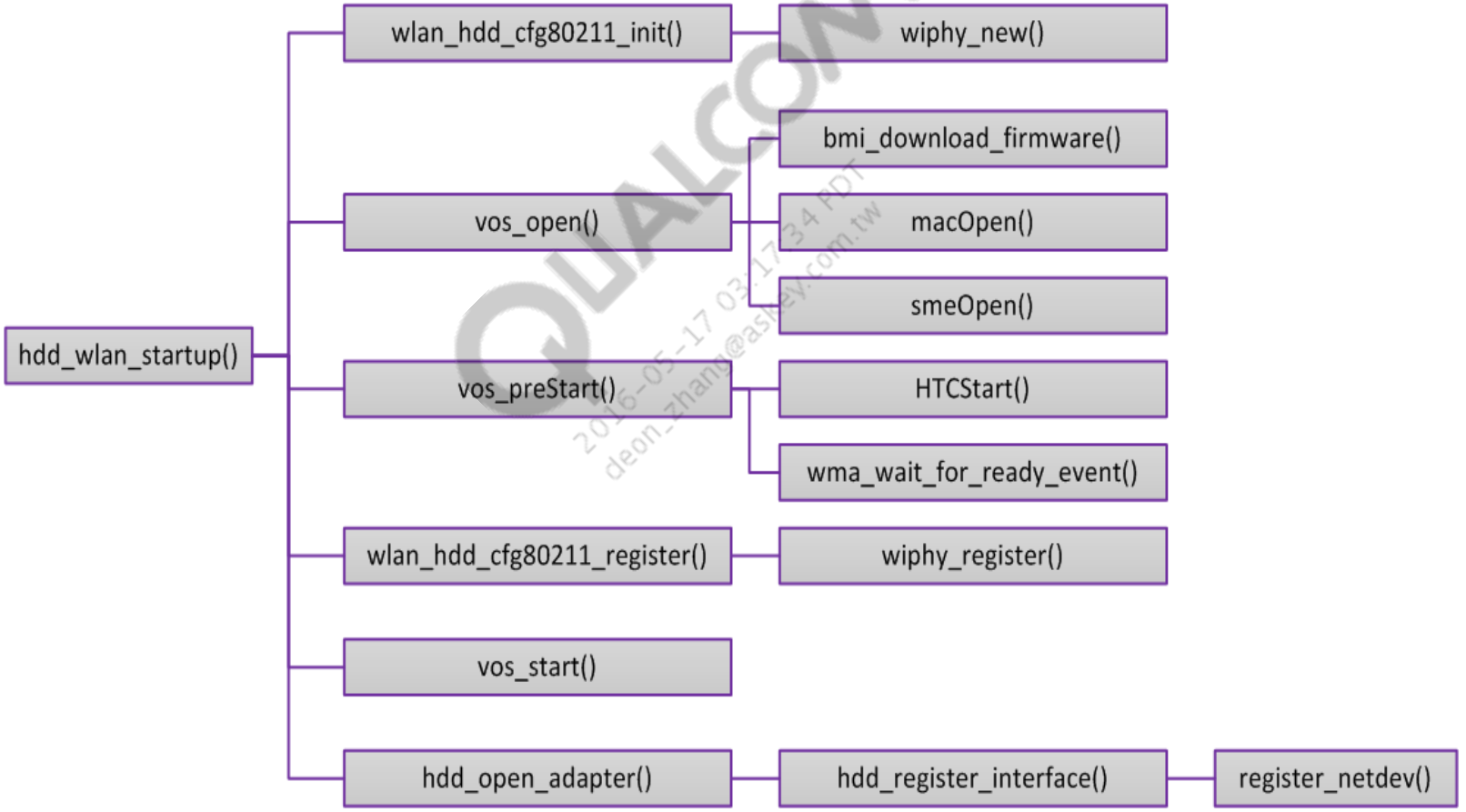
hif_pci_probe()

- If the PCI bus driver detects a PCI device and its vendor ID and device ID match, it calls the *hif_pci_probe()* callback function:
  - hif_pci_probe() calls hdd_wlan_startup()



hdd_wlan_startup()

- Download the firmware and wait until the firmware sends a ready event.
- Call *wiphy_register()* and *register_netdev()*



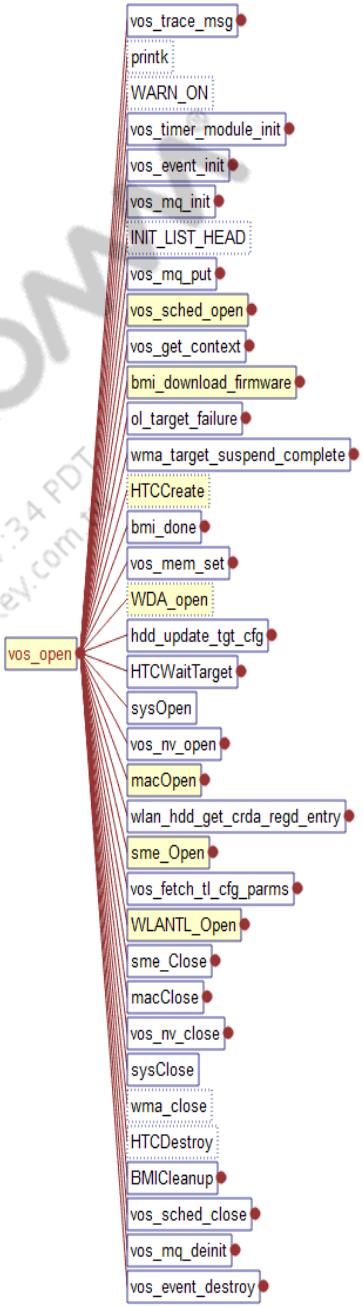
vos_open()

- vos_sched_open()
  - Create main control thread

- bmi_download_firmware()
  1. Get the target_info.
  2. Configure the target.
  3. Download the firmware to target RAM.

