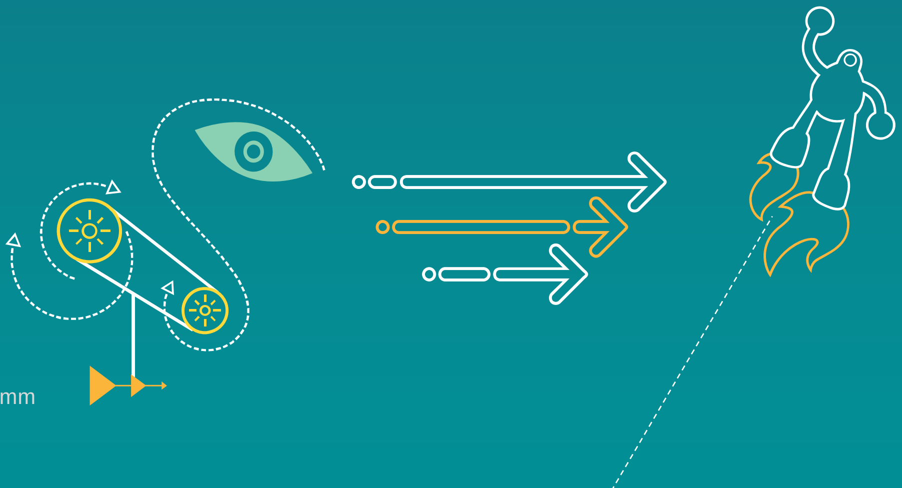
NOTICE REGARDING QUALCOMM ATHEROS, INC.

Effective June 2016, Qualcomm Atheros, Inc. (QCA) transferred certain of its assets, including substantially all of its products and services, to its parent corporation, Qualcomm Technologies, Inc. Qualcomm Technologies, Inc. is a wholly-owned subsidiary of Qualcomm Incorporated. Accordingly, references in this document to Qualcomm Atheros, Inc., Qualcomm Atheros, Atheros, QCA or similar references, should properly reference, and shall be read to reference, Qualcomm Technologies, Inc.

CNSS.SW_RM Wi-Fi Software Architecture Overview

Qualcomm Atheros, Inc.

80-Y7674-2 Rev. E



Confidential and Proprietary – Qualcomm Atheros, Inc.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm or its subsidiaries without the express approval of Qualcomm's Configuration Management.

Confidential and Proprietary – Qualcomm Atheros, Inc.

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

Revision History

Revision	Date	Description
A	Nov 2013	Initial release
B	Nov 2013	Removed slides 68 and 69 related to internal links
C	Jan 2014	Title change; added Firmware, Bluetooth Coexistence, and Advanced Features; generalized the document for QCA6174 software, instead of host driver-specific
D	March 2014	Update of Presentation: Wi-Fi Software Architecture Overview
E	April 2014	Changed presentation title

Agenda

- QCA6174 Wi-Fi Architecture Overview 6
  - QCA6174 Wi-Fi Features (1 of 2) 7
  - QCA6174 Wi-Fi Features (2 of 2) 8
  - Baseline Architecture Overview 9
  - Host HW/SW Block Diagram 10
- Driver 11
  - Driver 12
  - QCA6174 LA Driver 13
  - CLD 2.0 SW Threading Model: Discrete Solution (QCA6174) 14
  - SW Delta between Pronto and QCA6174: CLD 1.0 Host Driver and Execution Model 15
  - Software Delta between Pronto and QCA6174 16
  - Software Delta between Pronto and QCA6174: Components and Execution Model 17
  - Software Delta between Pronto and QCA6174: Firmware Loading and Host Driver init 18
  - Initialization for QCA6174 19
  - Initialization Sequence: insmod() to Load LA Host Driver 20
  - PCI bus driver callbacks 21
  - Host Driver Initialization Sequence for QCA6174: hif_pci_prob() 22
  - Host Driver Initialization Sequence for QCA6174: hdd_wlan_startup() 23
  - Host Driver Initialization Sequence for QCA6174: vos_open() 24
  - Host Driver Initialization Sequence for QCA6174: vos_preStart() 25
  - Host Driver Initialization Sequence for QCA6174: vos_start() 26
  - WLAN Bringup Sequence Log 27
  - SoftMAC 28
  - Control Path for QCA6174 29
  - Control Path for QCA6174: Add Virtual Interface 30
  - Control Path for QCA6174: Delete Virtual Interface 31
  - Control Path for QCA6174: Scan Request 32
  - Control Path for QCA6174: Scan Request Log 33
  - Control Path for QCA6174: Scan Event Log 34
  - Control Path for QCA6174: Connect 35
  - Control Path for QCA6174: Disconnect (Host-Initiated) 36
  - Control Path for QCA6174: Disconnect (AP-Initiated) 37
  - Subsystem Recovery: Overview 38
  - Data Path Flows for QCA6174 (Architecture) (1 of 2) 39
  - Data Path Flows for QCA6174 (Architecture) (2 of 2) 40
  - Data Path Flows for QCA6174 (OS Shim) 41
  - Data Path Flows for QCA6174 (HTT) 42
  - Data Path Flows for QCA6174 (HTC) 43
  - Data Path Flows for QCA6174 (HIF) 44
  - TX RX Context 45
  - Background: Data Path Offloads in QCA6174 PCIe System 46
- Firmware 47
  - QCA6174 Firmware Architecture 48
  - Three Layers 49
  - App Layer 50
  - OLF 51
  - OLF (Offload Stack Functions) 52
  - Key OLF Modules 53
  - OLF Modules (1 of 3) 54
  - OLF Modules (2 of 3) 55
  - OLF Modules (3 of 3) 56
  - WAL Services 57
  - WAL Objects 58
- PCIe 59
  - Host/QCA6174 PCIe Inter-Connect 60
  - Host/ QCA6174 System Low Power Mode 61
  - D0-L1SS Entry 62
  - D0-L1SS Exit – Host-Initiated 63
  - D0-L1SS Exit – Device-Initiated 64
  - D3hot Entry 65
  - D3hot Exit – Host-Initiated 66
  - D3hot Exit – Device-Initiated 67
  - D3cold Entry 68
  - D3cold Exit – Host-Initiated 69
  - D3cold Exit – Device-Initiated 70
  - QCA6174 PCIe EP Service Low Power State 71
- Advanced Features 72
  - Green Tx Feature (1 of 2) 73
  - Green Tx Feature (2 of 2) 74
  - RTT/Ranging Engine on QCA6174 75
  - Multi-Channel Concurrency (1 of 7) 76
  - Multi-Channel Concurrency (2 of 7) 77
  - Multi-Channel Concurrency (3 of 7) 78
  - Multi-Channel Concurrency (4 of 7) 79
  - Multi-Channel Concurrency (5 of 7) 80
  - Multi-Channel Concurrency (6 of 7) 81
  - Multi-Channel Concurrency (7 of 7) 82
  - MU-MIMO 83
  - WoW 84
  - DFS (1 of 2) 85
  - DFS (2 of 2) 86
  - Coexistence Architecture on QCA6174 87
  - What are the Benefits of MCI over 3-wire (PTA)? 88
  - New Use Cases with MCI 89
  - MCI Functionality Details (1 of 3) 90
  - MCI Functionality Details (2 of 3) 91
  - MCI Functionality Details (3 of 3) 92

QCA6174 Wi-Fi Architecture Overview

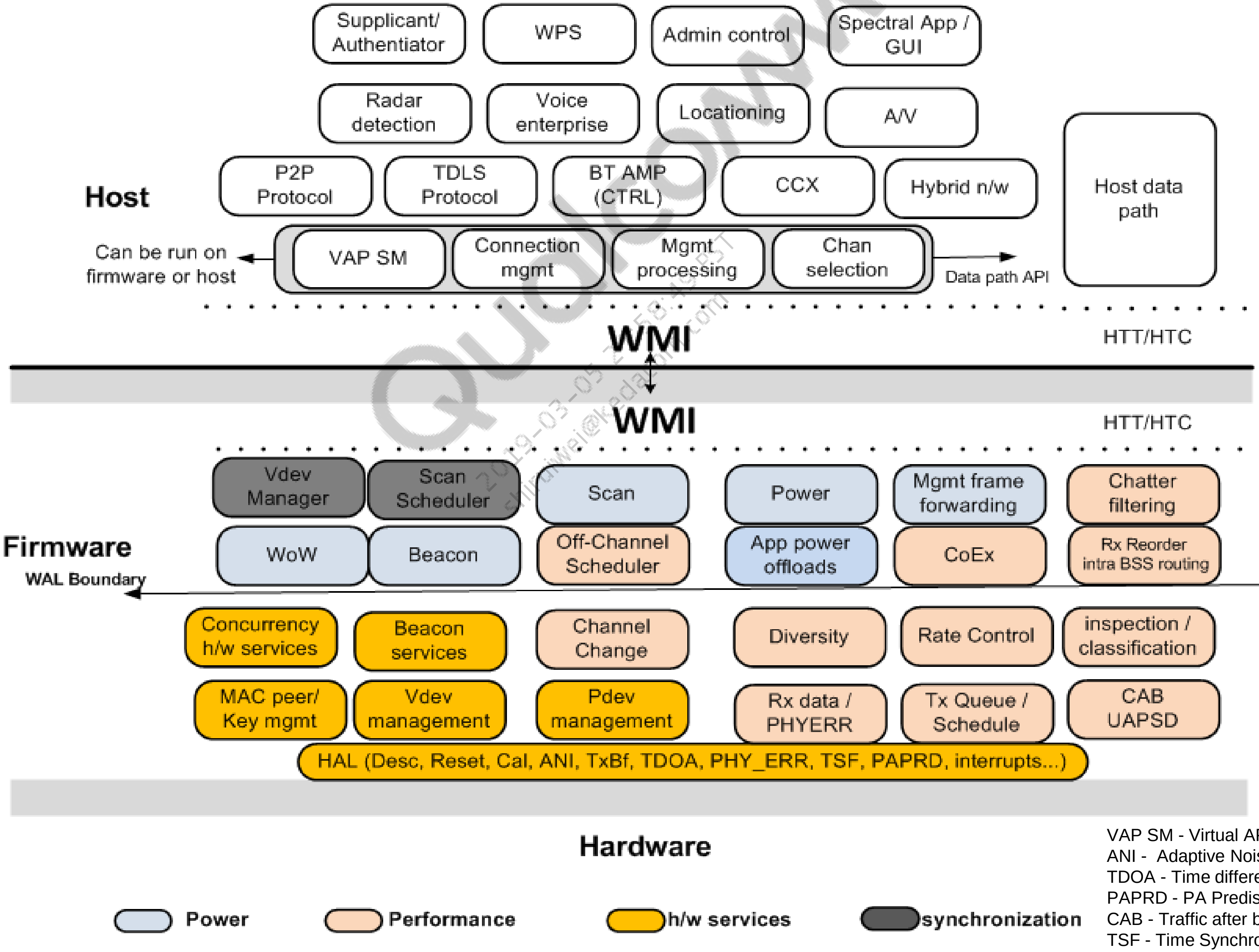
QCA6174 Wi-Fi Features (1 of 2)

Chain configuration	2x2 and 1x1
Spec compatibility	802.11a, b, g, n, ac 802.11d, e, h, i, j, k, r, u, v, w, z, ae
Bands Supported	2.4 and 5 GHz (including 4.9 GHz)
Bandwidths Supported	5, 10, 20, 40, 80 MHz
GI support	Full and Half GI
Coding schemes supported	Convolutional code and LDPC
MCSs supported	11ac: MCS 0 – MCS 9 11n: MCS 0 – MCS 15
Number of SU-MIMO spatial streams supported	1 and 2
DL MU-MIMO client configurations	1x1 single-stream MU client 2x2 single and two-stream MU client Interference suppression
Peak Throughput	PHY: 867 Mbps UDP: 724 Mbps TCP: 609 Mbps
Tx power levels (internal PA)	2G 22 dBm, 5G 17dBm per chain at chip output
2G Rx sensitivity / chain at chip input (3.6dB NF)	1Mbps/-98dBm 11Mbps/-89dBm MCS9/-69.5dBm
5G Rx sensitivity / chain at chip input (4.6dB NF)	80MHz: MCS9/-61.5dBm
Power save modes	MIMO-PS, UAPSD, Green TX, host power offload, Packet power save (PHY-NAP), Early Beacon termination

QCA6174 Wi-Fi Features (2 of 2)

Interface	PCIe, SDIO 3.0, USB 3.0
Aggregation schemes supported	AMSDU and AMPDU
Max length of AMSDU	11500 bytes
Encryption schemes supported	WAPI, WEP, AES(256), TKIP, CCMP
Advanced features	MU-MIMO client 11ac TxBFee HT-STF AGC STBC Digital Pre-Distortion Improved TPC (CLPC and OLPC) Programmable TxFIR Spectrum Analysis Improved fast channel switching Locationing with 802.11v support
Hardware Offload Engine	TCP/UDP/IP Checksum TCP Segmentation Offload HW assisted RTT engine for locationing
Other requirements	Proxy STA WDS mode On-chip OTP OTP enabled features
WFA certification	11n/ac, WMM, PMF, TDLS, Wi-Fi Direct, Hotspot 2.0
Certification	DFS, FCC, CCX

Baseline Architecture Overview



Supplicant/ Authentiator

WPS

Admin control

Spectral App / GUI

Radar detection

Voice enterprise

Locationing

A/V

Host

P2P Protocol

TDLS Protocol

BT AMP (CTRL)

CCX

Hybrid n/w

Host data path

Can be run on firmware or host

VAP SM

Connection mgmt

Mgmt processing

Chan selection

Data path API

WMI

HTT/HTC

WMI

HTT/HTC

Vdev Manager

Scan Scheduler

Scan

Power

Mgmt frame forwarding

Chatter filtering

Firmware

WAL Boundary

WoW

Beacon

Off-Channel Scheduler

App power offloads

CoEx

Rx Reorder intra BSS routing

Concurrency h/w services

Beacon services

Channel Change

Diversity

Rate Control

inspection / classification

MAC peer/ Key mgmt

Vdev management

Pdev management

Rx data / PHYERR

Tx Queue / Schedule

CAB UAPSD

HAL (Desc, Reset, Cal, ANI, TxBf, TDOA, PHY_ERR, TSF, PAPRD, interrupts...)

Hardware

Power

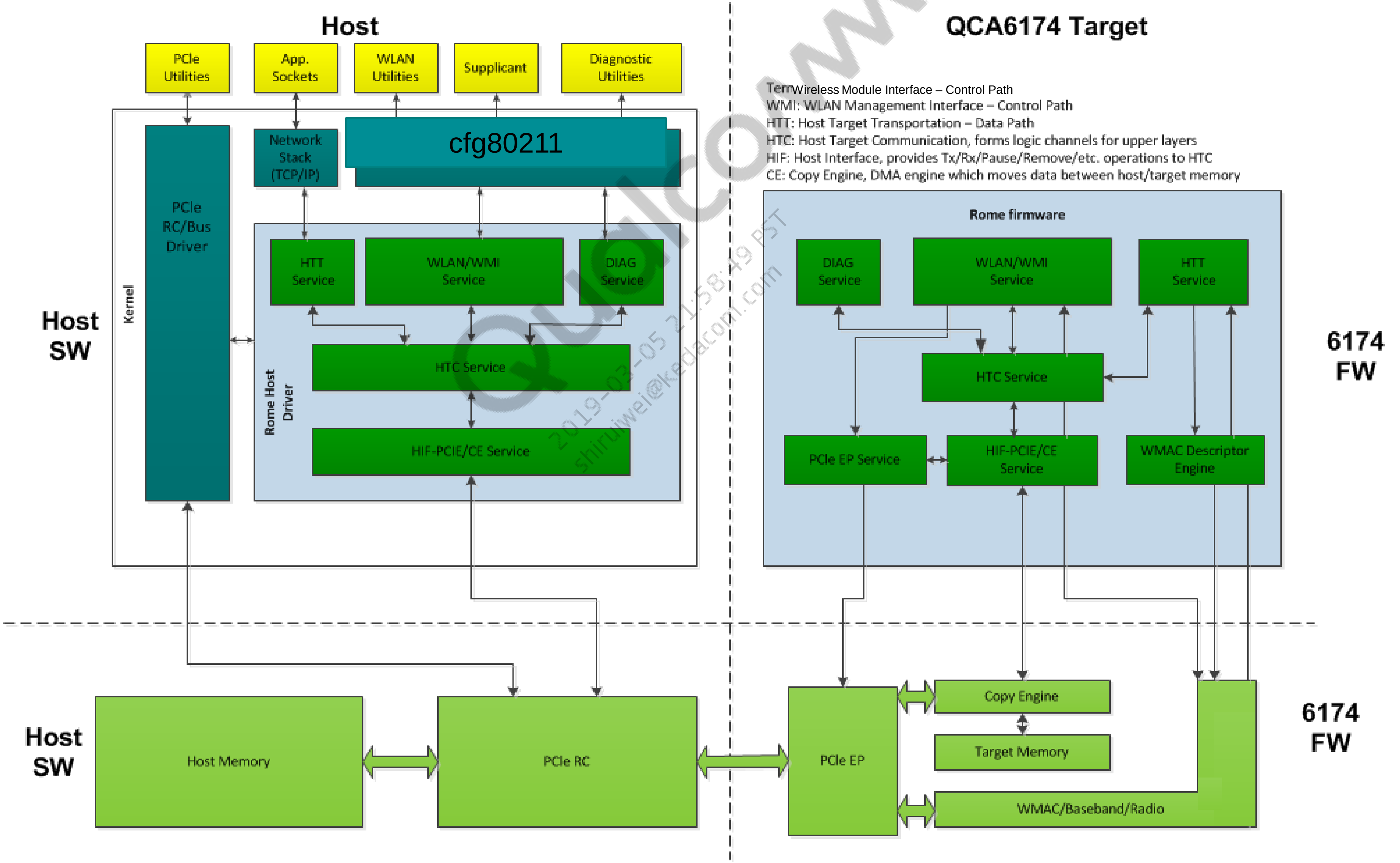
Performance

h/w services

synchronization

VAP SM - Virtual AP State Machine
ANI - Adaptive Noise Immunity
TDOA - Time difference of arrival
PAPRD - PA Predistortion
CAB - Traffic after beacon
TSF - Time Synchronization Function

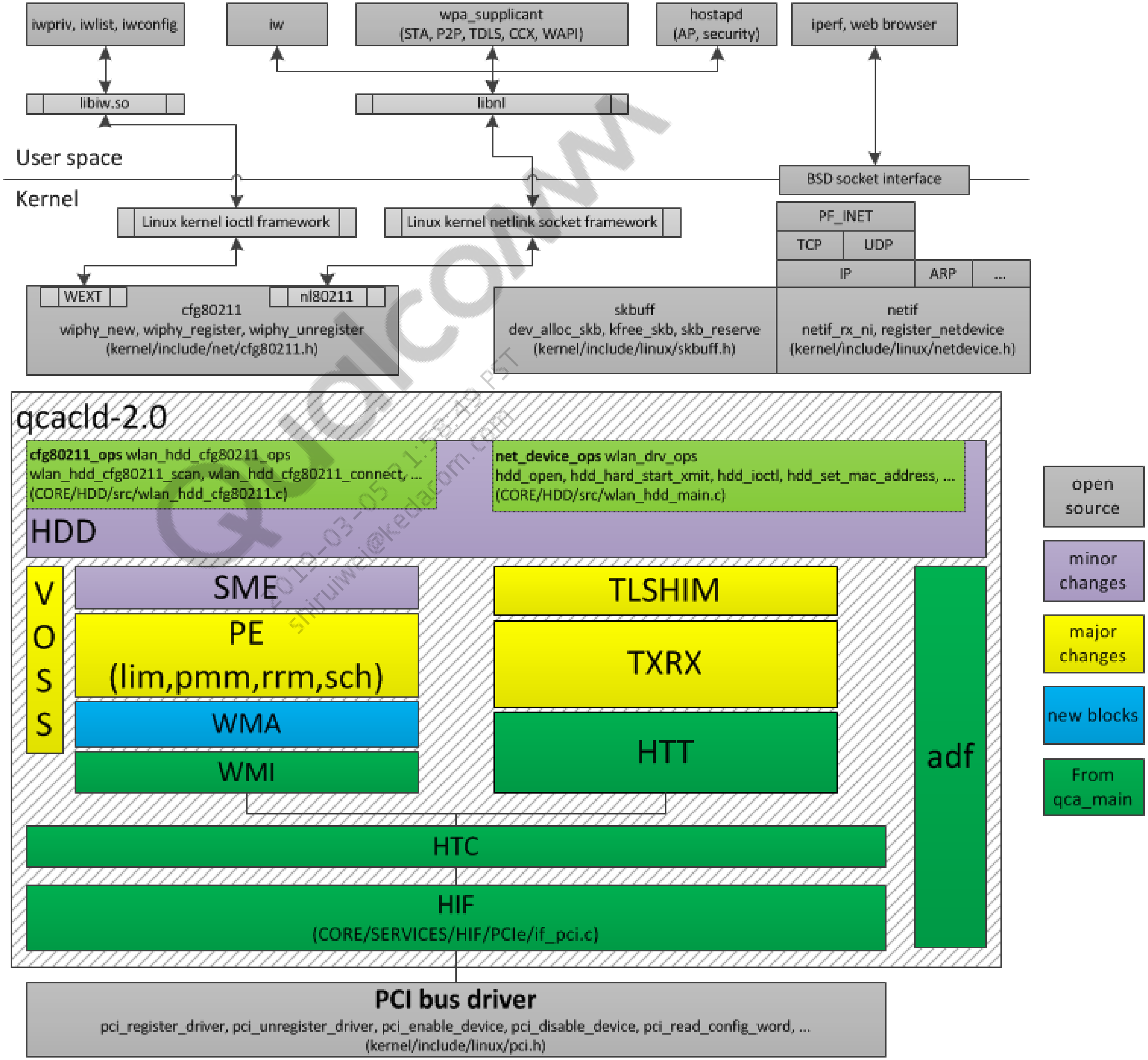
Host HW/SW Block Diagram



TemWireless Module Interface – Control Path
WMI: WLAN Management Interface – Control Path
HTT: Host Target Transportation – Data Path
HTC: Host Target Communication, forms logic channels for upper layers
HIF: Host Interface, provides Tx/Rx/Pause/Remove/etc. operations to HTC
CE: Copy Engine, DMA engine which moves data between host/target memory

Driver

Driver



iwpriv, iwlist, iwconfig

iw

wpa_supplicant
(STA, P2P, TDLS, CCX, WAPI)

hostapd
(AP, security)

iperf, web browser

libiw.so

libnl

User space

Kernel

BSD socket interface

Linux kernel ioctl framework

Linux kernel netlink socket framework

PF_INET

TCP

UDP

IP

ARP

...