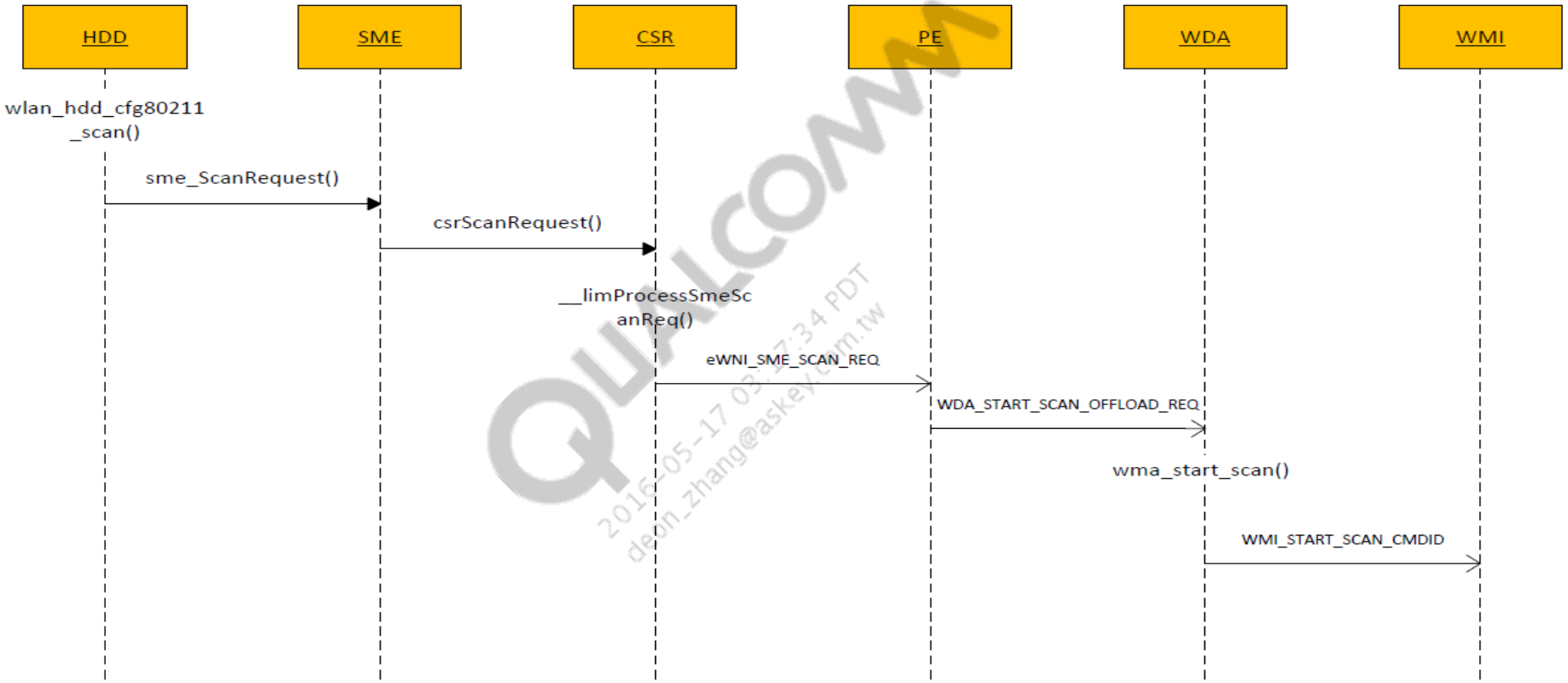
- HTCCreate()
  - Set up HIF layer callbacks.

- WDA_open()
  - Register WMI event handlers:
    - WMI_DEBUG_PRINT_EVENTID
    - WMI_VDEV_START_RESP_EVENTID
    - WMI_VDEV_STOPPED_EVENTID
    - WMI_PEER_STA_KICKOUT_EVENTID
    - WMI_OEM_DATA_RSP_EVENTID
    - WMI_OEM_DATA_ERROR_REPORT_EVENTID



Control path example (1 of 2)

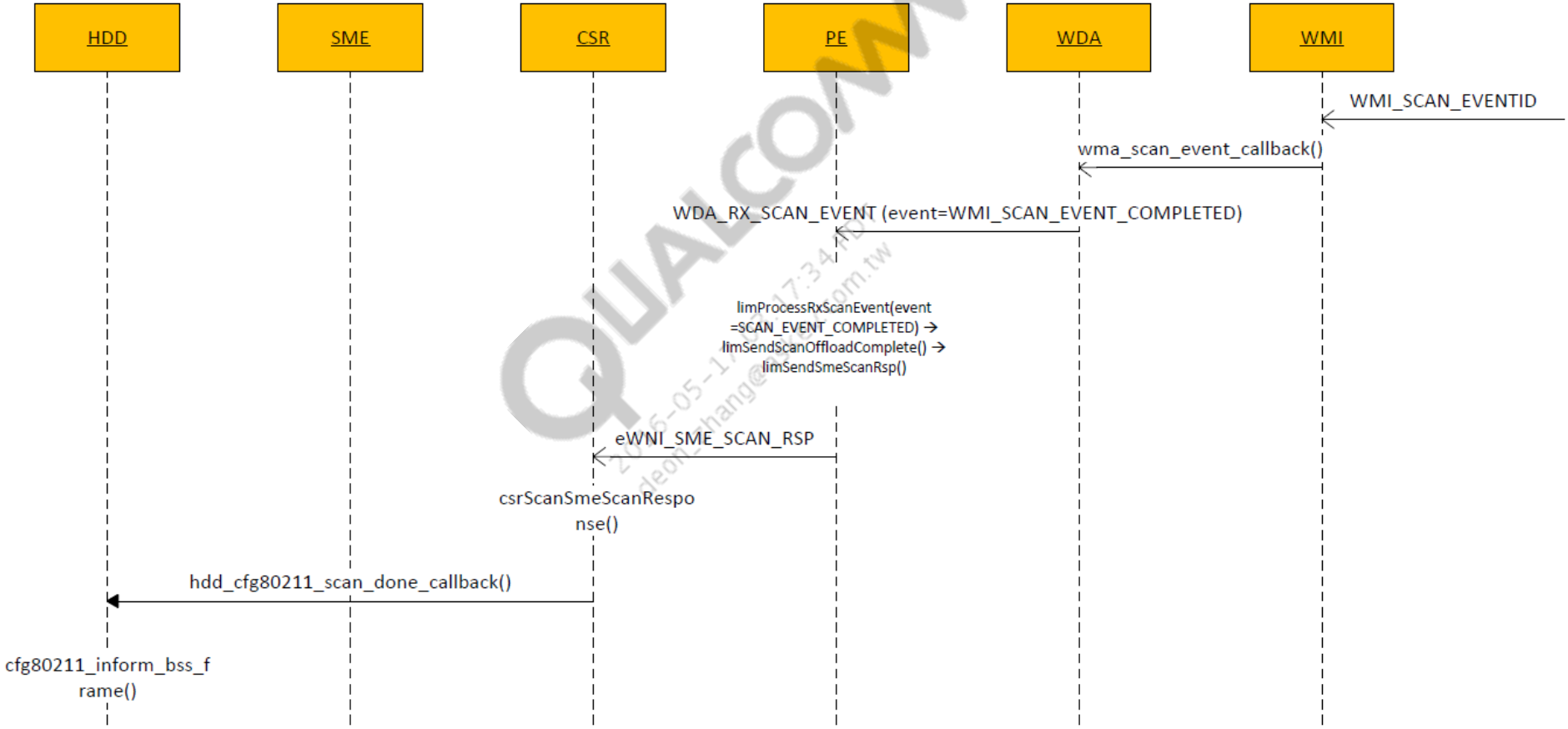
Scan request



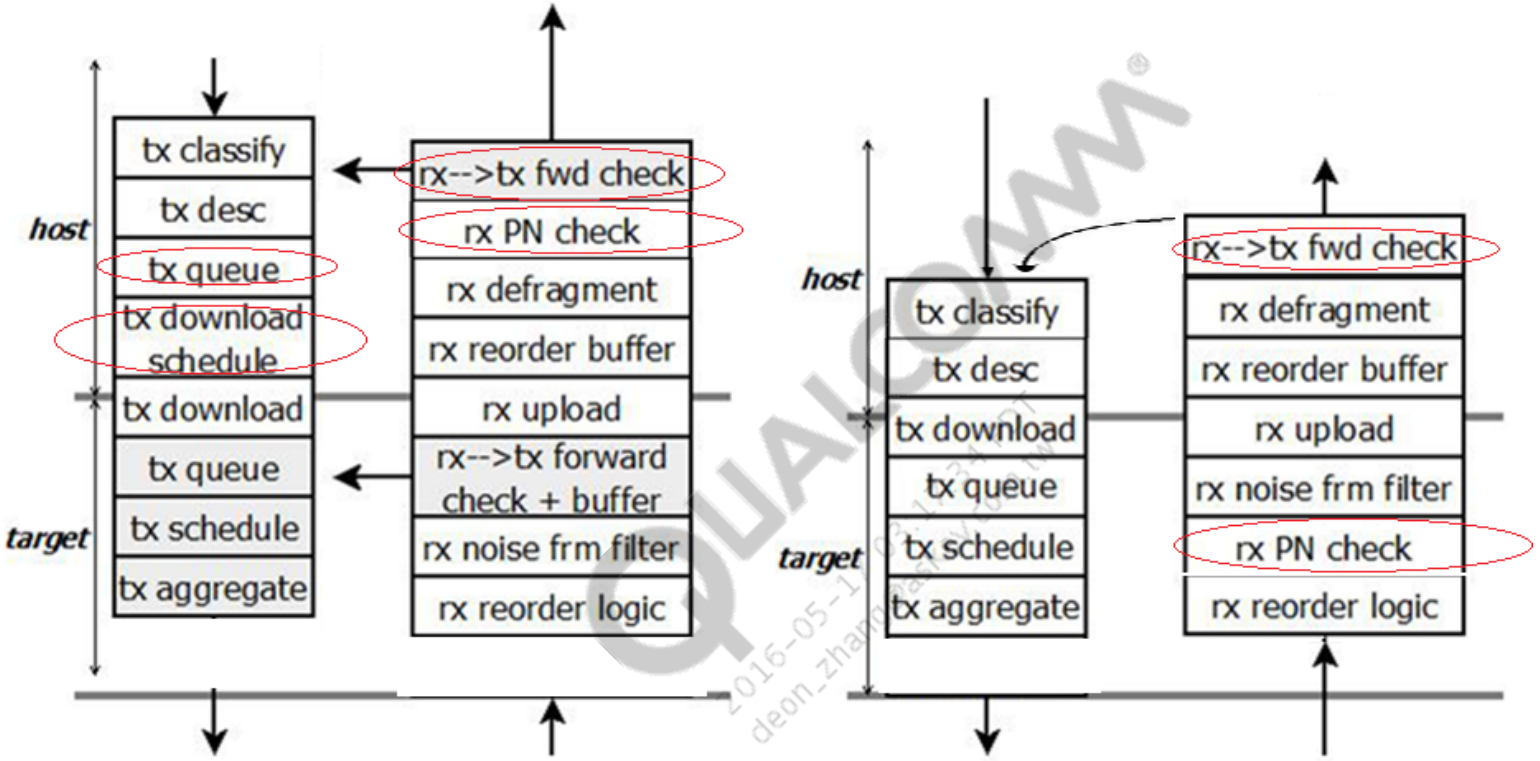
Control path example (2 of 2)

Scan result response.