WEXT

nl80211

cfg80211
wiphy_new, wiphy_register, wiphy_unregister
(kernel/include/net/cfg80211.h)

skbuff
dev_alloc_skb, kfree_skb, skb_reserve
(kernel/include/linux/skbuff.h)

netif
netif_rx_ni, register_netdevice
(kernel/include/linux/netdevice.h)

qcacld-2.0

cfg80211_ops wlan_hdd_cfg80211_ops
wlan_hdd_cfg80211_scan, wlan_hdd_cfg80211_connect, ...
(CORE/HDD/src/wlan_hdd_cfg80211.c)

net_device_ops wlan_drv_ops
hdd_open, hdd_hard_start_xmit, hdd_ioctl, hdd_set_mac_address, ...
(CORE/HDD/src/wlan_hdd_main.c)

HDD

VOSS

SME

PE
(lim,pmm,rrm,sch)

WMA

WMI

TLSHIM

TXRX

HTT

adf

HTC

HIF
(CORE/SERVICES/HIF/PCIe/if_pci.c)

PCI bus driver
pci_register_driver, pci_unregister_driver, pci_enable_device, pci_disable_device, pci_read_config_word, ...
(kernel/include/linux/pci.h)

open source

minor changes

major changes

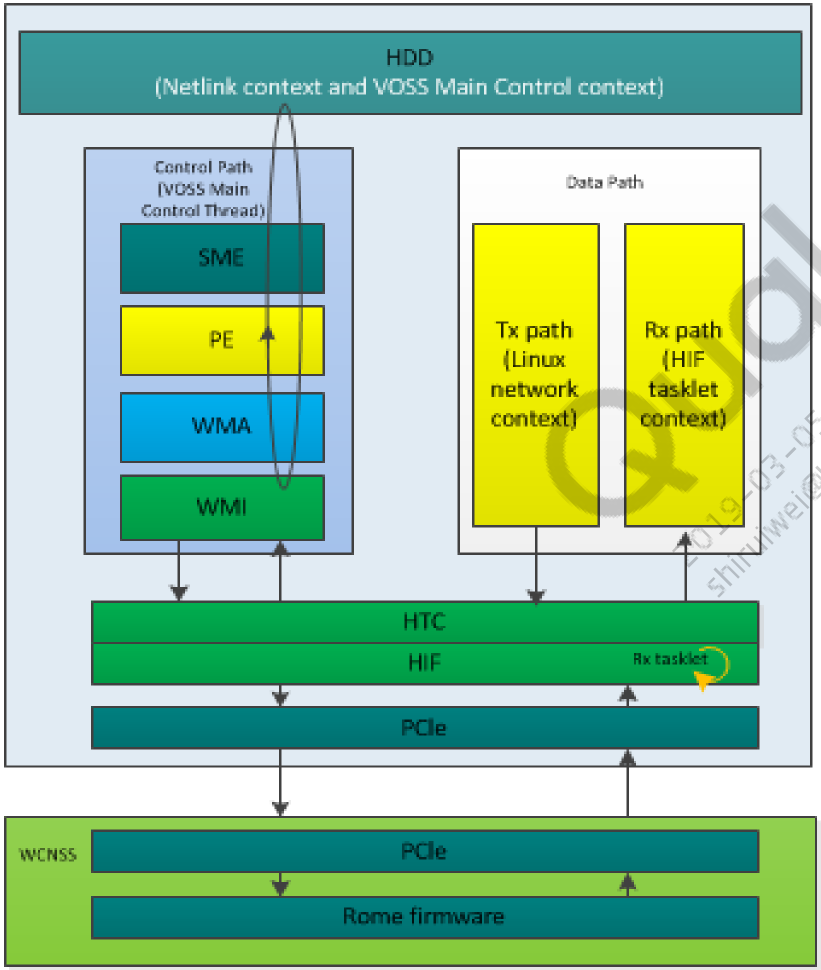
new blocks

From qca_main

QCA6174 LA Driver

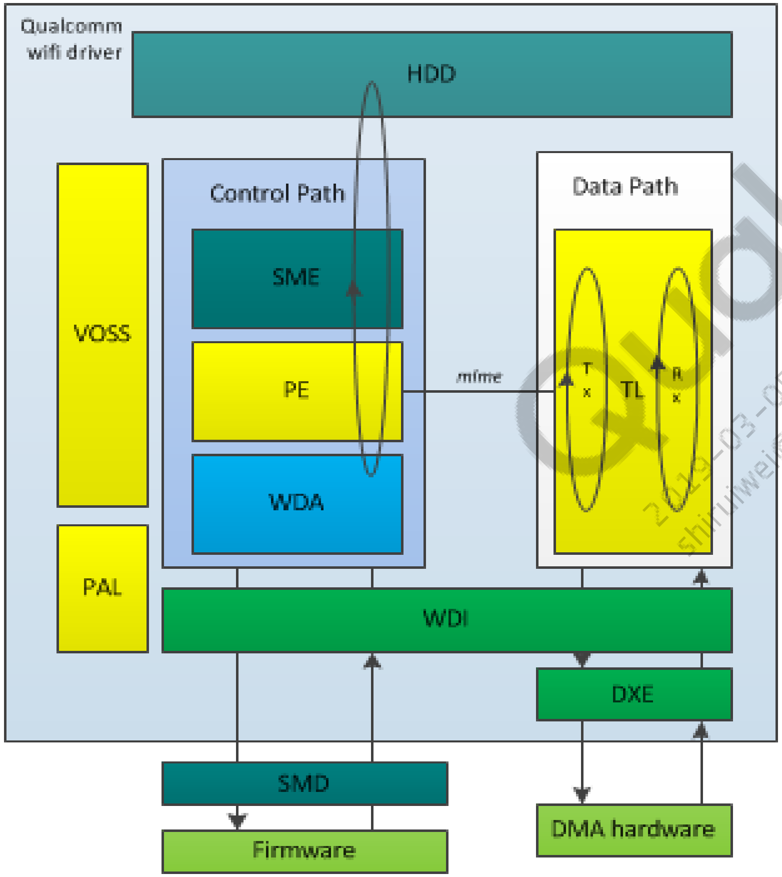
Layer	Description
HDD	Host Device Driver Interface with user space for control path (all user configuration comes through this layer) and interface with network stack in data path **Northbound**: HDD registers into netdevice and cfg80211 callbacks and implements the associated operation; also implements the private IOCTLs **Southbound**: Implements control plane operation by using the services of SME and on the data plane by interfacing with TXRX module
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 WLAN MAC driver
PE	Protocol Engine Implements the core Protocol Engine (ie. MLME)
VOSS	ALL OS-related implementation is done in VOSS layer
WMA	WMI Adaptation Layer Provides an adaptation layer to WMI
WMI	Wireless Module Interface Control Path Messaging interface between Host and FW
TL SHIM TXRX	Provides the abstraction to TxRx data and HDD that helps leverage most of the HDD and entire UMAC from the WCN36x0 driver
HTT	Host Target Transport Handle host’s side of tx/rx data frame flow to/from target SOCs
HTC	Host Target Communication Mbox for providing host/target flow control, TX/RX queue management and QoS
HIF	Host Interface Abstracts the hardware interface (PCI, USB, SDIO)

CLD 2.0 SW Threading Model: Discrete Solution (QCA6174)



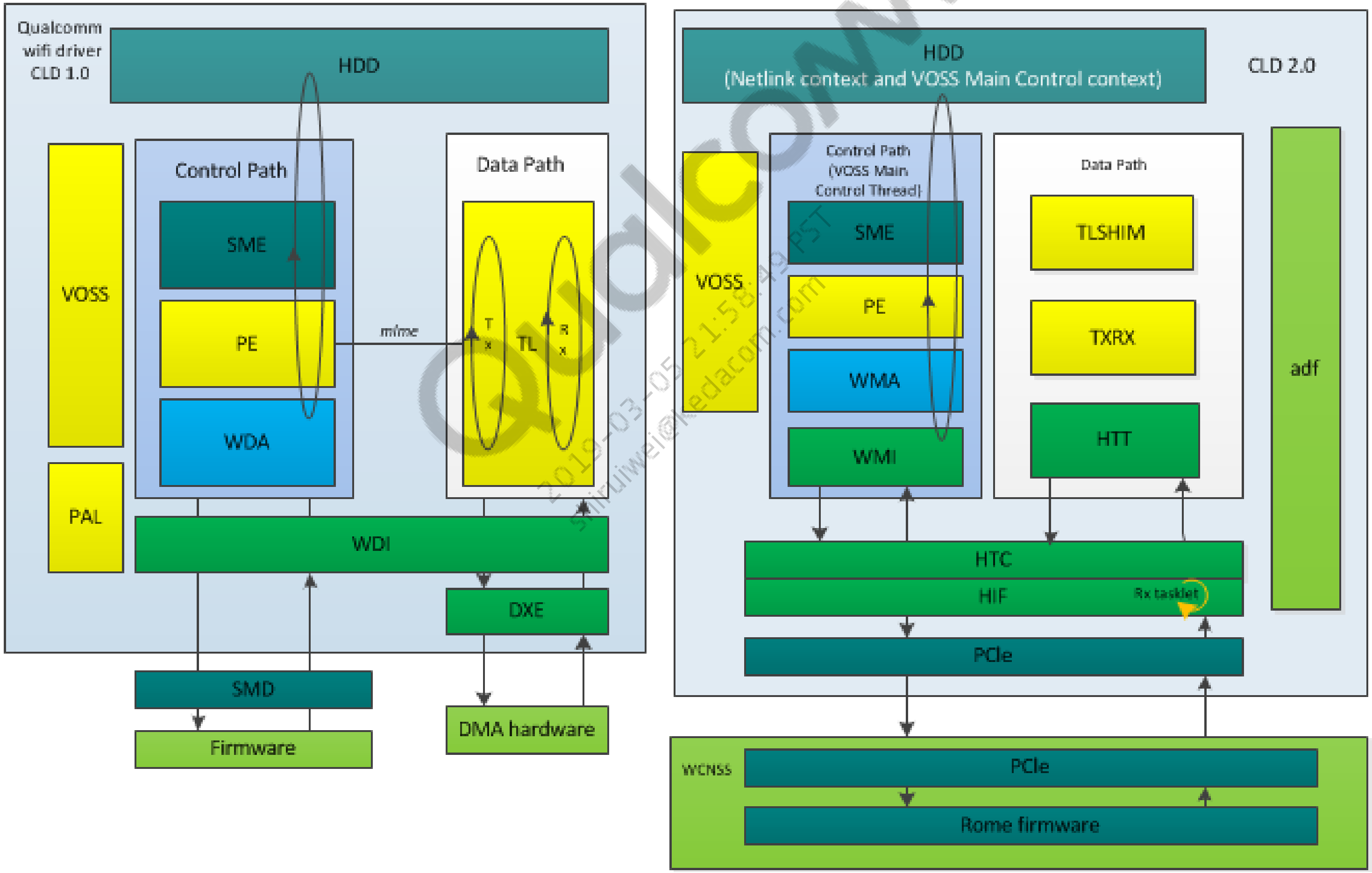
- Control path:
  - HDD callback primitive interface (outgoing) runs in Linux network context and the lower UMAC path (incoming) runs in VOSS main control thread context
  - UMAC, WMA, WMI, run in VOSS main control thread context
  - HTC and HIF outgoing path runs in main control thread context.
  - HTC and HIF incoming path runs in RX tasklet context
- Data path:
  - Tx path runs in network context
  - Rx path runs in the HIF Rx tasklet context

SW Delta between Pronto and QCA6174: CLD 1.0 Host Driver and Execution Model



- WDA: WDI adaptor layer for UMAC
- WDI: WiFi device interface for both control path and data path
- SMD used between LA host driver and firmware (ISOC) in control path
- DXE for data path (direct access to DMA hardware)
- Three VOSS threads:
  - Main control thread for control path
  - Tx data path thread
  - Rx data path thread

Software Delta between Pronto and QCA6174



Software Delta between Pronto and QCA6174: Components and Execution Model

- Pronto:
  - One VOSS main control thread for control path
  - One Tx data path thread for priority based data packet processing
  - One Rx data path thread for priority based data packet processing
  - WDA: WDI adaptor layer for UMAC
  - WDI: WiFi device interface for both SMD (control path) and platform bus (DXE data path)
  - DXE for Tx and Rx
  - WCNSS: ISOC

- QCA6174:
  - One VOSS main control thread for control path
  - Immediate processing of Tx data packet
  - One RX data path tasklet for priority based Rx data packet processing
  - WMA: equivalent to Pronto WDA
  - WMI: equivalent to Pronto WDI. But WMI resides only at control path
  - New HTC and HIF components for supporting PCIe
  - New transport layer: TLShim, TXRX, and HTT
  - PCIe for Tx & Rx (No DXE)

Software Delta between Pronto and QCA6174: Firmware Loading and Host Driver init

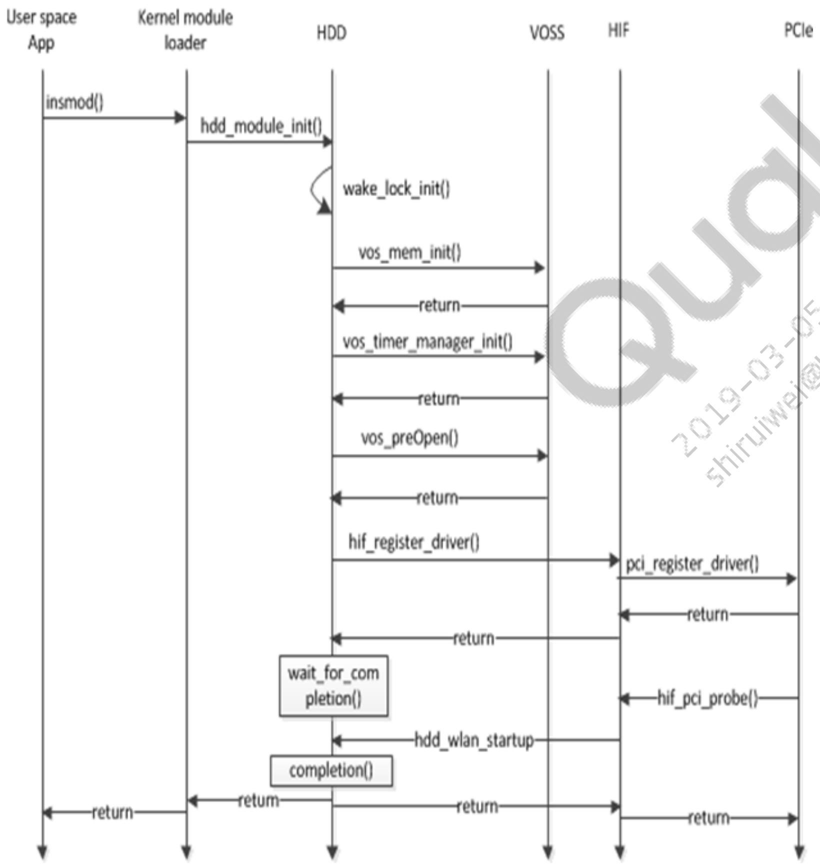
- Pronto (ISOC):
  - Firmware is loaded and initialized when the Linux kernel boots
  - When the user space triggers the LA host driver hdd_driver_init(), the hdd_wlan_startup() is called in hdd_driver_init() function. If the driver startup failed the driver is uninitialized

- QCA6174 (Discrete Solution):
  - When the Linux kernel boots the QCA6174 firmware is not loaded
  - When the user space triggers the LA host driver hdd_driver_init(), pci_register_driver() is called to trigger QCA6174 PCI device probing
  - The hdd_wlan_startup() is called by the PCIe probing callback. Firmware loading happens in hdd_wlan_startup()

Initialization for QCA6174

- insmod to load LA driver
- hif_pci_prob
- hdd_wlan_startup

Initialization Sequence: insmod() to Load LA Host Driver

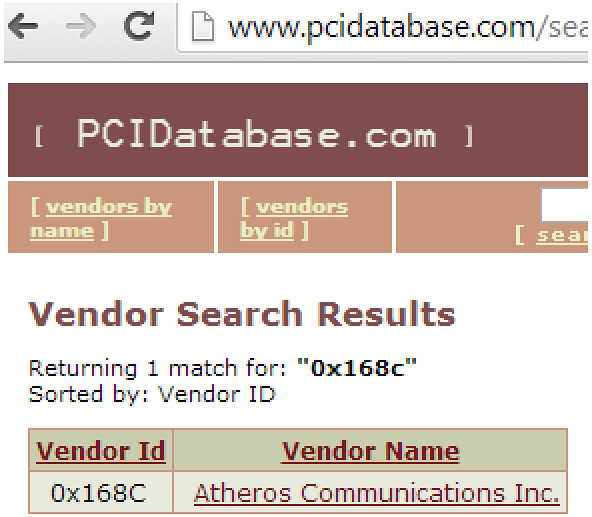


- Call insmod() to explicitly load the LA host driver
- hdd_driver_init() is called to initialize the LA host driver
- Call net core to get PCIe platform driver dev struct
- Wait till NV configuration finished
- Call pci_register_driver() to prob PCI device (probing will call hdd_wlan_startup)
- Wait for hdd_wlan_startup() to complete (it's called by hif_pci_prob)
- Return 0 for success or -1 for wait timeout