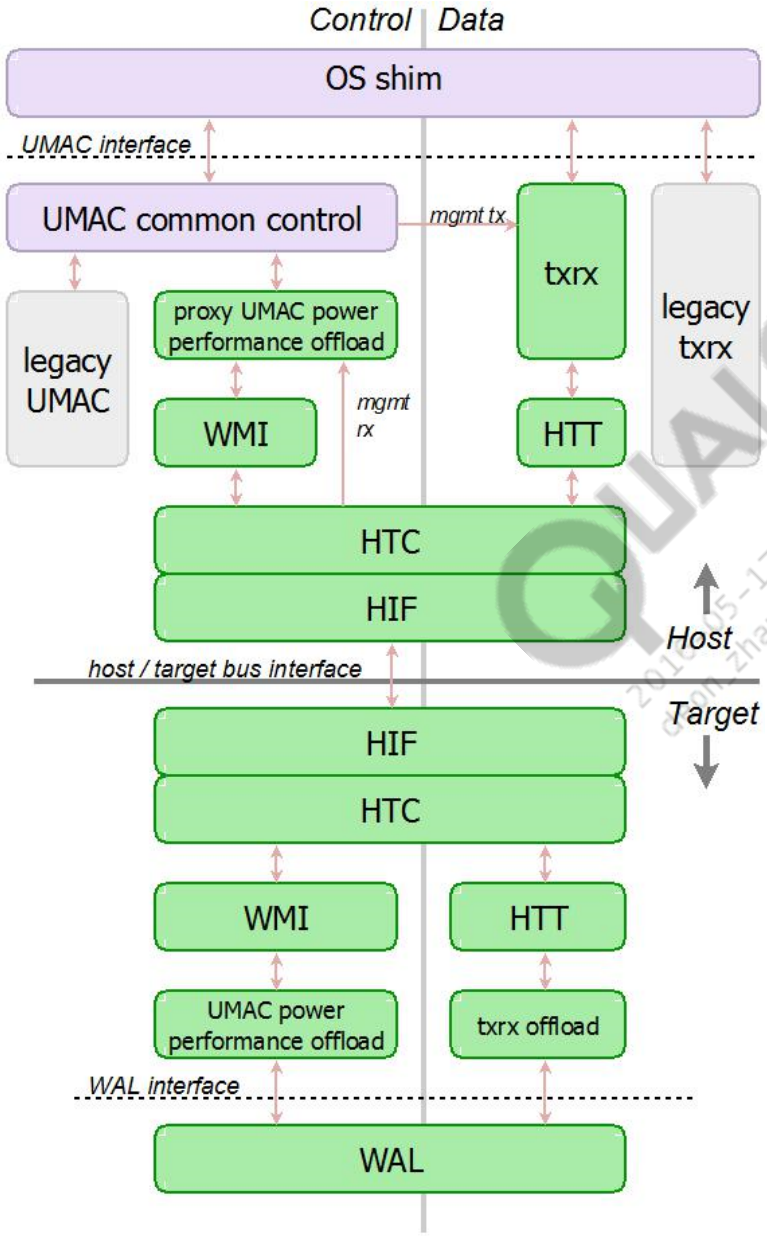
- Handle WMI_SCAN_EVENT_COMPLETED



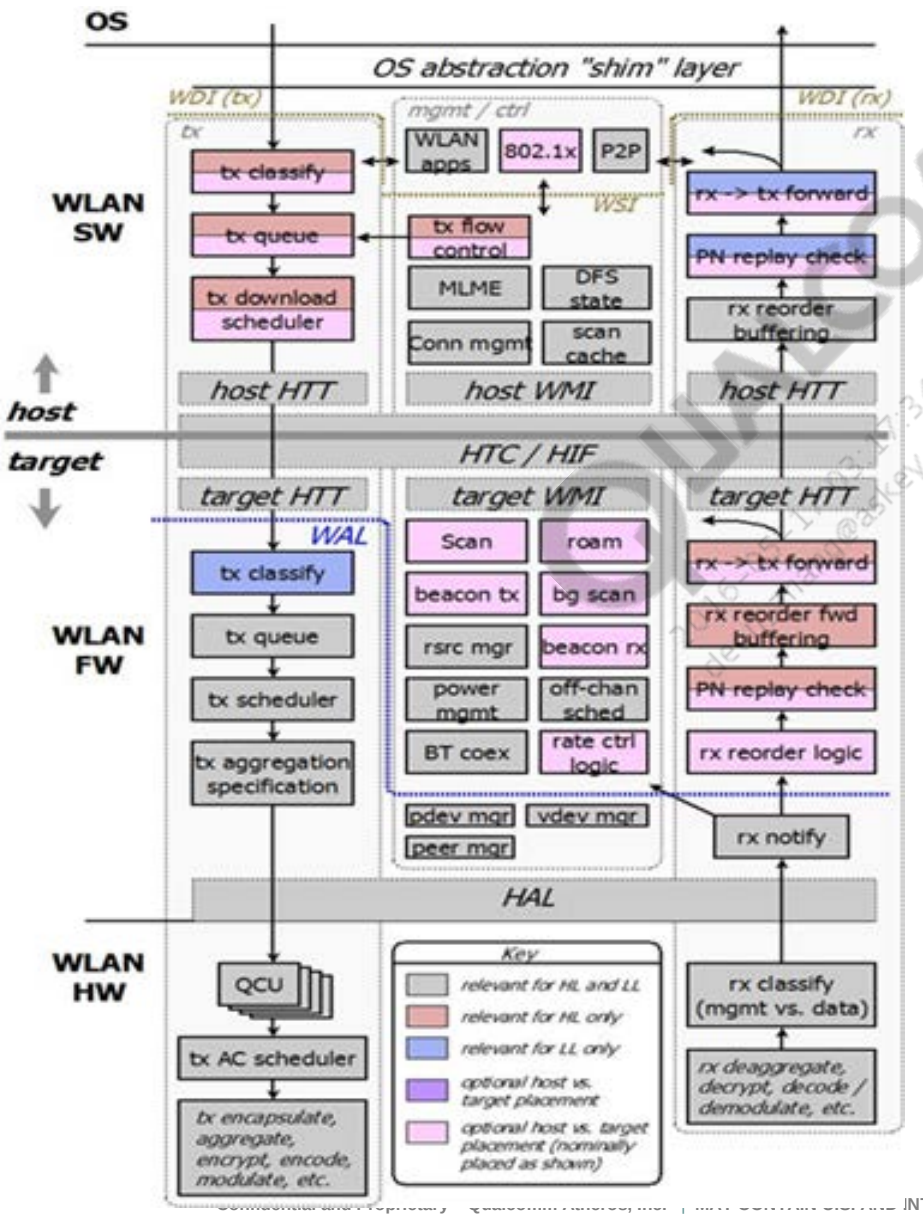
HL datapath vs. LL datapath

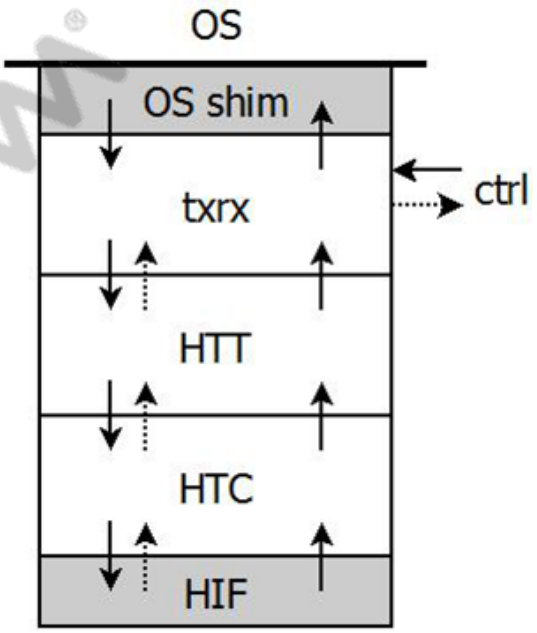


Datapath architecture



Data flow and data layer





Datapath components

OS Shim: Provides abstract interaction with OS

- API: Tx, Rx, monitor
- Data types: adf_nbuf

Tx and Rx: Performs protocol data processing

HTT: Abstraction of host and target data transfer

- Tx: Add Tx metadata needed by the target (HTT Tx desc)
- Rx: descriptor abstraction: MAC DMA Rx ring