PCI bus driver callbacks

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,
};
```

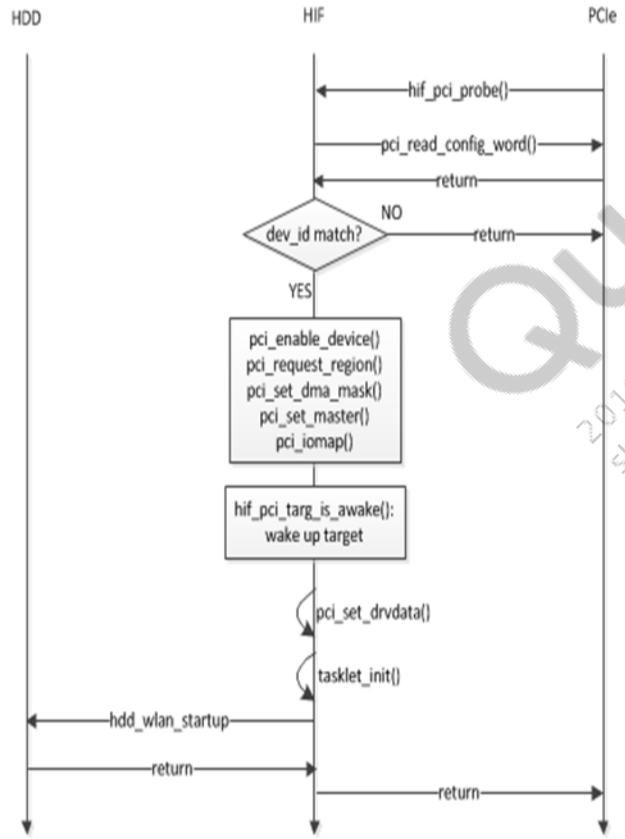


Vendor Search Results

Returning 1 match for: "0x168c"
Sorted by: Vendor ID

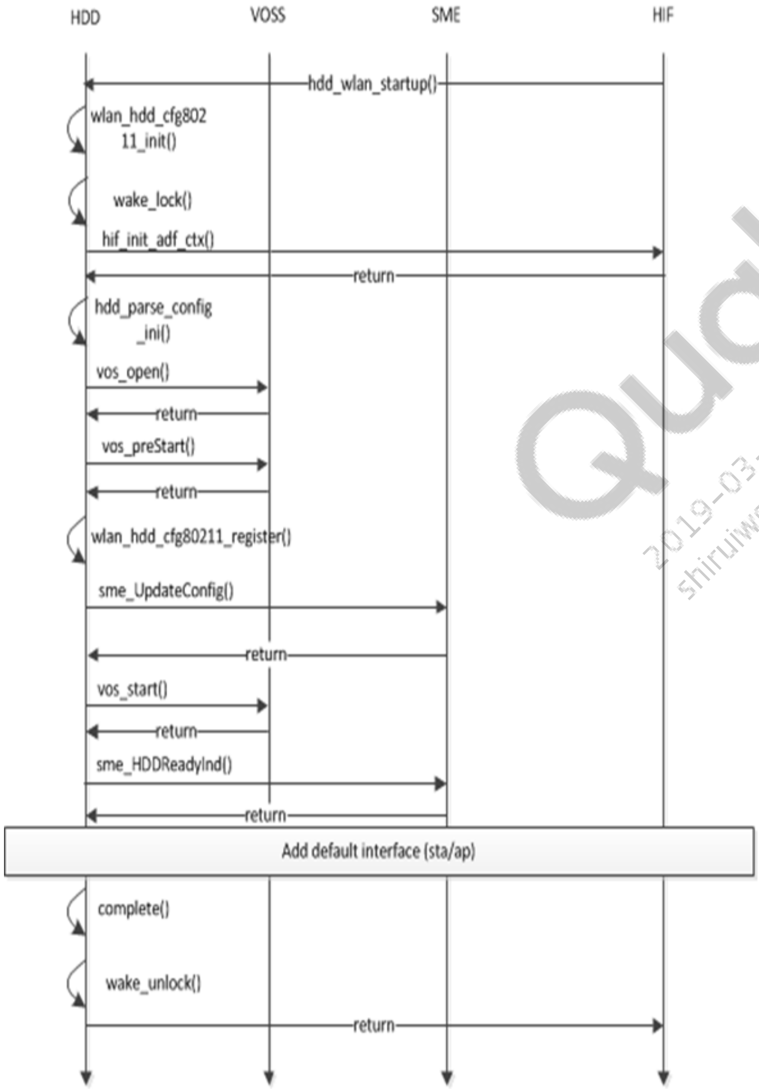
Vendor Id	Vendor Name
0x168C	Atheros Communications Inc.

Host Driver Initialization Sequence for QCA6174: hif_pci_prob()



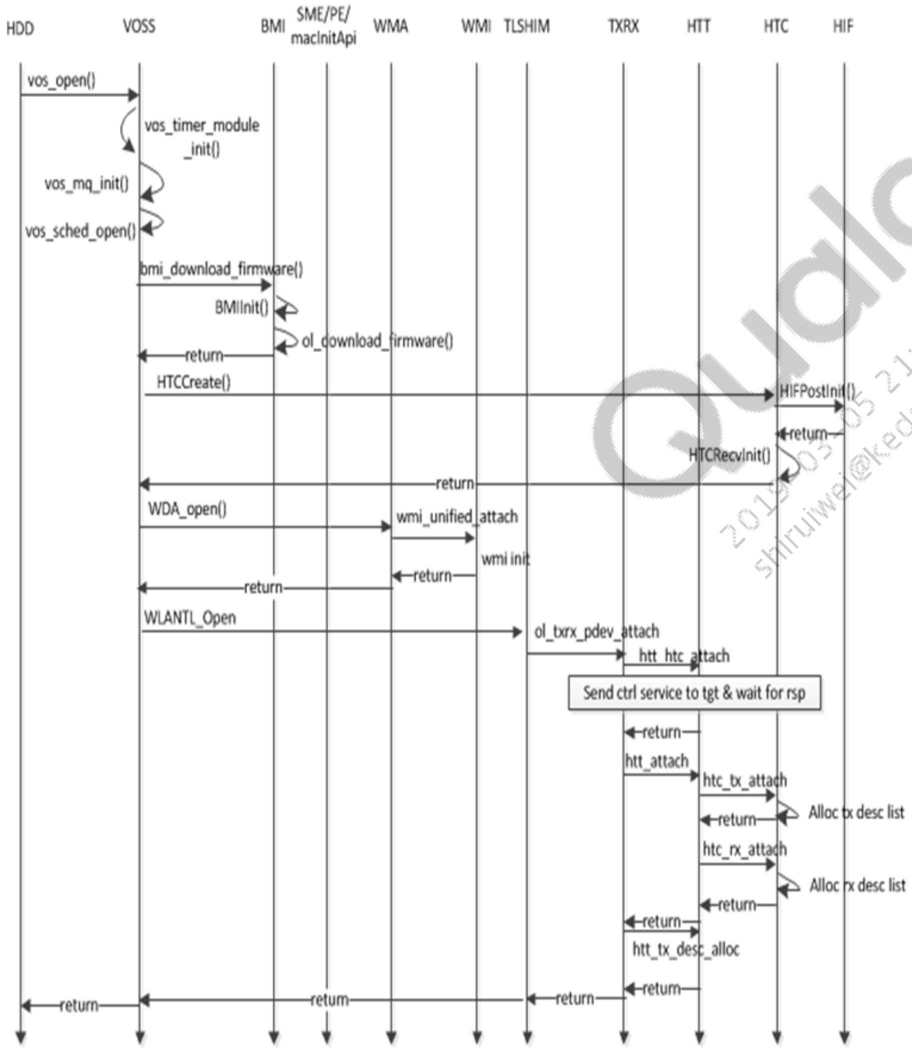
- hif_pci_prob() is triggered by pci_register_driver()
- pci_read_config_word() gets the current PCI device ID; if device_id does not match return error
- Enable device and register region
- Call pci_set_master() to enable DMA
- hif_pci_targ_is_awake() to wake up WCNSS
- tasklet_init
- hdd_wlan_startup()

Host Driver Initialization Sequence for QCA6174: hdd_wlan_startup()



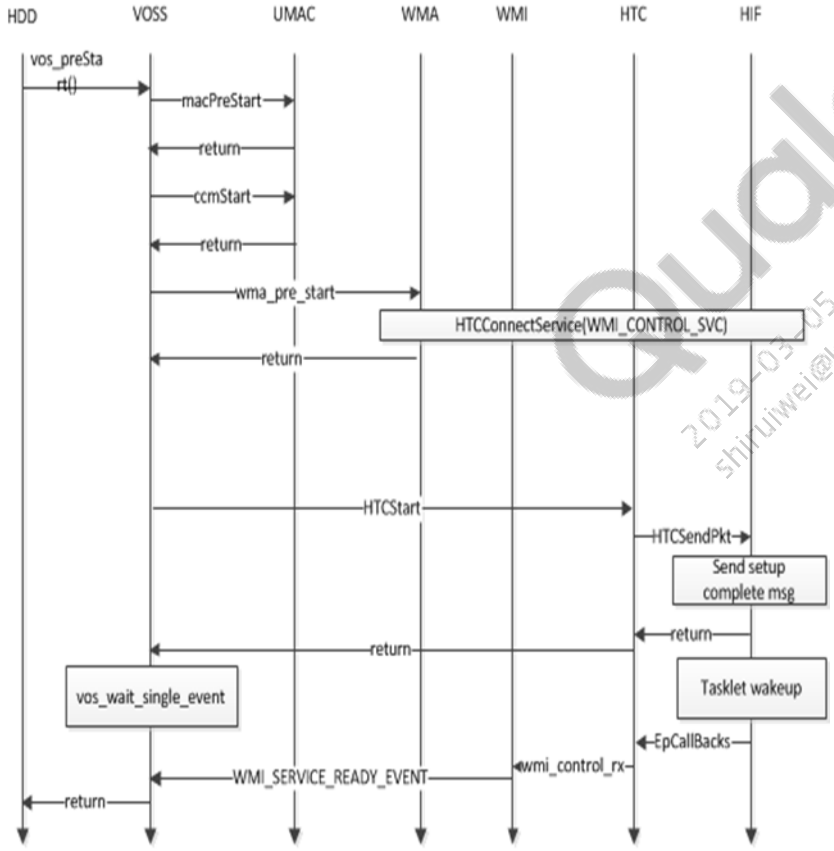
- wlan_hdd_cfg89211_init: call wiphy_new() to register cfg80211_ops
- wake_lock(): not allow suspend
- hif_init_adf_ctx: init adf context
- Call vos_open()
- Call vos_preStart()
- Call vos_start()
- Add default virtual interface (sta/ap)

Host Driver Initialization Sequence for QCA6174: vos_open()



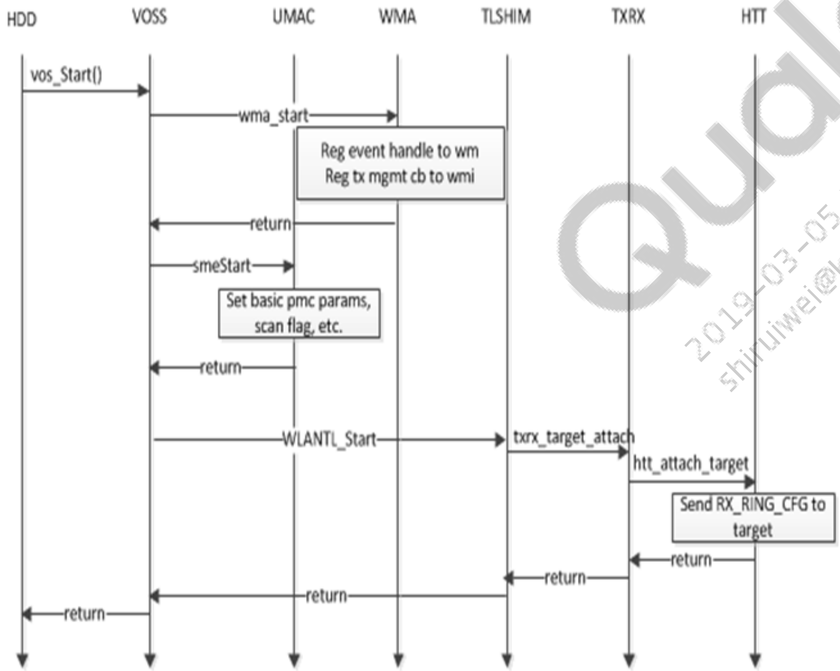
- vos_sched_open: Create main control thread.
- Bmi_download_firmware: (1) get the target_info; (2) config the target; (3) download the firmware to target RAM
- HTCCreate: memory allocation, spinlock init, etc.
- WDA_open: WMA and WMI init

Host Driver Initialization Sequence for QCA6174: vos_preStart()



- macPreStart: alloc and zero mem
- wma_pre_start: connect HTC_DATA_MSG_SRV to target
- HTCStart: send HTC_MSG_SETUP_COMPLETE_EX_ID to target and wait for the response

Host Driver Initialization Sequence for QCA6174: vos_start()



- wma_start: register event cb to wmi, set tx mgmt cb to wmi
- smeStart: set basic pmc params and state, set scan flag
- WLANTL_Start: send rx ring cfg to target

WLAN Bringup Sequence Log

Note: See the wifi_bringup.log file that is attached to the PDF.

- pci_driver probe

```
<6>[  166.305944] hif_pci_probe
<6>[  166.305952] msm_pcie_oper_conf: rd - bus 1 devfn 0
<6>[  166.305958] msm_pcie_oper_conf: 1:0x00 + 0x0002[2] -> 0x0000003e; rd 0x003e168c
```

- 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
```

- 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
```

- 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
```

SoftMAC

- User can overwrite the MAC address store in the OTP by
  - /persist/wlan_mac.bin

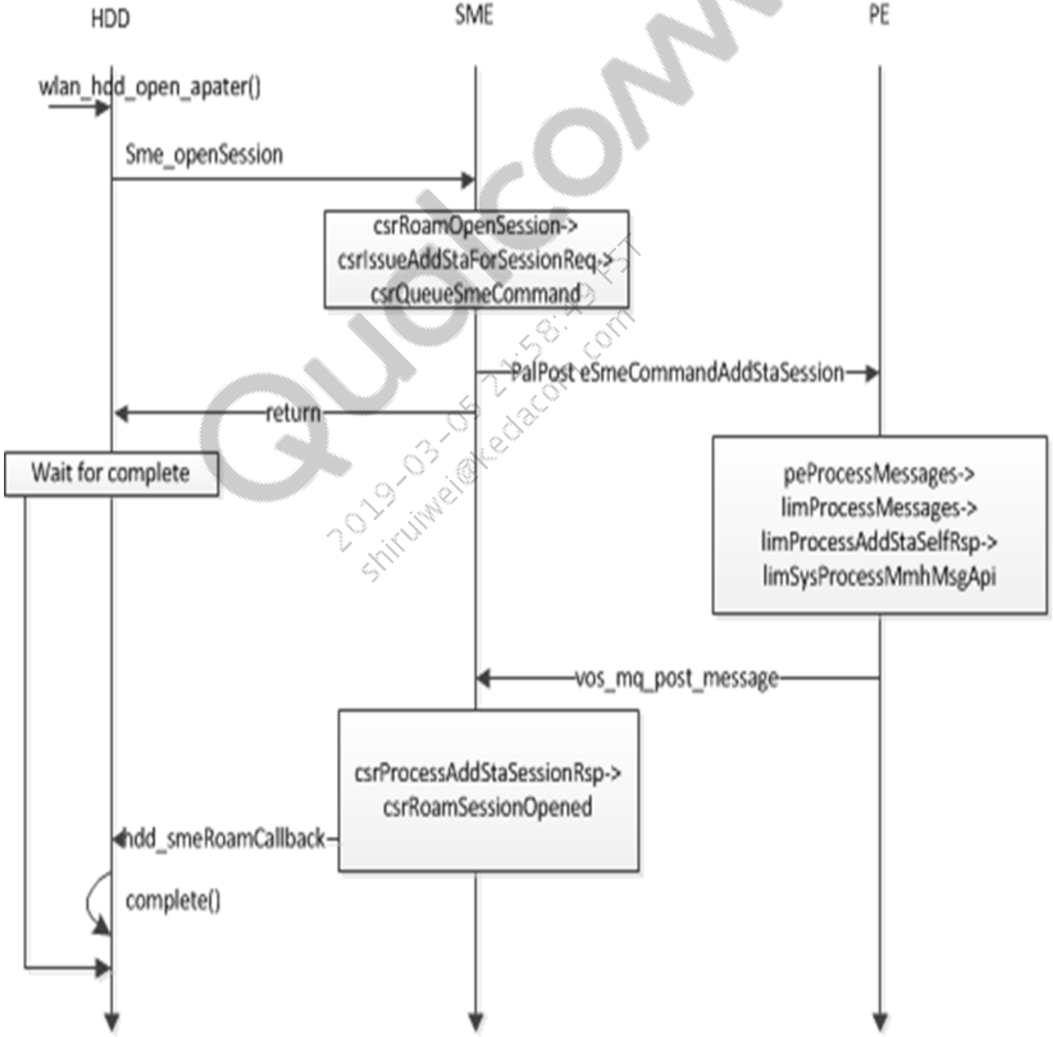
  - Intf0MacAddress=002233445566
  - Intf1MacAddress=002223242526
  - Intf2MacAddress=003233343536
  - Intf3MacAddress=004243444556
  - END

    - Intf0MacAddress - STA and AP mode (wlan0)
    - Intf1MacAddress - P2P (p2p0)
    - Intf2MacAddress - Not used
    - Intf3MacAddress - Not used

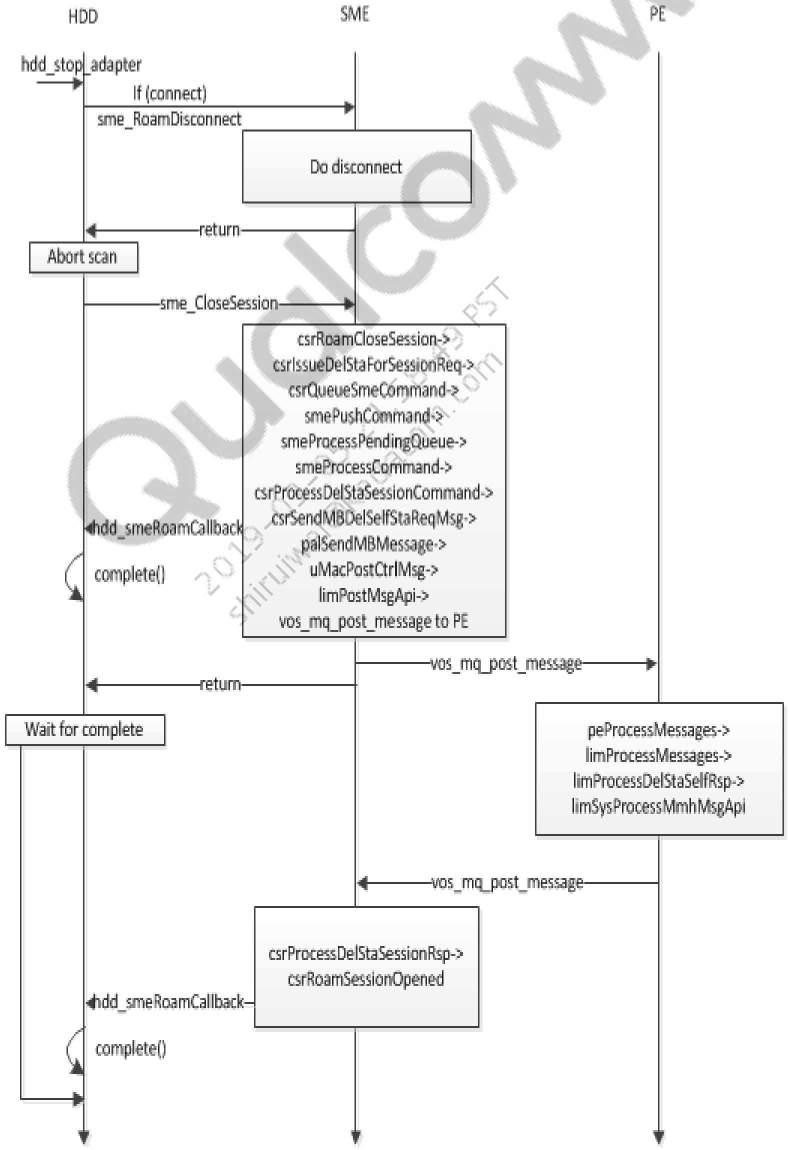
Control Path for QCA6174

- Add virtual interface
- Delete virtual interface
- Scan request
- Connect
- Disconnect (host-initiated and AP-initiated)

Control Path for QCA6174: Add Virtual Interface



Control Path for QCA6174: Delete Virtual Interface



HDD

SME

PE

hdd_stop_adapter

If (connect)

sme_RoamDisconnect

Do disconnect

return

Abort scan

sme_CloseSession

csrRoamCloseSession->
csrIssueDelStaForSessionReq->
csrQueueSmeCommand->
smePushCommand->
smeProcessPendingQueue->
smeProcessCommand->
csrProcessDelStaSessionCommand->
csrSendMBDelSelfStaReqMsg->
palSendMBMessage->
uMacPostCtrlMsg->
limPostMsgApi->
vos_mq_post_message to PE

hdd_smeRoamCallback

complete()

vos_mq_post_message

return

Wait for complete

peProcessMessages->
limProcessMessages->
limProcessDelStaSelfRsp->
limSysProcessMmhMsgApi

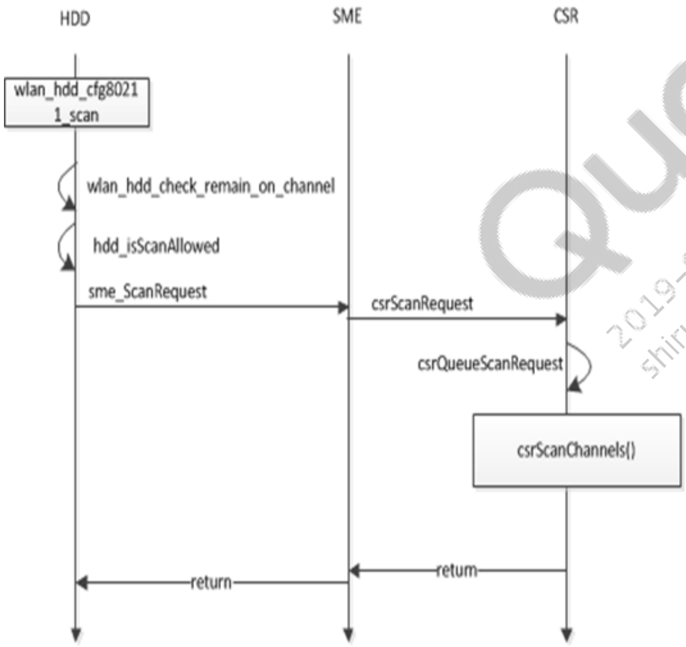
vos_mq_post_message

csrProcessDelStaSessionRsp->
csrRoamSessionOpened

hdd_smeRoamCallback

complete()

Control Path for QCA6174: Scan Request



- If wlan_hdd_check_remain_on_channel failed return error
- hdd_isScanAllowed: if in the middle of WPS/EAPOL exchange no scan allowed
- csrScanChannels: send the scan request to target and return
- Scan result is sent to user space by cfg80211_sched_scan_results after the scan is completed

Control Path for QCA6174: Scan Request Log

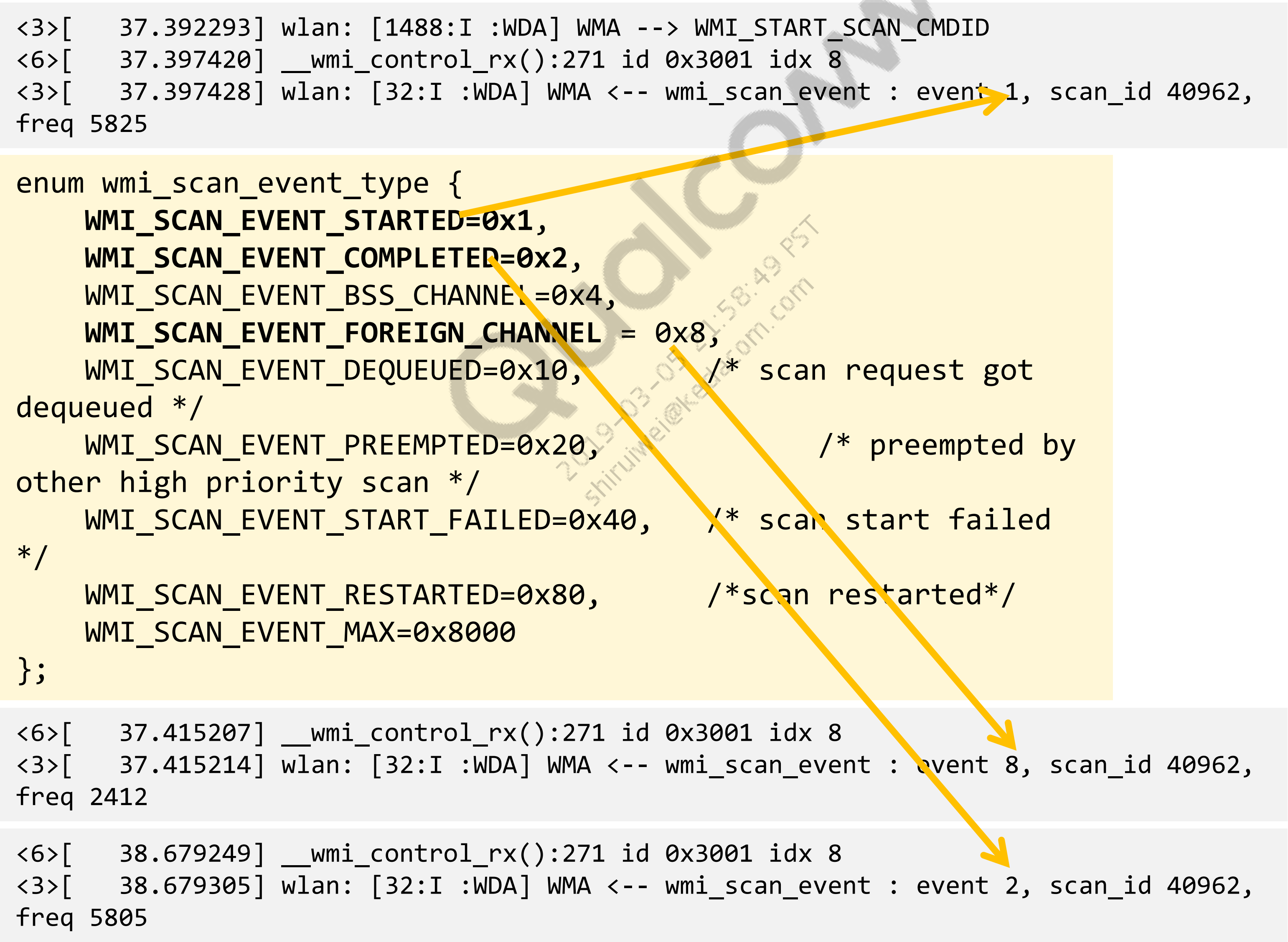
```
wlan: [1521:I :HDD] Enter:wlan_hdd_cfg80211_scan

wlan: [1521:I :HDD] wlan_hdd_cfg80211_scan: device_mode = 0


wlan: [1521:I :HDD] hdd_isScanAllowed: Adapter with device mode 0 exists

wlan: [1521:I :HDD] hdd_isScanAllowed: Adapter with device mode 7 exists

wlan: [1521:I :HDD] hdd_isScanAllowed: Scan allowed

wlan: [1521:I :HDD] scan request for ssid = 1

wlan: [1521:I :SME]  Processing scan offload command

wlan: [1521:I :SME] CSR RoamState[0]: [ 2 <== 1 ]

wlan: [1521:I :SME] csrSendMBScanReq: 5234: domainIdCurrent 7 scanType 1 bssType 5 requestType 2
numChannels 32

wlan: [1488: D:WDA] wma_mc_process_msg: Enter

wlan: [1521:I :HDD] Exit:wlan_hdd_cfg80211_scan
```

```
typedef enum device_mode
{
  WLAN_HDD_INFRA_STATION,
  WLAN_HDD_SOFTAP,
  WLAN_HDD_P2P_CLIENT,
  WLAN_HDD_P2P_GO,
  WLAN_HDD_MONITOR,
  WLAN_HDD_FTM,
  WLAN_HDD_IBSS,
  WLAN_HDD_P2P_DEVICE
} device_mode_t;
```

smeProcessScanQueue()

Control Path for QCA6174: Scan Event Log

Note: See the scan_host_driver.log file that is attached to the PDF.



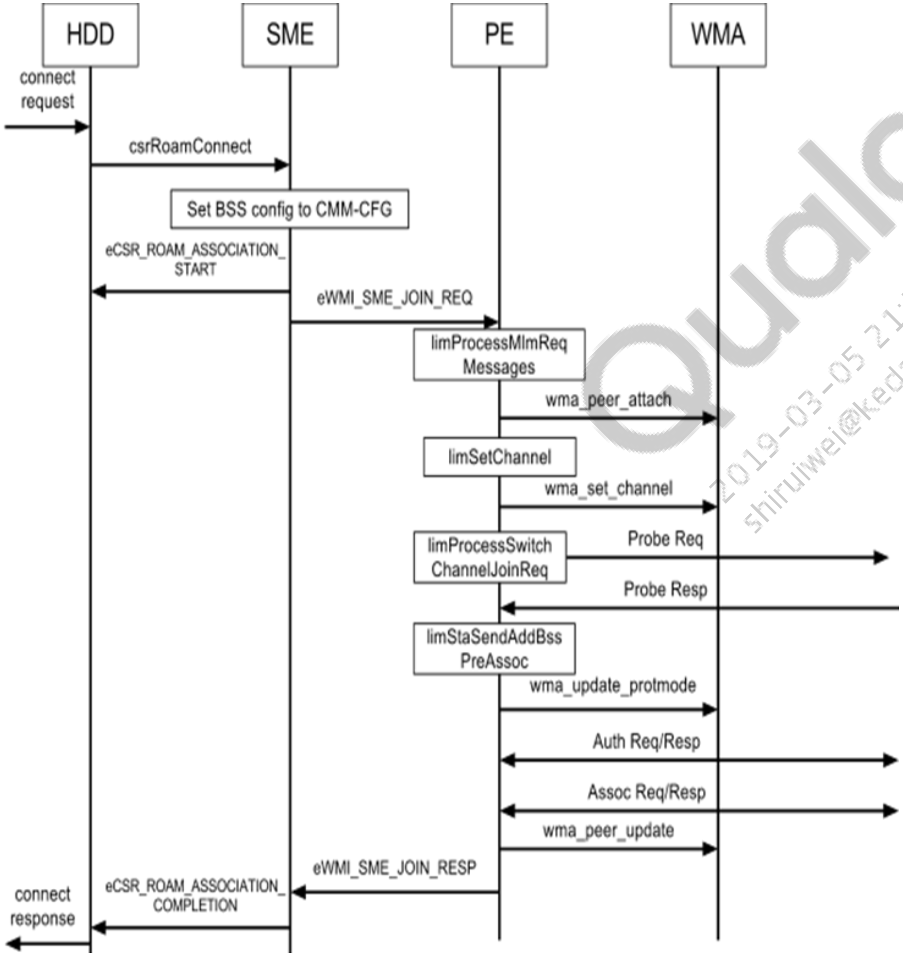
```
<3>[   37.392293] wlan: [1488:I :WDA] WMA --> WMI_START_SCAN_CMDID
<6>[   37.397420] __wmi_control_rx():271 id 0x3001 idx 8
<3>[   37.397428] wlan: [32:I :WDA] WMA <-- wmi_scan_event : event 1, scan_id 40962,
freq 5825
```

```
enum wmi_scan_event_type {
    WMI_SCAN_EVENT_STARTED=0x1,
    WMI_SCAN_EVENT_COMPLETED=0x2,
    WMI_SCAN_EVENT_BSS_CHANNEL=0x4,
    WMI_SCAN_EVENT_FOREIGN_CHANNEL = 0x8,
    WMI_SCAN_EVENT_DEQUEUED=0x10,      /* scan request got
dequeued */
    WMI_SCAN_EVENT_PREEMPTED=0x20,            /* preempted by
other high priority scan */
    WMI_SCAN_EVENT_START_FAILED=0x40,   /* scan start failed
*/
    WMI_SCAN_EVENT_RESTARTED=0x80,      /*scan restarted*/
    WMI_SCAN_EVENT_MAX=0x8000
};
```

```
<6>[   37.415207] __wmi_control_rx():271 id 0x3001 idx 8
<3>[   37.415214] wlan: [32:I :WDA] WMA <-- wmi_scan_event : event 8, scan_id 40962,
freq 2412
```

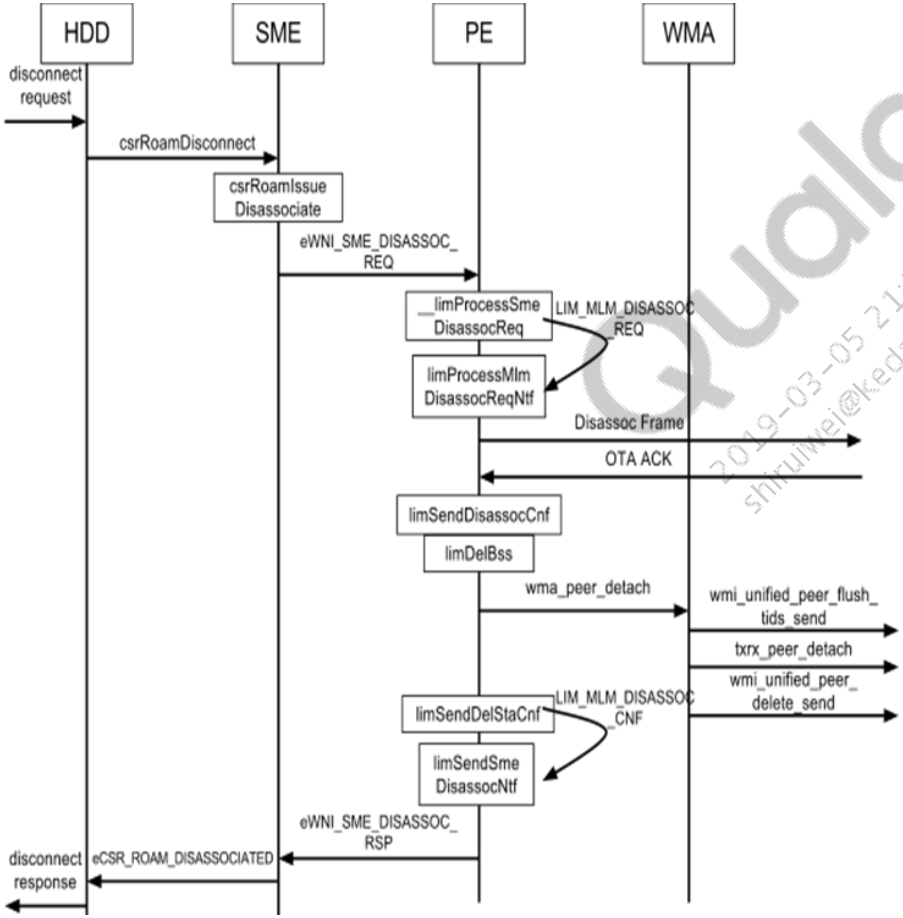
```
<6>[   38.679249] __wmi_control_rx():271 id 0x3001 idx 8
<3>[   38.679305] wlan: [32:I :WDA] WMA <-- wmi_scan_event : event 2, scan_id 40962,
freq 5805
```

Control Path for QCA6174: Connect



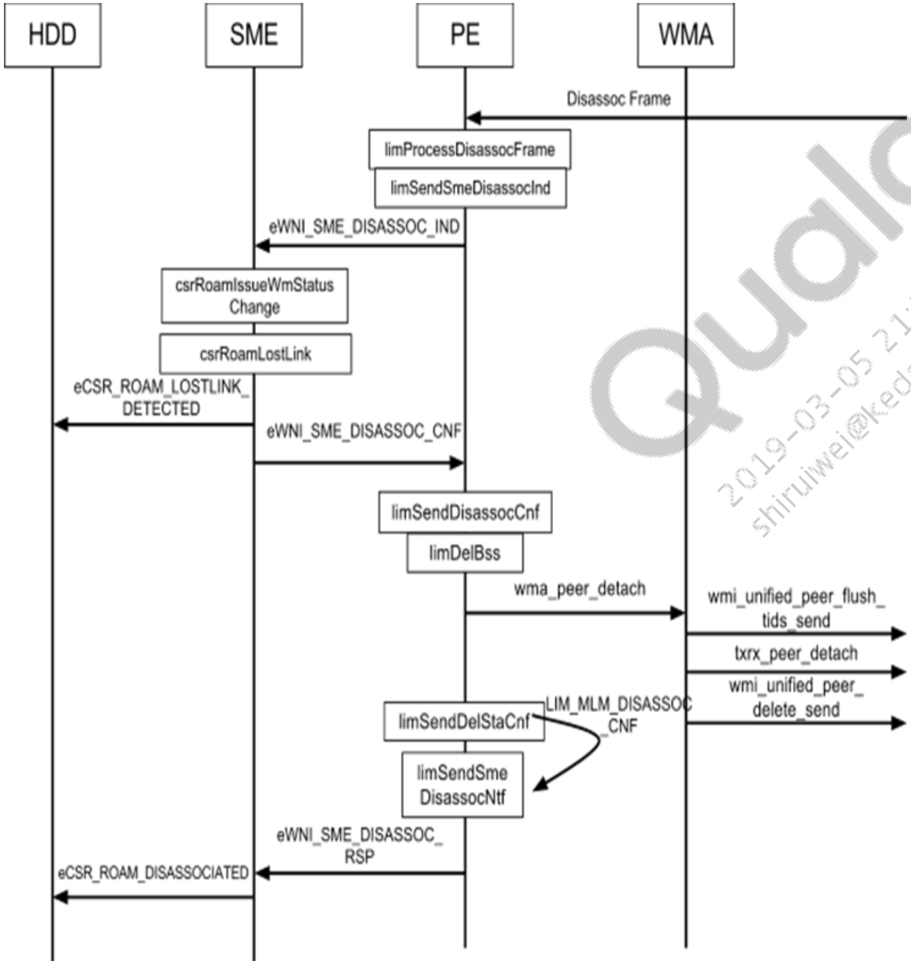
- wlan_hdd_cfg80211_connect is HDD callback for connecting to BSS network; separate sessions are maintained in SME and PE layers for a single roam session

Control Path for QCA6174: Disconnect (Host-Initiated)



- Host-initiated disconnect: disconnect STA, AP, P2P_GO, P2P_CLIENT, etc.