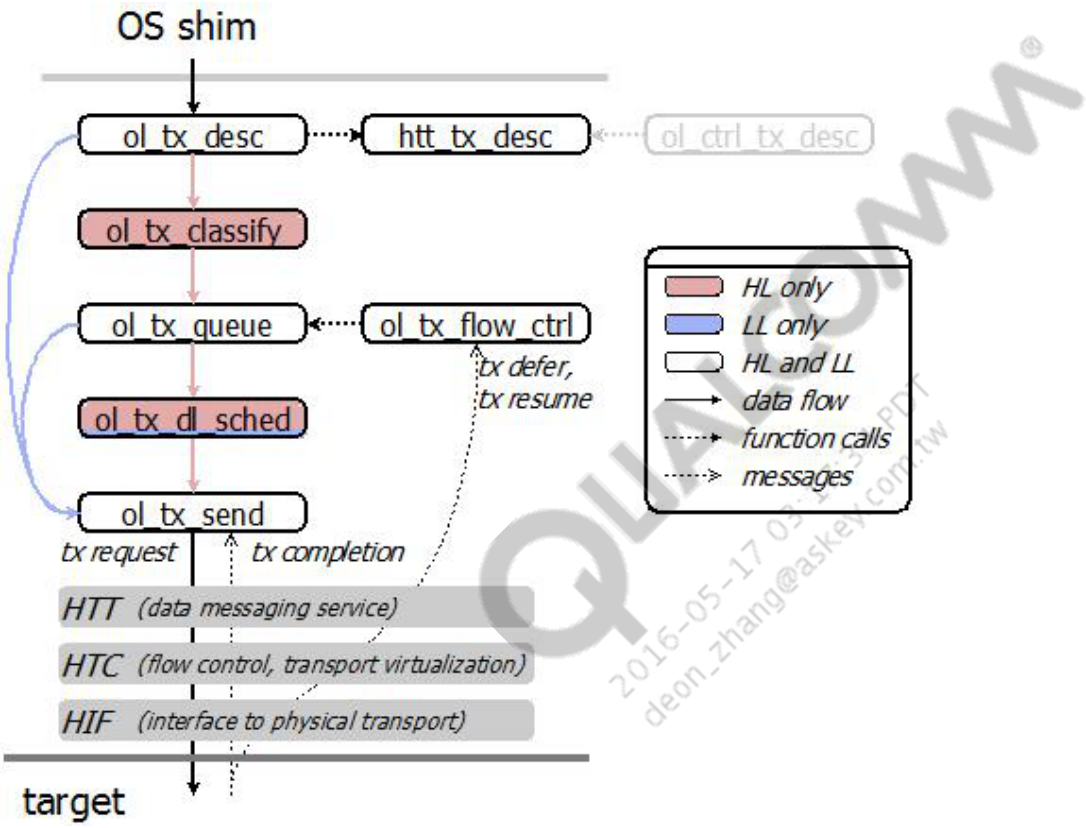
HTC: Virtualization of host and target communication

- Stores frames until the underlying HIF layer can accept them
- Maps physical *pipes* to virtual *endpoints*
- Provides flow control between endpoints

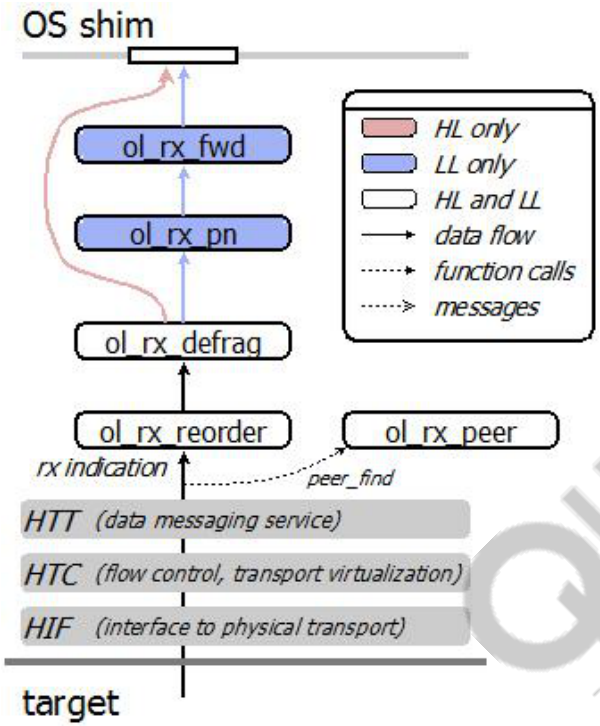
HIF: Abstraction of physical host and target communication

- Purpose: abstraction of physical host and target communication
- Invokes callback when the send is completed and when data is received from the target

Tx datapath flow



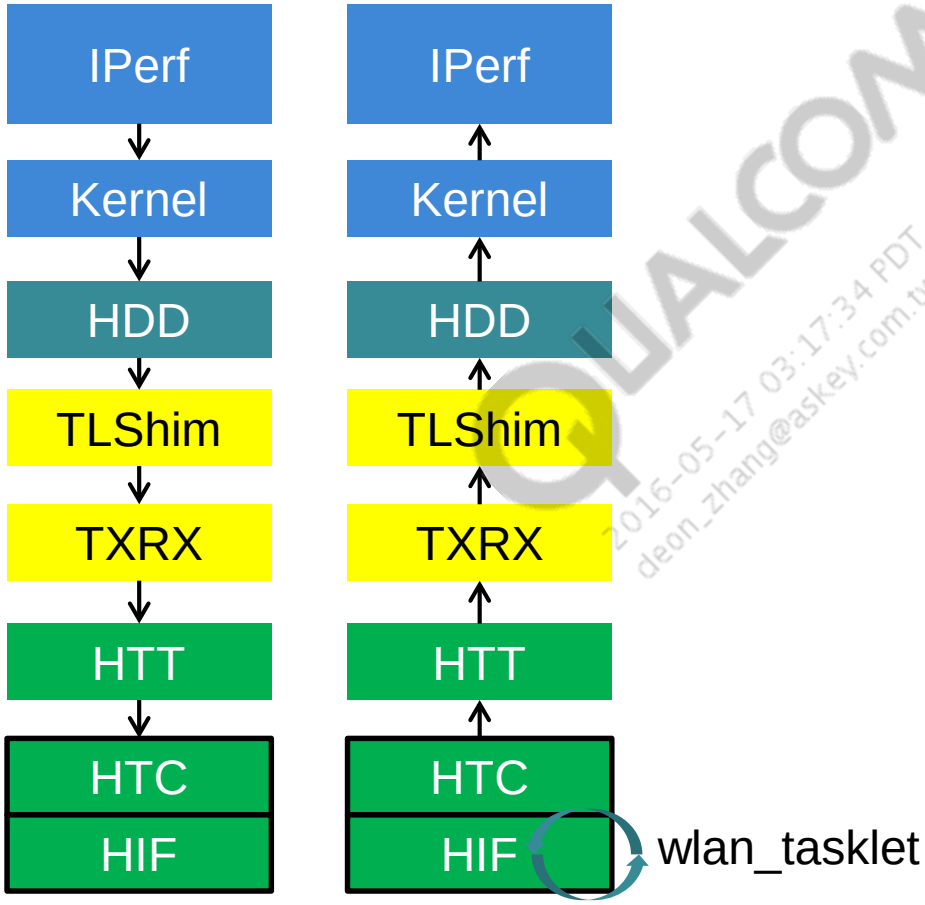
Rx datapath flow



TxRx context

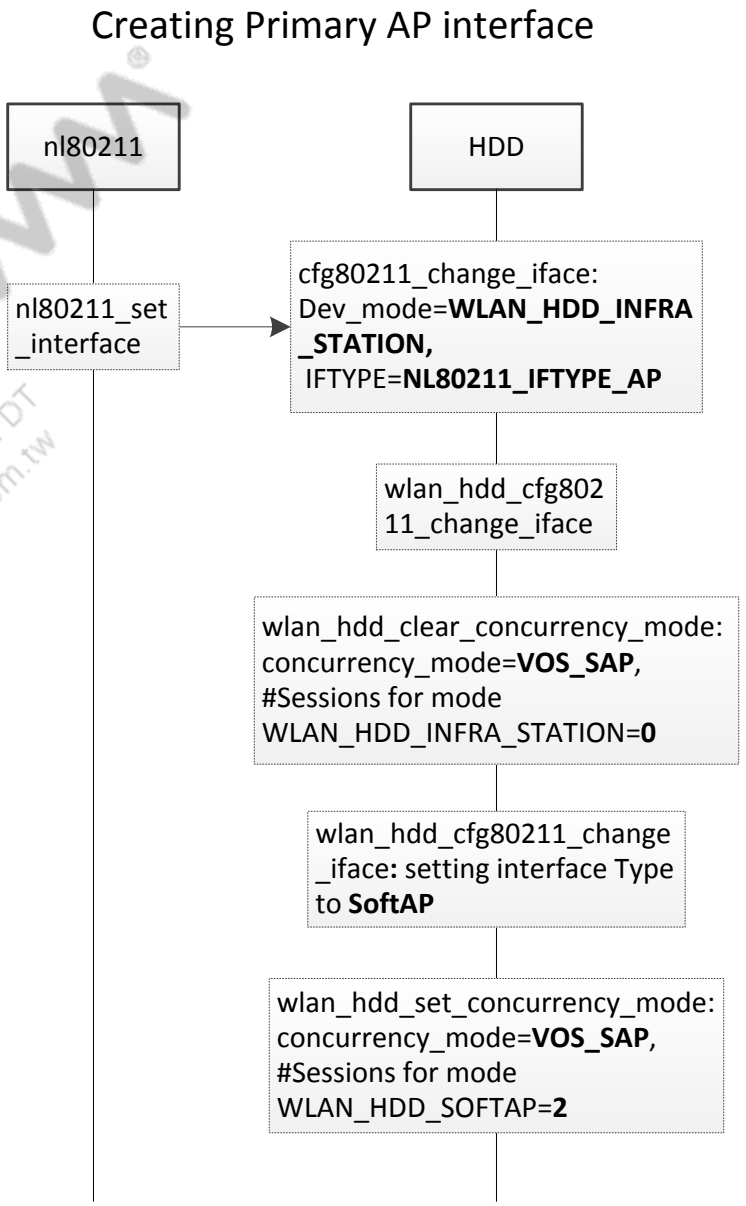
Tx path runs in network app context (for example, iperf).

Rx path runs in the HIF Rx tasklet context (wlan_tasklet).



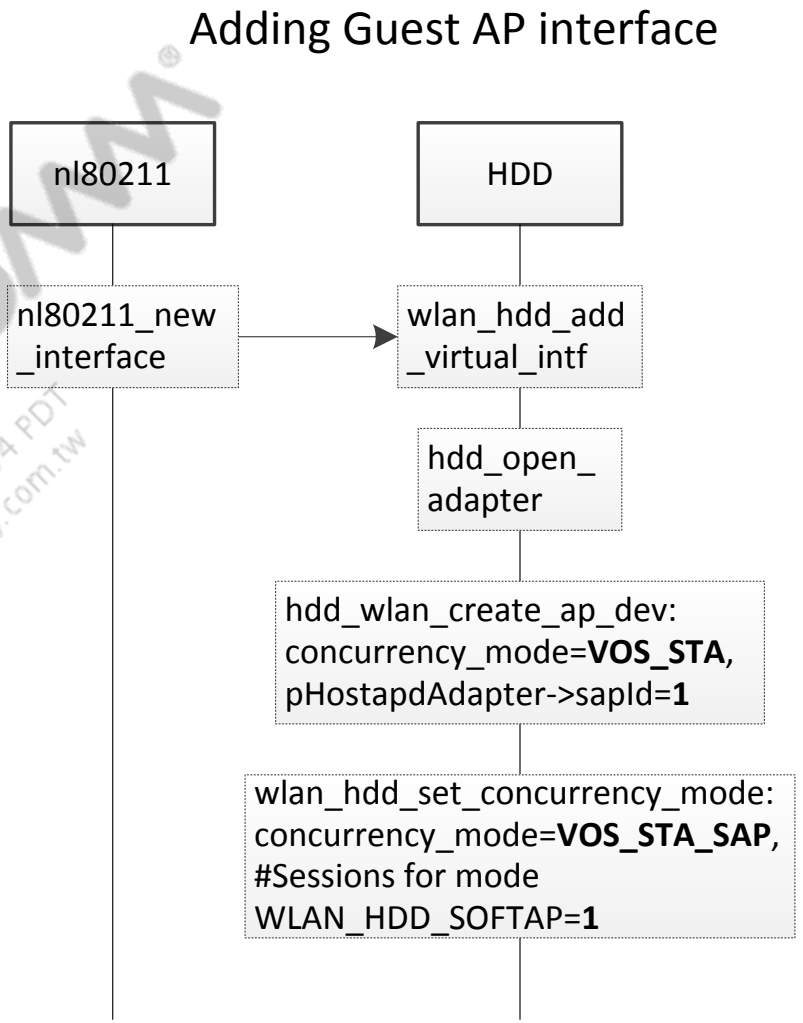
Create the Primary AP interface

- When the WLAN driver starts, it creates an initial interface (wlan0) with INFRA_STATION mode.
- The Primary AP interface is created when hostapd starts for Primary SAP and the initial interface is changed from STA mode to SAP mode.