Control Path for QCA6174: Disconnect (AP-Initiated)

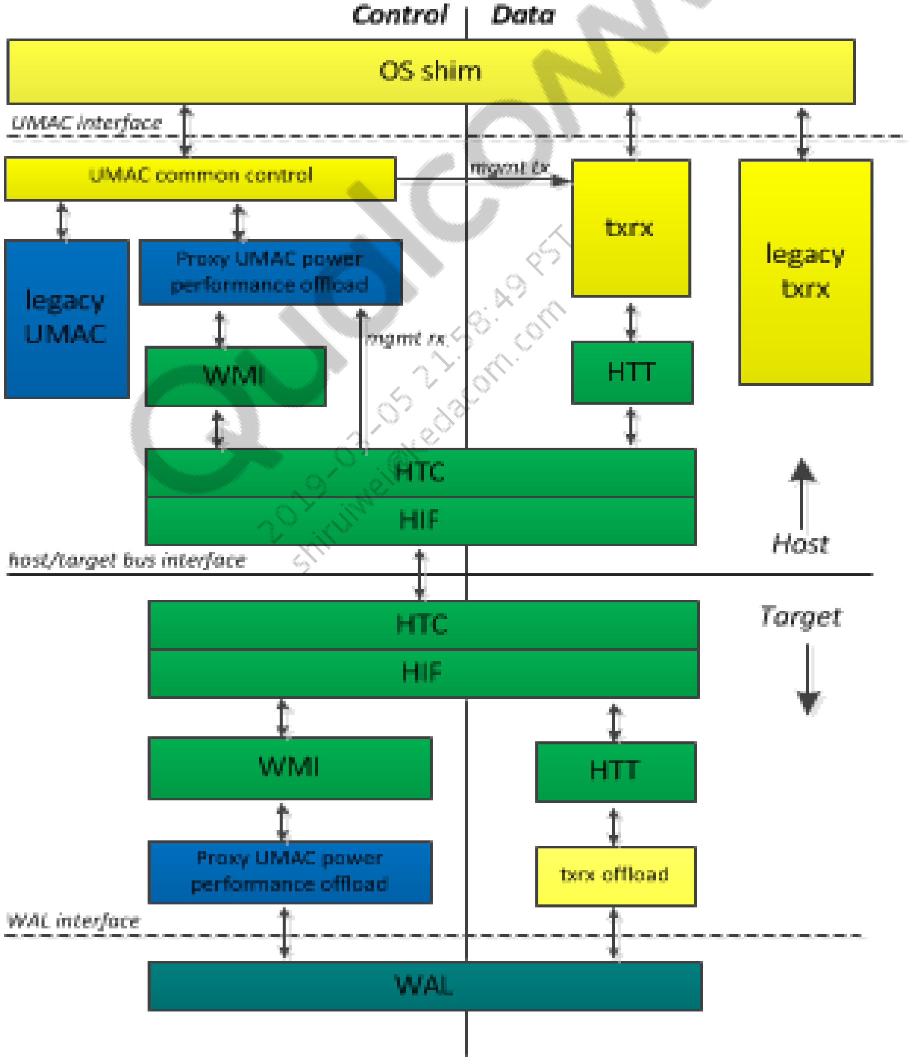


- AP-initiated disconnect

Subsystem Recovery: Overview

- When the firmware encounters a fatal error, exception or hang, a PCI interrupt is raised
- The WLAN platform driver processes the interrupt and passes it to the ol_target_failure() function
- The ol_target_failure() function collects all the necessary firmware details – the stack trace, core dump of the Iram, Dram and entire register set from the QCA6174 chip
- The WLAN platform driver performs the following actions:
  - Stops all outstanding transactions
  - Tears down connections
  - Resets the chip and re-downloads the firmware to return to a working state
- The SSR framework is notified that a firmware reset occurred

Data Path Flows for QCA6174 (Architecture) (1 of 2)



Data Path Flows for QCA6174 (Architecture) (2 of 2)

- OS Shim: Provides abstract interaction with OS

- Tx/Rx: Performs protocol data processing

- HTT: Abstraction of host/target data transfer

- HTC: Virtualization of host/target communication

- HIF: Abstraction of physical host/target communication

Data Path Flows for QCA6174 (OS Shim)

- Purpose – Provides abstract interaction with OS
- API – Tx, Rx, monitor
- Data types – adf_nbuf
- Do DMA mapping for tx frames
- If required, provides a pool of pre-mapped Rx and management netbufs
- Invokes TSO segmentation

Data Path Flows for QCA6174 (HTT)

- Purpose – Abstraction of host/target data transfer
- Tx – Add tx metadata needed by target (HTT Tx desc)
- Rx – Descriptor abstraction
- Rx – MAC DMA rx ring
- Host ←→ target data-related messages

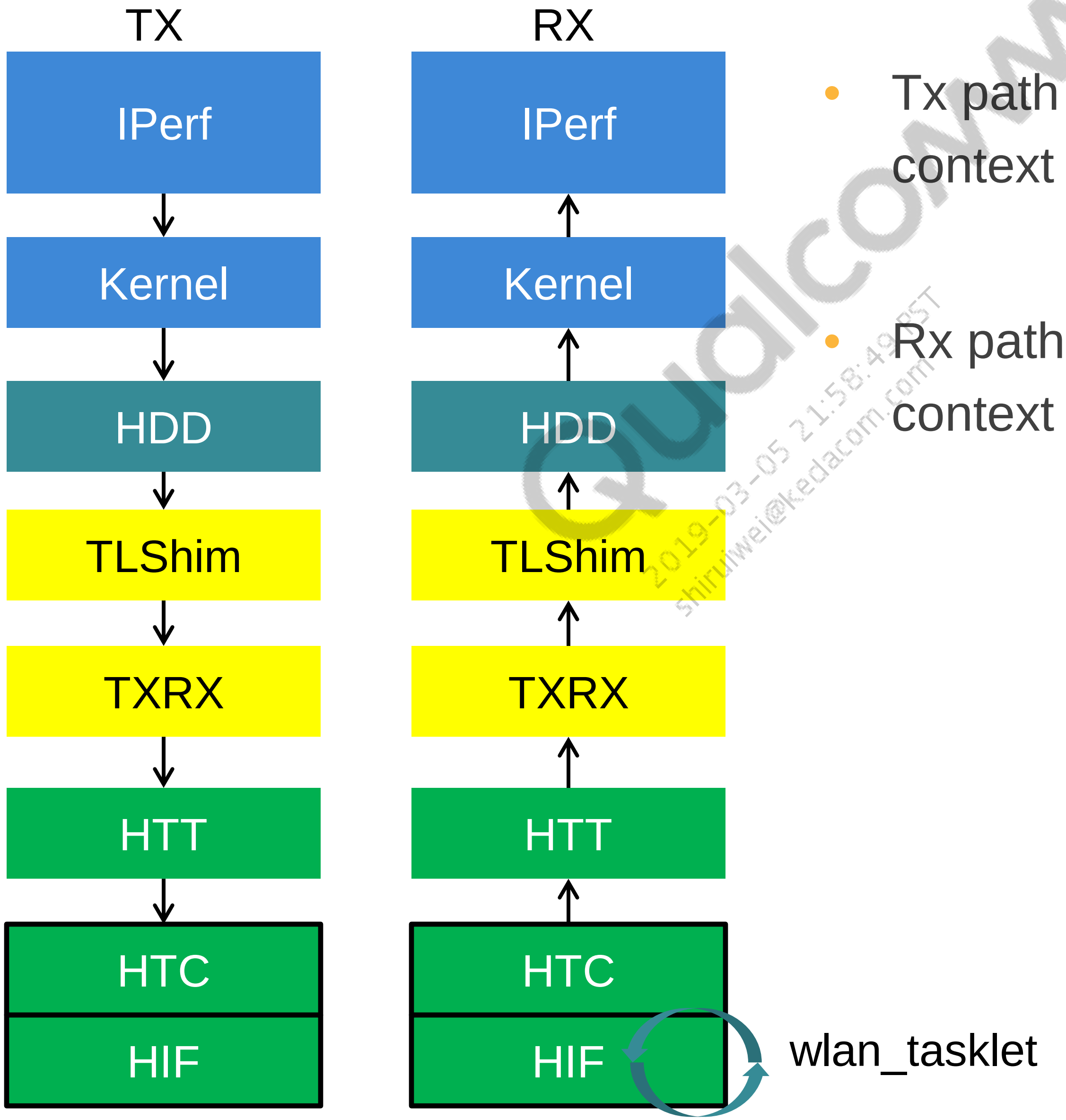
Data Path Flows for QCA6174 (HTC)

- Purpose – Virtualization of host / target communication
- Stores frames until the underlying HIF layer can accept them
- Maps physical “pipes” to virtual “endpoints”
- Provides flow control between endpoints

Data Path Flows for QCA6174 (HIF)

- Purpose – Abstraction of physical host/target communication
- Sends data to target
- Invokes callback when the send is completed
- Invokes callback when data is received from the target

TX RX Context



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

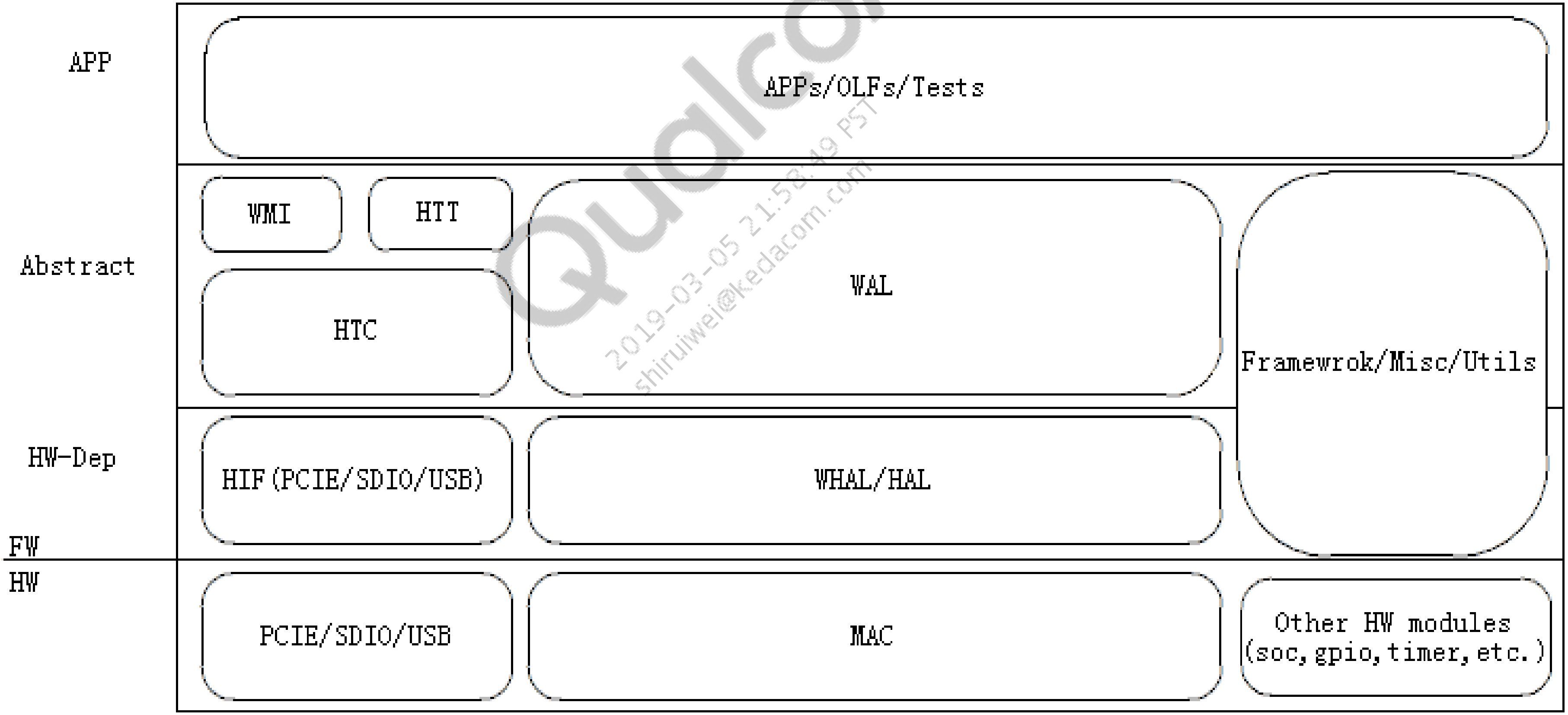
- Rx path runs in the HIF Rx tasklet context (wlan_tasklet)

Background: Data Path Offloads in QCA6174 PCIe System

- Tx
  - Host creates only an MSDU tx descriptor
  - Target determines the recipient + traffic type, and stores the Tx descriptor in the relevant queue
  - Target decides which Tx queue to transmit from, and how much data from the queue to aggregate
- Rx
  - Target tells the host when Rx frames are present in order
  - Host does PN/replay check (optionally could be done by target instead)
  - Host does Rx → Tx forwarding check

Firmware

QCA6174 Firmware Architecture



Three Layers

- App layer (offloaded functions (OLFs), apps, tests)
  - OLFs – chatter, offload_mgr, p2p, etc.
  - Apps – eeprom_console, utf, etc.
  - Tests – endpoint_ping, otp_test, htt_wal_rx, etc.
- Abstract layer (MAC service, interface service)
  - NIC - Data transfer task, MAC ←→ host interfaces
  - Higher level service abstract; make app layer independent of service
  - MAC – WAL (WLAN Abstract Layer) focus in this document
  - Interface – WMI/HTT/HTC
- HW-dependent layer (WHAL/HAL/HIF)
  - Abstract MAC and interfaces (pcie, sdio, usb); make upper layer independent of HW
  - WHAL
  - HIF

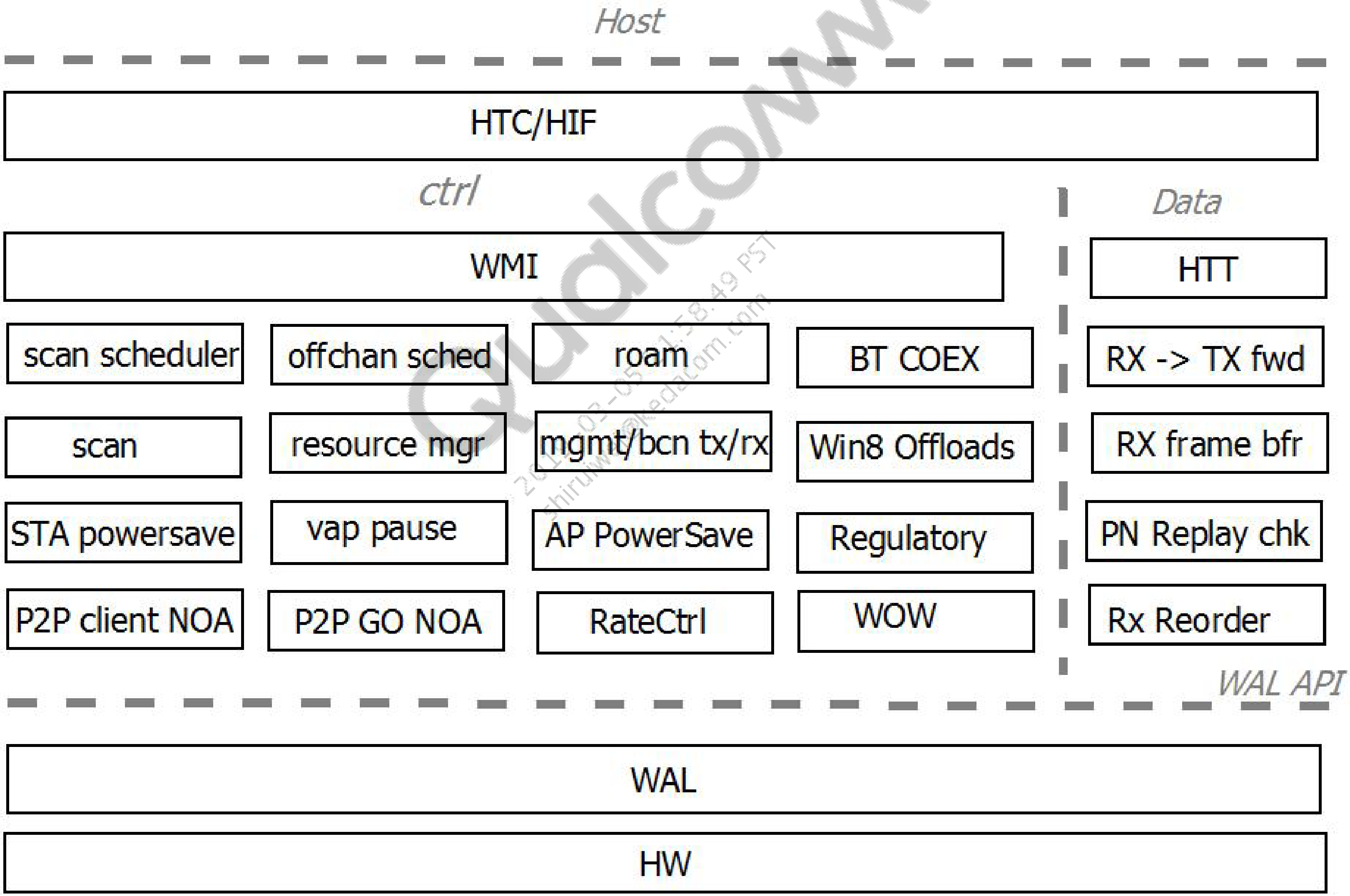
App Layer

- There are many kind of app modules, such as OLF, App and test function
- We focus on OLF in this layer

OLF

- Functions that are offloaded from the host for power/ performance reasons
- Performance offloads
  - Frame Forwarding
  - Rx reorder
- Power offloads
  - STA/AP powersave
  - P2P GO/client NOA
  - Chatter
  - Roaming

OLF (Offload Stack Functions)



Key OLF Modules

- Scan
  - Implements a scan state machine; exposes very flexible API to support several types of scans
    - User-requested scans directed and undirected
    - P2P device scans
    - CCX scan
    - All passive scans
    - Background scans
- Scan scheduler
  - Accepts scan requests from multiple modules
  - Queues and prioritizes scan requests,then issues them one at a time to the scan module
  - Allows prioritization to be configurable

OLF Modules (1 of 3)

- Resource Manager
  - Manages the channel
  - Arbitrates channel between multiple requestors (VDEV,scan ..etc) Resolves conflicting channel requests from multiple requestors
  - Switches channel from one requestor to another by coordinating them
    - For example, when resmgr moves channel from vdev(s) to scanner, it asks each vdev to perform a pause operation (a STA vdev enters network sleep part of pause) before taking the channel away from vdev(s)
  - Invokes off-channel scheduler for multi-channel operation

OLF Modules (2 of 3)

- Roam
  - Performs background scanning; the conditions to trigger a BG scan are configurable from the host (rssi threshold, periodicity ..etc)
  - Invokes the host MLME via a WMI event to roam to a new AP only if a better AP is found, i.e., Soft Roam
  - If beacon misses, invokes the host MLME with the candidate AP list
  - Supports two modes for indicating BG scan results to host
    - Mode 1 – During the background scan, if a beacon/probe response is received from a better AP, sends the whole beacon/probe response to host
    - Mode 2 – Collects all the candidate Aps and sends up the list at the end of the BG scan

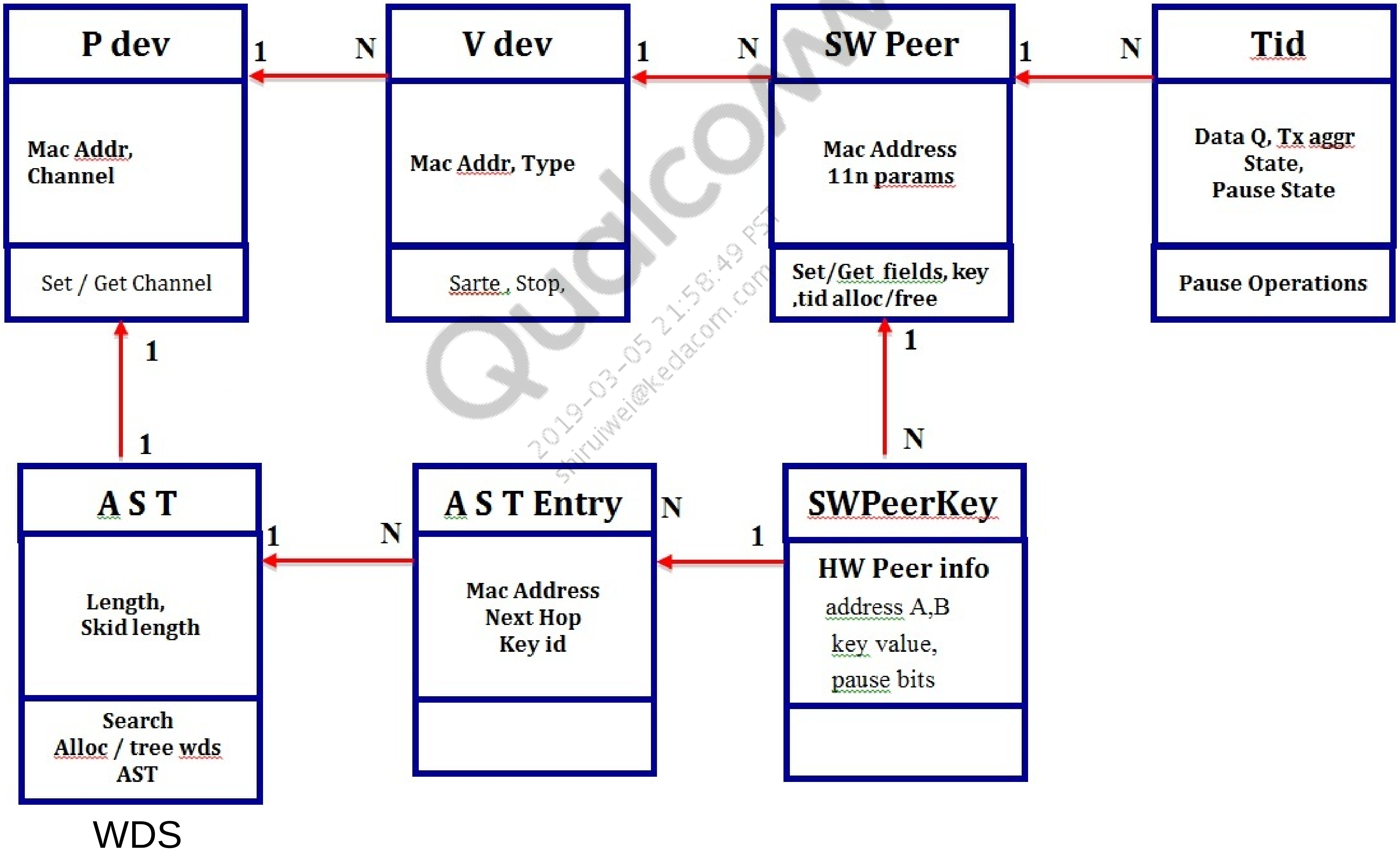
OLF Modules (3 of 3)

- Management and Beacon Tx/Rx
  - Most of the management frames are generated on the host
  - The management frames can be sent to the target either by reference or by value
    Send by reference is only valid for an LL (low latency) interface like pcie
  - Supports two modes for beacon and probe response transmission
    - Host mode – All the beacons and probe responses are generated by host mlme; they can be sent to the target by reference for LL interface; the FW sends periodic SWBA events to the host
    - Firmware mode – The beacons and probe responses are generated by FW; the host supplies all the required beacon content to the FW; host updates the content whenever it is changed
    - In both the modes the content of beacons/probe response is always generated by the host
    - In both the modes the power save-specific Ies (like TIM, p2p GO NOA ..) in beacons/probe responses are generated by the firmware; in Host mode, the FW sends these along with the SWBA event
  - Received beacons are sent up to the host based on beacon filter criteria set up by the host

WAL Services

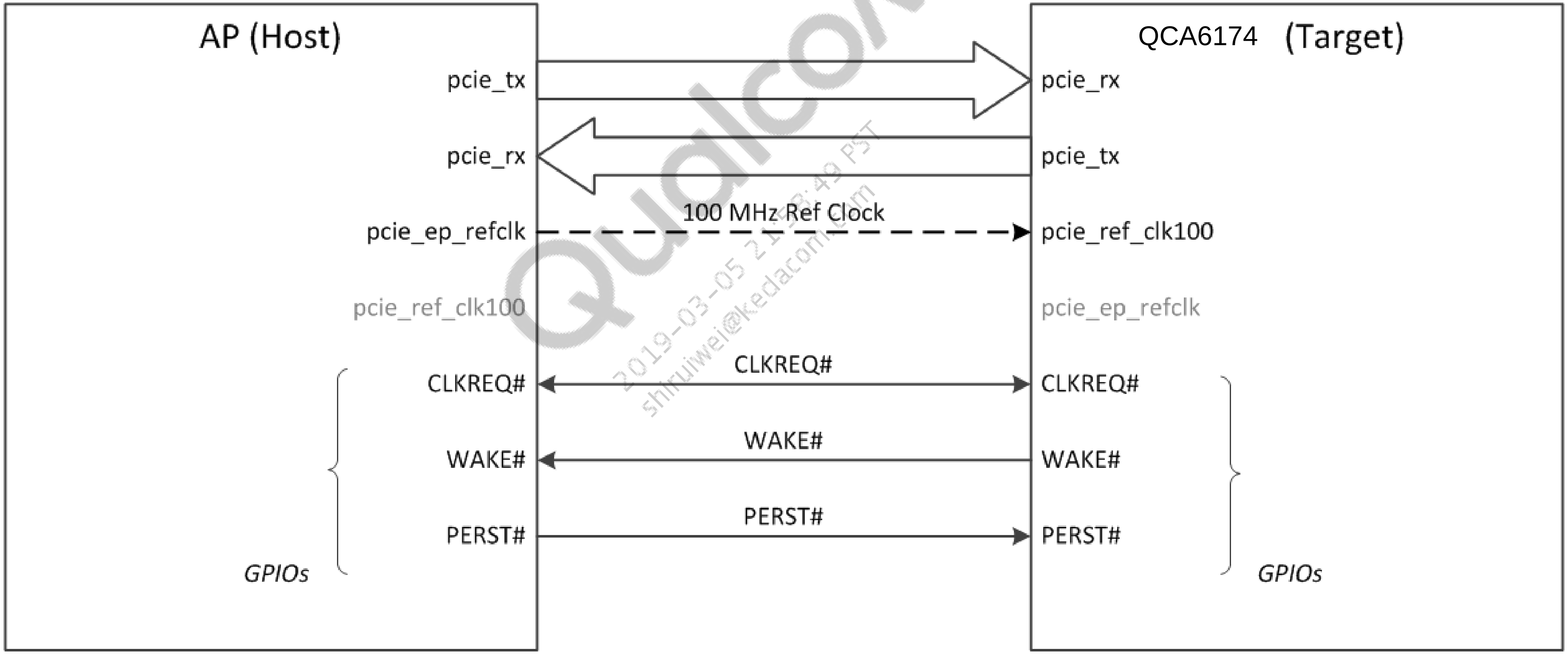
- Provides a set of services through the following WAL objects
  - Pdev – Represents and provides services for a physical radio
  - Vdev – Represents and provides services for a virtual 802.11 mac (virtual interfaces)
  - Peer – Represents and provides services for a connected peer device (analogous to conn_t on olca and node on newma)
  - TID Queue – Represents and provides a data queue for a given qos tid with in a peer

WAL Objects

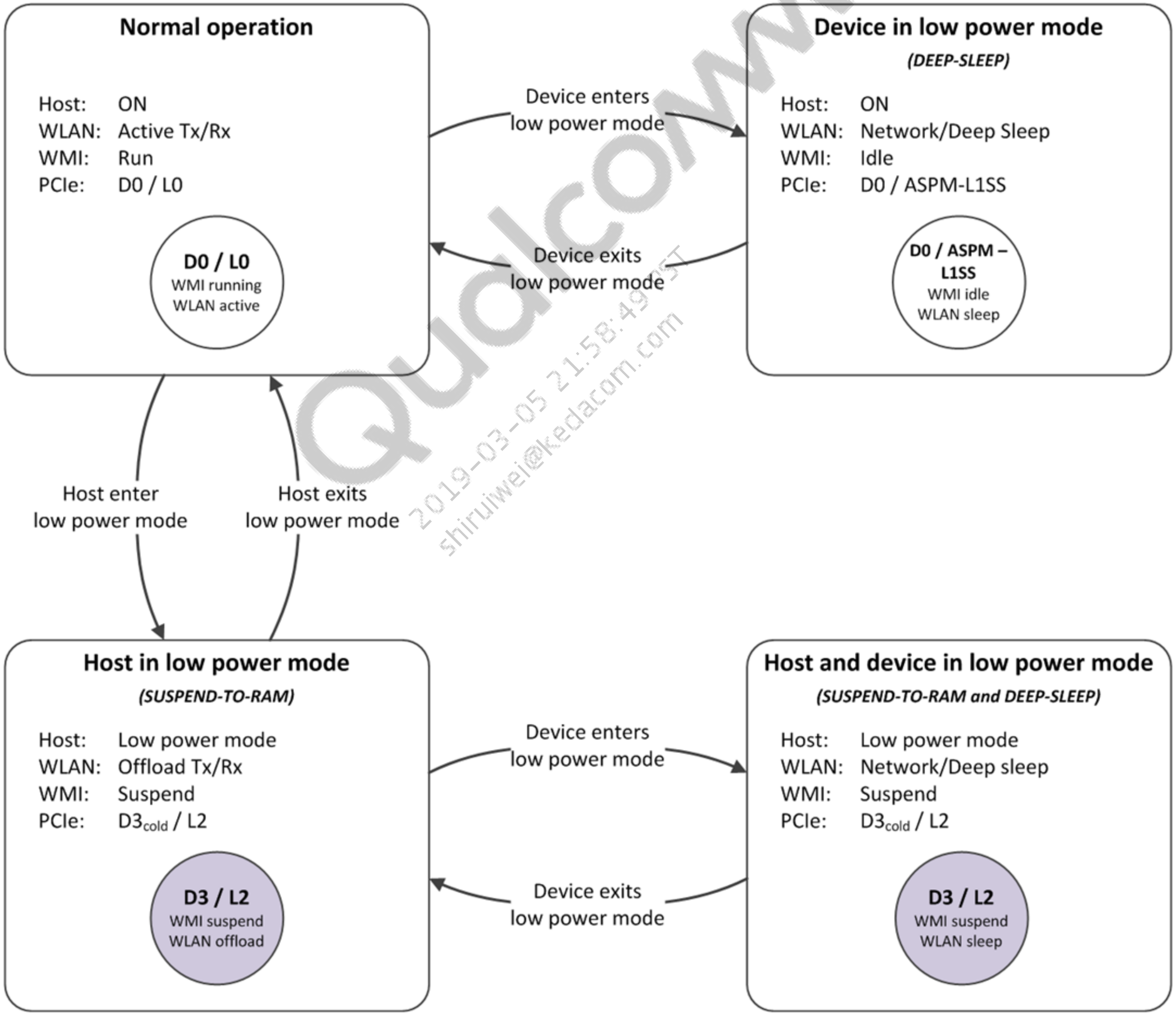

P dev
Mac Addr,
Channel
Set / Get Channel
1
N
V dev
Mac Addr, Type
Sarte , Stop,
1
N
SW Peer
Mac Address
11n params
Set/Get fields, key
,tid alloc/free
1
N
Tid
Data Q, Tx aggr
State,
Pause State
Pause Operations
1
1
1
N
A S T
Length,
Skid length
Search
Alloc / tree wds
AST
1
N
A S T Entry
Mac Address
Next Hop
Key id
N
1
SWPeerKey
HW Peer info
address A,B
key value,
pause bits
WDS

PCIe

Host/QCA6174 PCIe Inter-Connect



Host/ QCA6174 System Low Power Mode



Normal operation

Host: ON
WLAN: Active Tx/Rx
WMI: Run
PCIe: D0 / L0

D0 / L0
WMI running
WLAN active

Device enters low power mode

Device exits low power mode

Device in low power mode

(DEEP-SLEEP)

Host: ON
WLAN: Network/Deep Sleep
WMI: Idle
PCIe: D0 / ASPM-L1SS

D0 / ASPM – L1SS
WMI idle
WLAN sleep

Host enter low power mode

Host exits low power mode

Host in low power mode

(SUSPEND-TO-RAM)

Host: Low power mode
WLAN: Offload Tx/Rx
WMI: Suspend
PCIe: $D3_{cold}$ / L2

D3 / L2
WMI suspend
WLAN offload

Device enters low power mode

Device exits low power mode

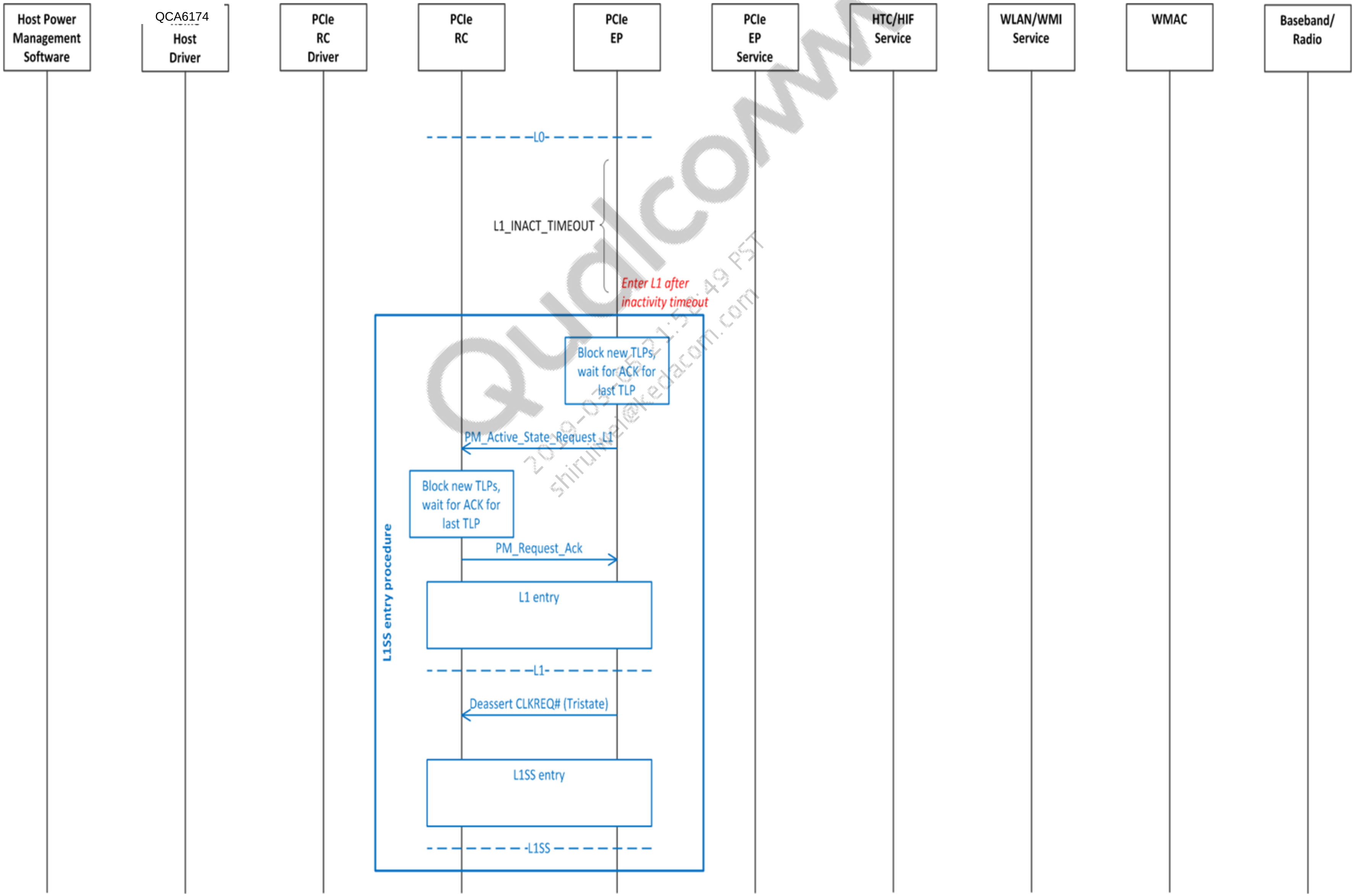
Host and device in low power mode

(SUSPEND-TO-RAM and DEEP-SLEEP)

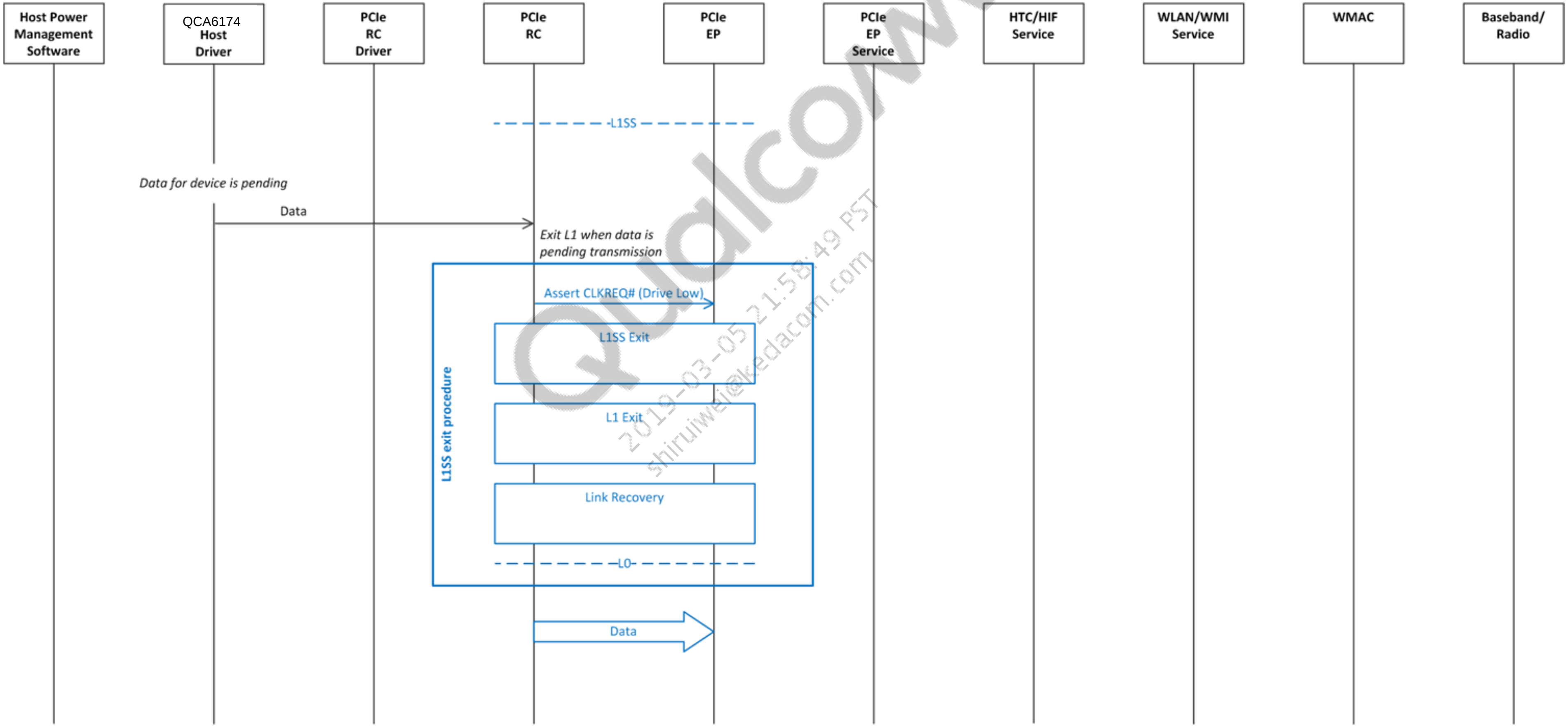
Host: Low power mode
WLAN: Network/Deep sleep
WMI: Suspend
PCIe: $D3_{cold}$ / L2

D3 / L2
WMI suspend
WLAN sleep

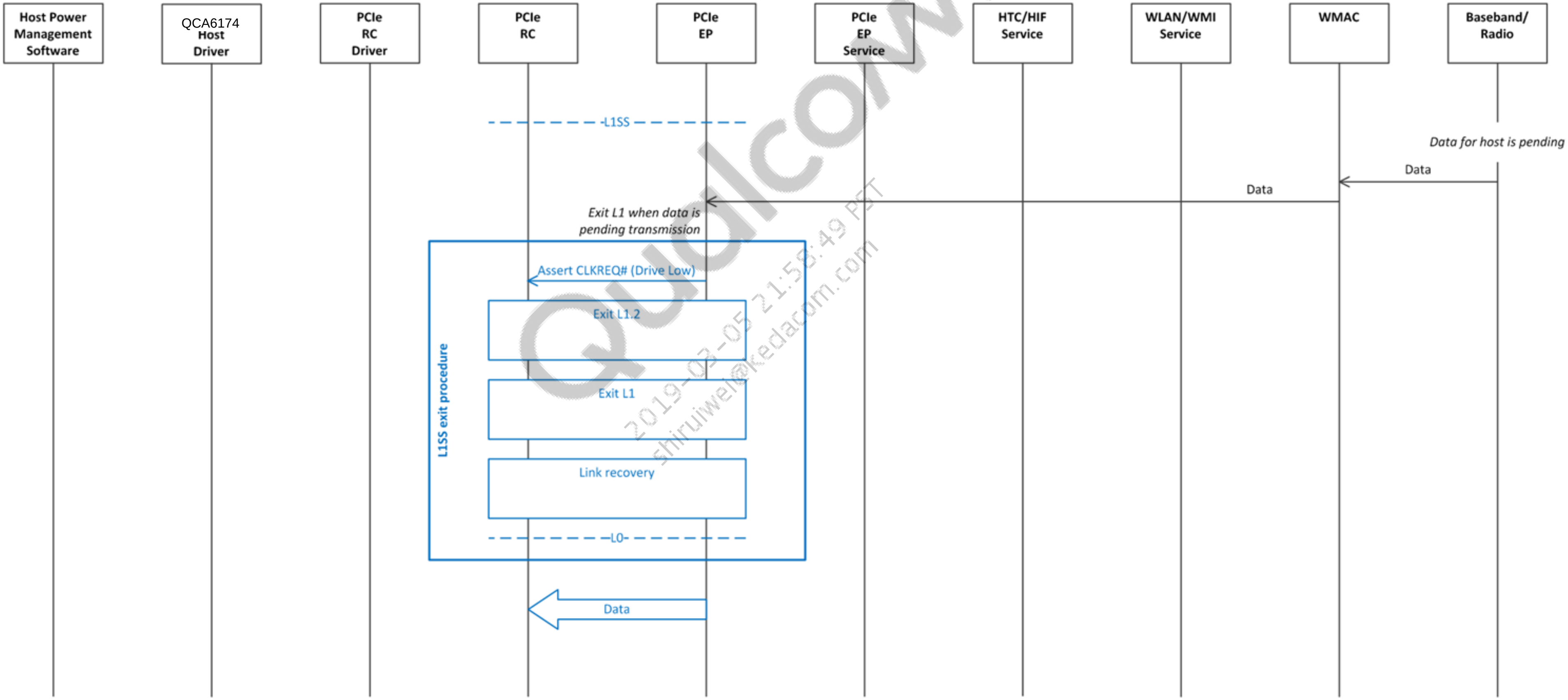
D0-L1SS Entry


Host Power Management Software
QCA6174 Host Driver
PCIe RC Driver
PCIe RC
PCIe EP
PCIe EP Service
HTC/HIF Service
WLAN/WMI Service
WMAC
Baseband/ Radio
-L0-
L1_INACT_TIMEOUT
Enter L1 after inactivity timeout
L1SS entry procedure
Block new TLPs, wait for ACK for last TLP
PM_Active_State_Request_L1
Block new TLPs, wait for ACK for last TLP
PM_Request_Ack
L1 entry
-L1-
Deassert CLKREQ# (Tristate)
L1SS entry
-L1SS-