Add a Guest AP interface

- The Guest AP interface is created when adding an interface (wlan1) with SAP mode as the following command illustrates:
  - $iw wlan0 interface add wlan1 type_ap



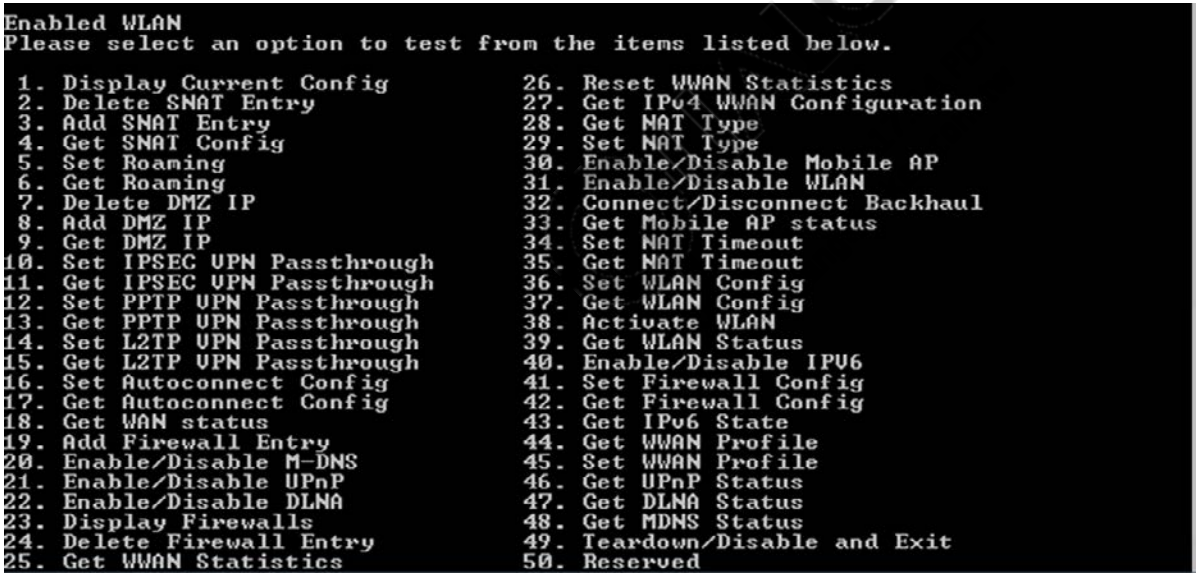
Section 4

QCA61x4(A)/QCA9377 WLAN bringup

Mobile router bringup using QCMAP

You can also bring up the WLAN interface and Single Mode AP using QCMAP (Connection Manager):

1. Power on the device.
2. Type QCMAP_CLI in the root. This command line tool shows router options for the customer to customize/configure mobile router standalone and concurrent functionality.
3. QCMAP options follow. Modify these options based on customer expectations, and this source code becomes part of the 9x30/35/40/45/9x07 meta-releases.



For example: Type 30, followed by 1, and then 31, followed by 1 to bring up the Mobile AP (30) and to bring up the WLAN interface (31). Note that the value 1 enables the devices.

WLAN bringup sequence

1. pci_driver probe

<6>[166.305944] hif_pci_probe

<6>[166.305952] PCI device id is 003e :003e

<6>[166.305958] PCI: enabling device 0000:01:00.0 (0140 -> 0142)

2. Download firmware files

<6>[166.783988] Board extended Data download address: 0x0

<6>[166.799667] ol_download_firmware: Using 0x1234 for the remainder of init

3. WMI_READY_EVENT

<6>[168.052429] wma_rx_service_ready_event-8101: WMA <-- WMI_SERVICE_READY_EVENTID

<6>[168.052473] wma_rx_service_ready_event-8168: WMA --> WMI_INIT_CMDID<6>[168.059508] __wmi_control_rx: WMI UNIFIED READY event

<6>[168.059542] wma_rx_ready_event-8223: WMA <-- WMI_READY_EVENTID

4. Create virtual devices

<6>[168.096291] wma_unified_vdev_create_send-1536: wma_unified_vdev_create_send: ID = 0 VAP Addr = 00:03:7f:2a:05:f3:

<6>[168.096304] TXRX: Created vdev eafe4880 (00:03:7f:2a:05:f3)

<6>[168.096309] wma_vdev_attach-1824: vdev_id 0, txrx_vdev_handle = eafe4880

<6>[168.119575] wma_unified_vdev_create_send-1536: wma_unified_vdev_create_send: ID = 1 VAP Addr = 12:03:7f:2a:05:f3:

<6>[168.119598] TXRX: Created vdev eb3c2880 (12:03:7f:2a:05:f3)

<6>[168.119632] wma_vdev_attach-1824: vdev_id 1, txrx_vdev_handle = eb3c2880

Compile WLAN driver

1. Import environment
   - $ cd apps_proc/oe-core
   - $ source build/conf/set_bb_env.sh

2. Clean WLAN objects
   - $ bitbake -fc clean qcacld-ll (for PCIe interface of QCA6174)
   - $ bitbake -fc clean qcacld-hl (for SDIO interface of QCA6174/9377)

3. Compile WLAN driver
   - $ bitbake -fc compile qcacld-ll (for PCIe interface of QCA6174)
   - $ bitbake -fc compile qcacld-hl (for SDIO interface of QCA6174/9377)

4. Install WLAN driver
   - $ bitbake -fc install qcacld-ll (for PCIe interface of QCA6174)
   - $ bitbake -fc install qcacld-hl (for SDIO interface of QCA6174/9377)

wlan.ko driver module is created in
apps_proc/oe-core/build/tmp-eglibc/work/mdm9635-oe-linux-gnueabi/qcacld-ll-git r0/image/lib/modules/3.10.0+/extra