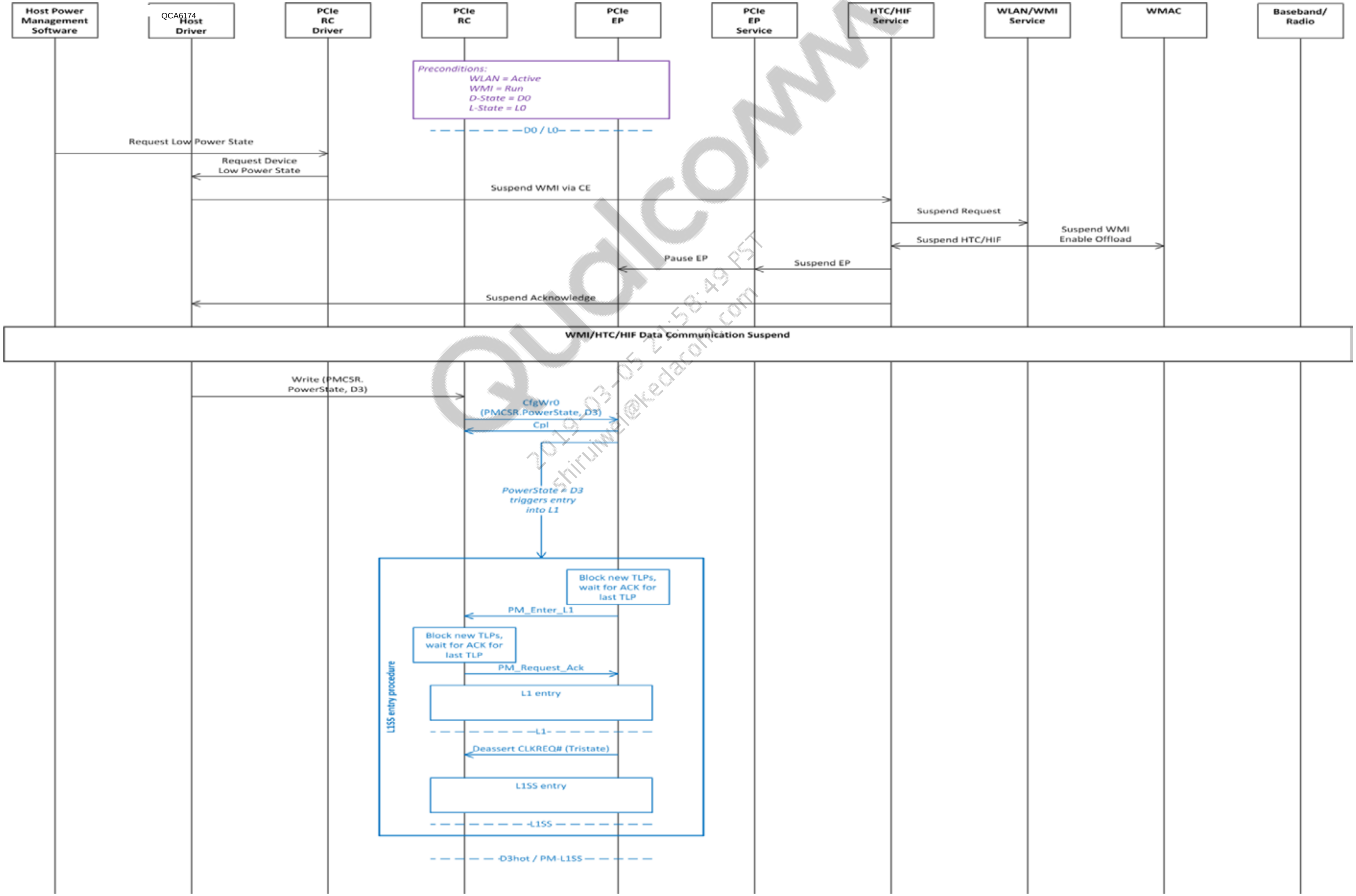
D0-L1SS Exit – Host-Initiated



D0-L1SS Exit – Device-Initiated



D3hot Entry



Host Power Management Software

QCA6174 Host Driver

PCIe RC Driver

PCIe RC

PCIe EP

PCIe EP Service

HTC/HIF Service

WLAN/WMI Service

WMAC

Baseband/ Radio

Preconditions:
WLAN = Active
WMI = Run
D-State = D0
L-State = L0

D0 / L0

Request Low Power State

Request Device Low Power State

Suspend WMI via CE

Suspend Request

Suspend WMI Enable Offload

Suspend HTC/HIF

Pause EP

Suspend EP

Suspend Acknowledge

WMI/HTC/HIF Data Communication Suspend

Write (PMCSR. PowerState, D3)

CfgWr0 (PMCSR.PowerState, D3)

Cpl

PowerState = D3 triggers entry into L1

L1SS entry procedure

Block new TLPs, wait for ACK for last TLP

PM_Enter_L1

Block new TLPs, wait for ACK for last TLP

PM_Request_Ack

L1 entry

L1

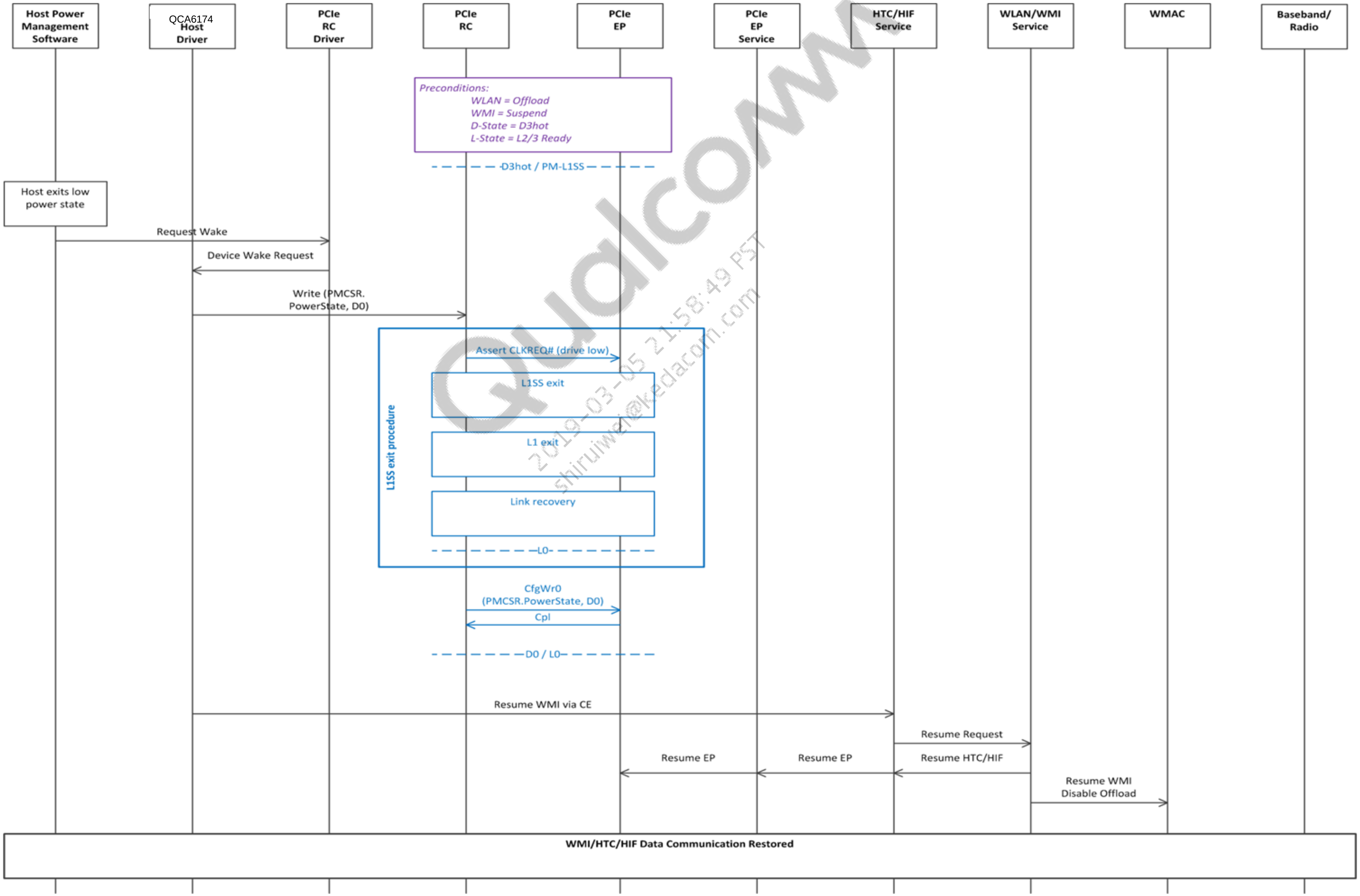
Deassert CLKREQ# (Tristate)

L1SS entry

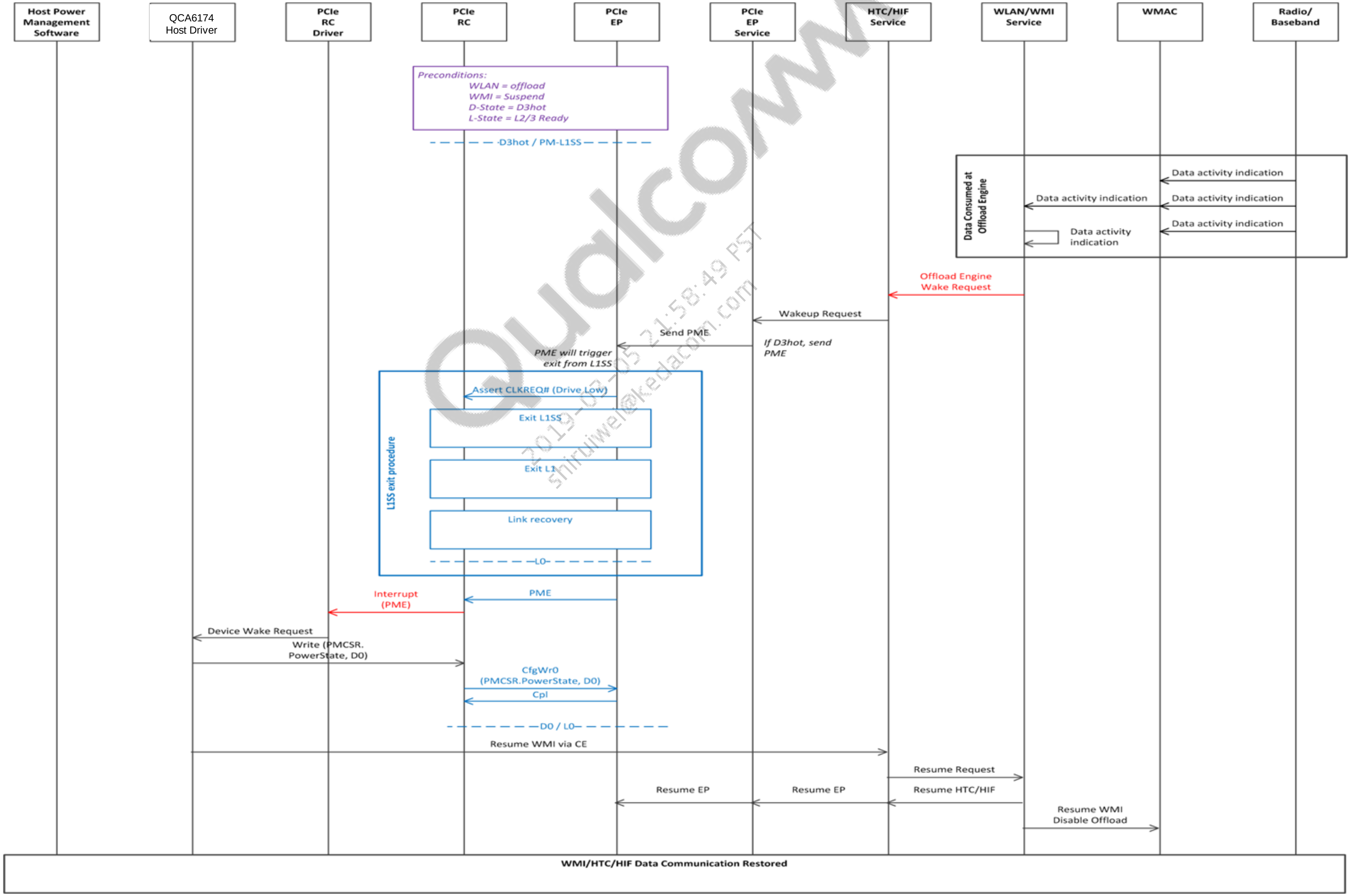
L1SS

D3hot / PM-L1SS

D3hot Exit – Host-Initiated



D3hot Exit – Device-Initiated



Host Power Management Software

QCA6174 Host Driver

PCIe RC Driver

PCIe RC

PCIe EP

PCIe EP Service

HTC/HIF Service

WLAN/WMI Service

WMAC

Radio/ Baseband

Preconditions:
WLAN = offload
WMI = Suspend
D-State = D3hot
L-State = L2/3 Ready

D3hot / PM-L1SS

Data Consumed at Offload Engine

Data activity indication

Data activity indication

Data activity indication

Data activity indication

Data activity indication

Offload Engine Wake Request

Wakeup Request

Send PME

If D3hot, send PME

PME will trigger exit from L1SS

L1SS exit procedure

Assert CLKREQ# (Drive Low)

Exit L1SS

Exit L1

Link recovery

L0

PME

Interrupt (PME)

Device Wake Request

Write (PMCSR. PowerState, D0)

CfgWr0 (PMCSR.PowerState, D0)

Cpl

D0 / L0

Resume WMI via CE

Resume Request

Resume EP

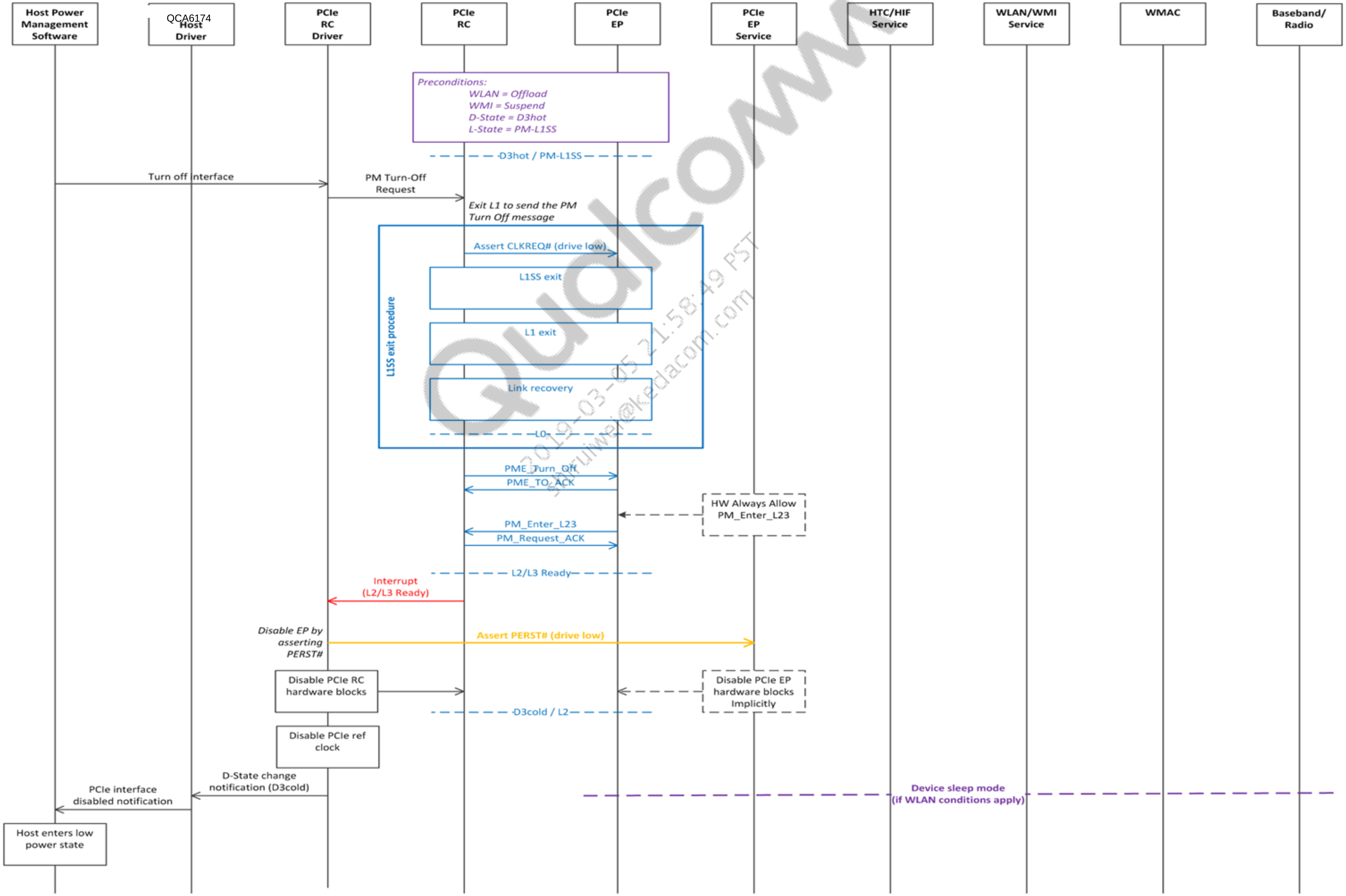
Resume EP

Resume HTC/HIF

Resume WMI Disable Offload

WMI/HTC/HIF Data Communication Restored

D3cold Entry



Host Power Management Software

QCA6174 Host Driver

PCIe RC Driver

PCIe RC

PCIe EP

PCIe EP Service

HTC/HIF Service

WLAN/WMI Service

WMAC

Baseband/ Radio

Preconditions:
WLAN = Offload
WMI = Suspend
D-State = D3hot
L-State = PM-L1SS

D3hot / PM-L1SS

Turn off interface

PM Turn-Off Request

Exit L1 to send the PM Turn Off message

L1SS exit procedure

Assert CLKREQ# (drive low)

L1SS exit

L1 exit

Link recovery

L0

PME_Turn_Off

PME_TO_ACK

HW Always Allow PM_Enter_L23

PM_Enter_L23

PM_Request_ACK

L2/L3 Ready

Interrupt (L2/L3 Ready)

Disable EP by asserting PERST#

Assert PERST# (drive low)