WLAN driver binary files

- WLAN driver module
  - /lib/modules/3.10.0+/extra/wlan.ko

- Configuration files are located in /lib/firmware/wlan
  - qcom_cfg.dat: Default configuration parameters
  - qcom_cfg.ini: OEM configurable parameters to override settings in qcom_cfg.dat
  - qcom_wlan_nv.bin: Reserved

WLAN firmware files

- Released in cnss_proc folder as binary
  - Downloaded from Qualcomm ChipCode

- Binary files are symbolically linked to the /lib/firmware folder in the MDM platform
  - athwlan.bin: QCA61x4(A) firmware
  - fakeboar.bin, otp.bin and utf.bin: Calibration files

Turn on the WLAN hotspot

1. Load the WLAN driver module: # */etc/init.d/wlan start*
2. Run hostapd: # *hostapd -dddd /etc/hostapd.conf*
3. Configure the IP address: # *ifconfig wlan0 192.168.1.1 netmask 255.255.255.0 up*
4. Configure the client with a static IP and connect it to the WLAN hotspot. The default SSID is *QSoftAP*.

Turn on WLAN FTM

1. Install QPST and QRCT on the PC.
2. Boot up the device.
3. Launch WLAN FTM mode with # */etc/init.d/wlan start_ftm*
4. Launch ftmdaemon with # *ftmdamon -n -dd*
5. Copy the board data to *C:\Qualcomm\WCN\ProdTests\refDesigns\boardData*
6. Make sure the board datapath in the following XML file is correct
   - *C:\Program Files (x86)\Qualcomm\QDART\bin\WLAN_DUT_Config_QC6174.xml*
7. Launch QPST and make sure there is a COM port for the MDM device.
8. Launch QRCT.
9. From the QRCT *Target* list, select **MSM/MDM**
10. From the QRCT *COM Port* list, select the correct COM port.
11. From the QRCT *Select Chipset* list, select **QC6174**
12. From the QRCT *Select setup* option, select the correct XML file. By default, that file is *WLAN_DUT_Config_QC6174.xml*
13. Press **Load DUT**.
    A. Board data is loaded from C:\Qualcomm\WCN\ProdTests\refDesigns\boardData\
    B. You see this debug message in the QRCT debug field: *QLIB_FTM_WLAN_Atheros_LoadDUT(qc6174,C:\QUALCOMM\WCN\ProdTests\refDesigns\BoardData\fakeboar.bin,5,62)*
14. Begin RF testing.

WLAN debug log (1 of 3)

- The two WLAN log types are:
  - WLAN driver-side log
  - WLAN firmware-side log
- These tools are useful on the host side:
  - *iwpriv*, issues private commands to the WLAN driver
  - *iw* is an open source standard application to scan, connect, or get information
  - *lspci*, checks the PCI enumeration
- Find the WLAN version:
  - Issue this private command: *iwpriv wlan0 version*
  - Read the results in the kernel log: *Host SW:<driver ver.>; FW:<FW ver.>; HW:<HW ver.>*
- Configure the WLAN driver-side log level in separate WLAN driver modules with these *WCNSS_qcom_cfg.ini* parameters:
  - *vosTraceEnableHDD*
  - *vosTraceEnableWMA*
  - *vosTraceEnableWDA*
  - *vosTraceEnableTL*
  - *vosTraceEnableSME*
  - *vosTraceEnablePE*
  - *vosTraceEnableSYS*
  - *vosTraceEnableVOSS*
  - *vosTraceEnableSAP*
  - *vosTraceEnableHDDSAP*

WLAN debug log (2 of 3)

- Configure the WLAN driver-side log level like this:
  - VOS_TRACE_LEVEL_NONE = 0 (*No* traces will be enabled)
  - VOS_TRACE_LEVEL_FATAL – Bit 0 (default)
  - VOS_TRACE_LEVEL_ERROR – Bit 1
  - VOS_TRACE_LEVEL_WARN – Bit 2
  - VOS_TRACE_LEVEL_INFO – Bit 3
  - VOS_TRACE_LEVEL_INFO_HIGH – Bit 4
  - VOS_TRACE_LEVEL_INFO_MED – Bit 5
  - VOS_TRACE_LEVEL_INFO_LOW – Bit 6
  - VOS_TRACE_LEVEL_DEBUG – Bit 7

- *Enable FATAL, ERROR, AND WARNING for HDD*, example:
  - vosTraceEnableHDD=7

- The WLAN firmware-side log can be enabled and forwarded to the kernel log with these settings:
  - iwpriv wlan0 dl_type 0 (type = 0 to log to kernel dmesg)
  - iwpriv wlan0 dl_loglevel 0

- The log level can be as follows:
  - DBGLOG_VERBOSE = 0
  - DBGLOG_INFO
  - DBGLOG_INFO_LVL_1
  - DBGLOG_INFO_LVL_2
  - DBGLOG_WARN
  - DBGLOG_ERR

WLAN debug log (3 of 3)

- The log message is output with *loglevel* ≥ *set level*

- The WLAN firmware-side log can be enabled and forwarded to diag with the following settings; QXDM on a PC/NB can show the firmware log messages:
  - *iwpriv wlan0 dl_report 1*
  - *iwpriv wlan0 dl_type 3*
  - *iwpriv wlan0 dl_loglevel 0*
  - *cld_fwlog_netlink -q* (the *q* parameter forwards the WLAN firmware log to the QXDM tool)
  - Launch QXDM on a PC/NB and enable the WLAN option in *Message View Config*

- WLAN firmware log messages show in the kernel log (examples):
  - <6>[116.653958] FWLOG: [76343] WAL_DBGID_WHAL_CHANNEL_CHANGE_COMPLETE (0x0, 0x4540e3c)
  - <6>[116.653967] FWLOG: [76343] WAL_DBGID_WAL_CHANNEL_CHANGE_COMPLETE (0x98f, 0x4540e72)
  - <6>[116.653973] FWLOG: [76343] RESMGR_CHMGR_CHANNEL_CHANGE (0x98f)
  - <6>[116.653987] FWLOG: [76343] RESMGR_OCS_CHANNEL_SWITCHED (0x0)

- The WLAN firmware log mechanism is being improved and is intended to work like this:
  - The WLAN firmware log is forwarded to the host side without additional commands.
  - The user launches QXDM on a PC/NB and enables the WLAN option in *Message View Config*.
  - The user does not need to issue an *iwpriv* command or launch *cld_fwlog_netlink*.
  - The log mechanism is more user-friendly than the current version.

Thank You