Disable PCIe RC hardware blocks

Disable PCIe EP hardware blocks Implicitly

D3cold / L2

Disable PCIe ref clock

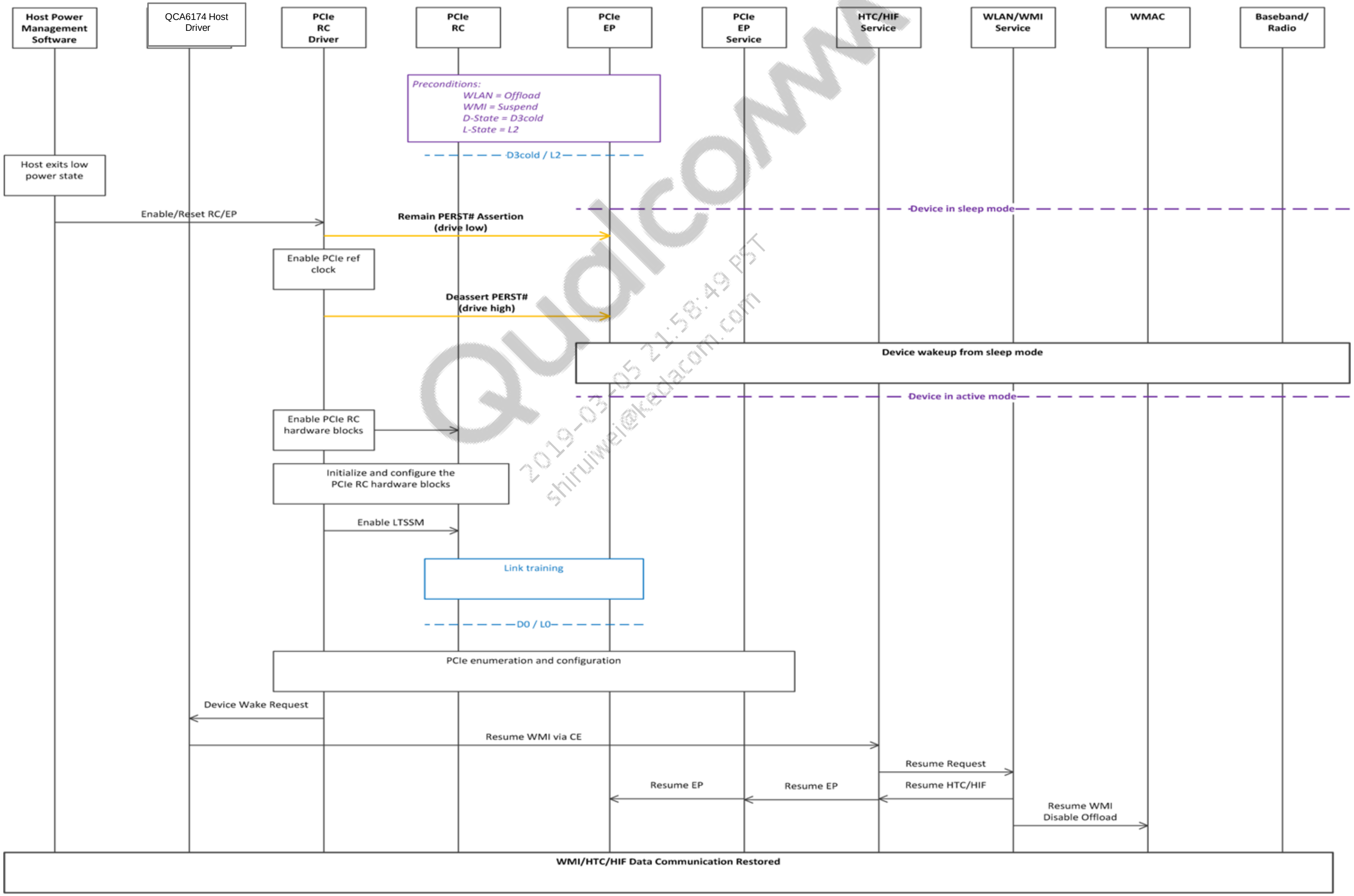
D-State change notification (D3cold)

PCIe interface disabled notification

Device sleep mode (if WLAN conditions apply)

Host enters low power state

D3cold Exit – Host-Initiated



Host Power Management Software

QCA6174 Host Driver

PCIe RC Driver

PCIe RC

PCIe EP

PCIe EP Service

HTC/HIF Service

WLAN/WMI Service

WMAC

Baseband/ Radio

Preconditions:
WLAN = Offload
WMI = Suspend
D-State = D3cold
L-State = L2

D3cold / L2

Host exits low power state

Device in sleep mode

Enable/Reset RC/EP

Remain PERST# Assertion (drive low)

Enable PCIe ref clock

Deassert PERST# (drive high)

Device wakeup from sleep mode

Device in active mode

Enable PCIe RC hardware blocks

Initialize and configure the PCIe RC hardware blocks

Enable LTSSM

Link training

D0 / L0

PCIe enumeration and configuration

Device Wake Request

Resume WMI via CE

Resume Request

Resume EP

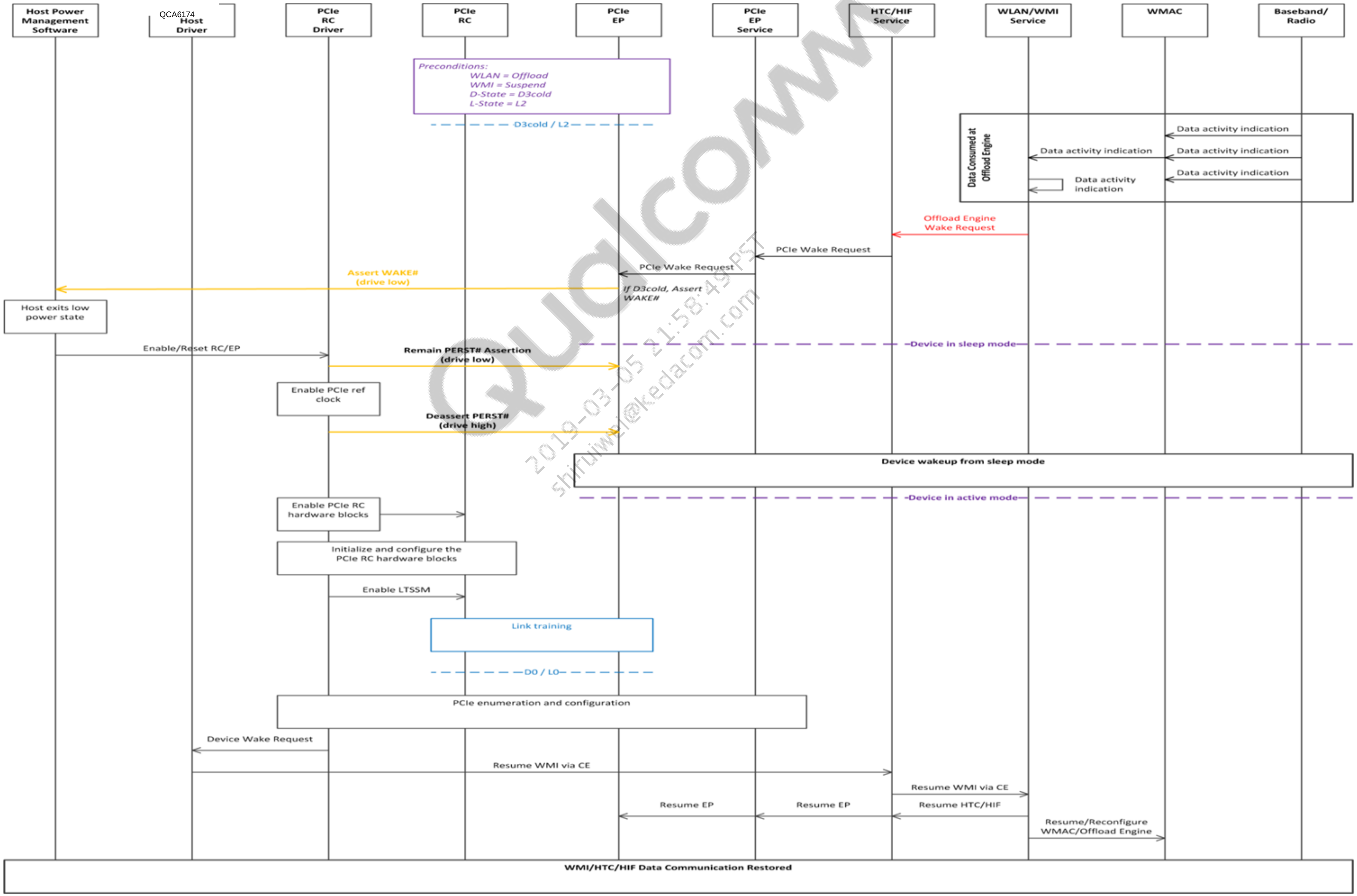
Resume EP

Resume HTC/HIF

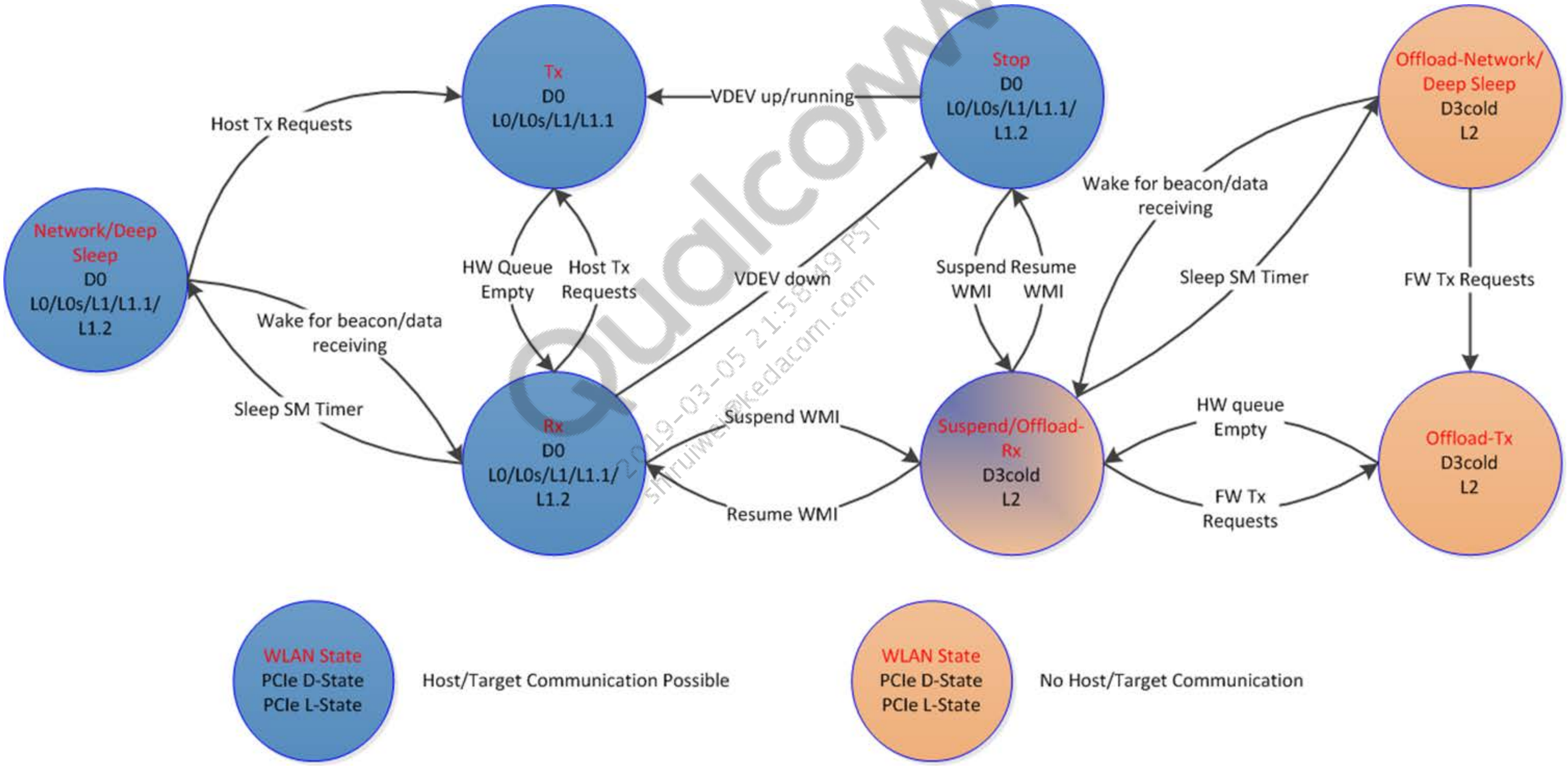
Resume WMI Disable Offload

WMI/HTC/HIF Data Communication Restored

D3cold Exit – Device-Initiated



QCA6174 PCIe EP Service Low Power State

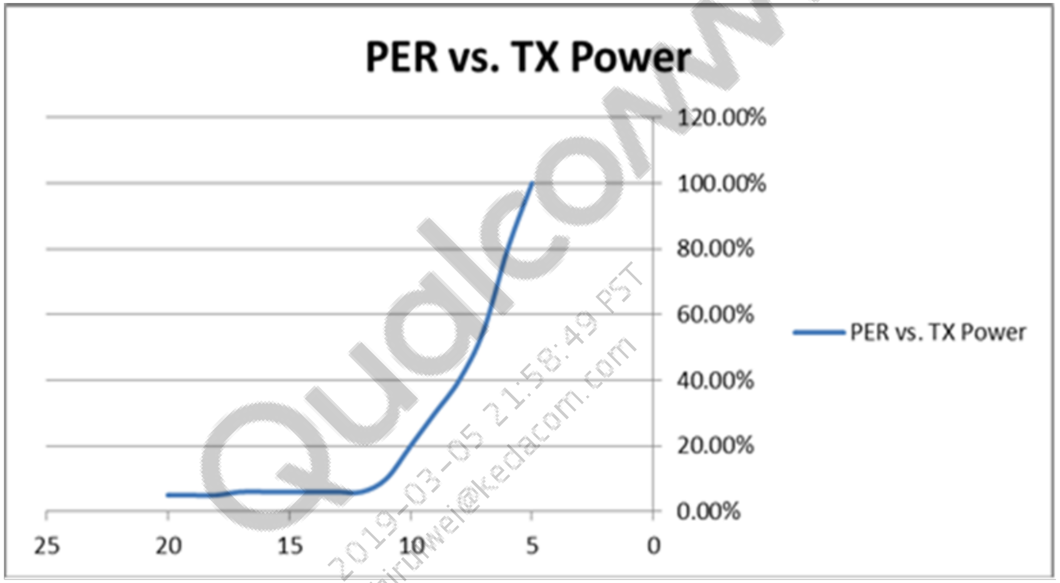


Advanced Features

Green Tx Feature (1 of 2)

- GreenTx is a proprietary QCA output power control algorithm
  - Dynamically adjusts Tx power to ensure the lowest output power is used to maintain the highest PHY rate
  - Algorithm uses PER to determine the lowest output power needed to sustain a link at the highest PHY rate
- Benefitting Use Cases
  - Most router devices are used at close distance; MIMO APs ensure good signal coverage; this allows Tx power to be throttled down, resulting in significant power savings
  - When implemented in mobile routers and client devices, it extends battery life of both devices since they are frequently used within short distances
- Basic algorithm
  - Search from the default Tx power from the OTP/HAL PHY rate table
  - GTX’s principal prerequisite is not impact throughput, so if we can sit in the same rate with the last transmit, and current PER of this rate is < PER cap (3 % currently), we begin to reduce the Tx power by steps
  - If the Tx power is reduced while PER persists, we keep reducing it gradually until it fills in the range of PER cap + margin

Green Tx Feature (2 of 2)

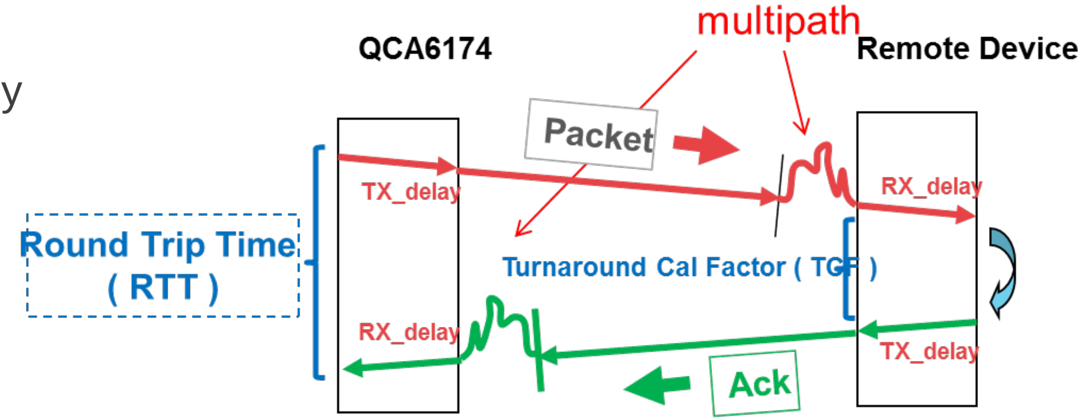


	2G, Single Tone			5G, Single Tone		
	No GreenTx	GreenTx		No GreenTx	GreenTx	
Output power per chain (dBm)	21.4	9.7	1.8	10.6	4.7	0.7
Current consumption per chain(mA)	269	87	69	258	140	128

QCA6174 single chain analog test chip measured data

RTT/Ranging Engine on QCA6174

- Qualcomm's RTT ranging engine, which is integrated with Wi-Fi core across all Qualcomm integrated and discrete Wi-Fi solutions can deliver
  - Precise indoor positioning (3 to 5 m accuracy in all environments - combined with RSSI/indoor maps
  - Accurate P2P ranging – Proximity services, e.g., geo-fencing, relative/collaborative positioning)
- QCA6174 includes significant enhancements to the RTT engine
  - ~50% ranging error improvements compared to WCN3680/8974
    - <2 m ranging error ⇒ ~2 m positioning accuracy @ CEP68%
  - Better multipath mitigation
  - Better measurement consistency
  - Faster response time and lower power consumption
    - For example, hardware-based computations and corrections



Multi-Channel Concurrency (1 of 7)

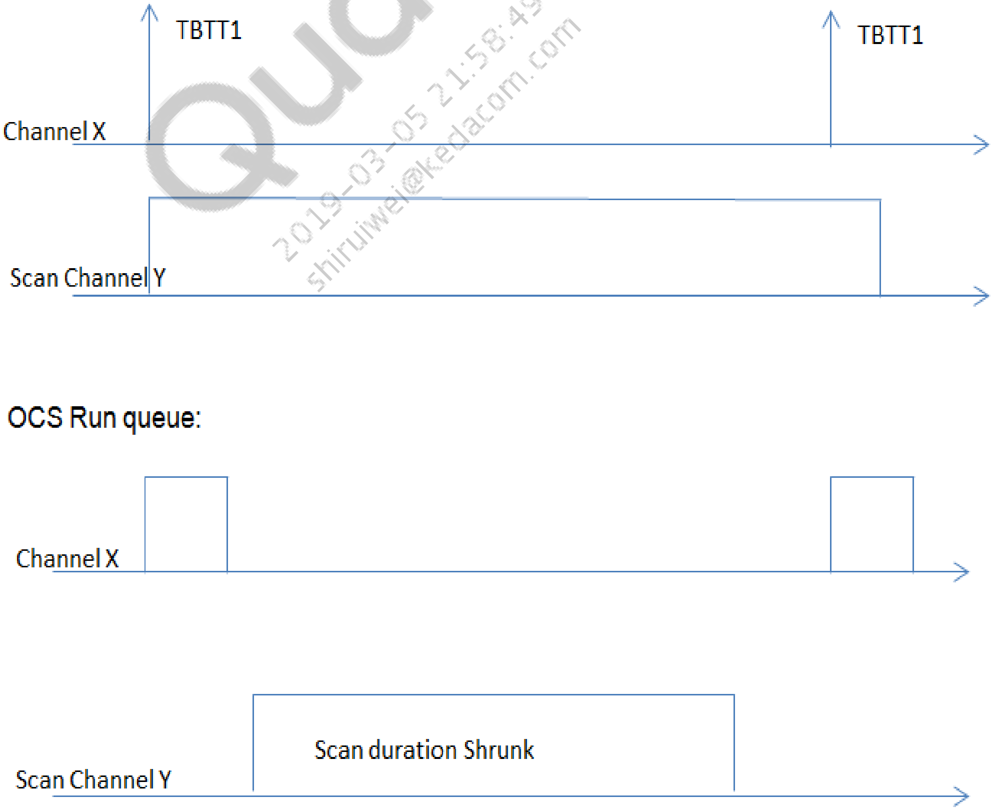
- The following MCC configurations are supported
  - STA + p2p-client
  - STA + p2p-go
  - p2p-go + p2p-client
  - P2P-client+P2P-client

Multi-Channel Concurrency (2 of 7)

- Adaptive multi-channel concurrency algorithm based on
  - Priority
    - Low priority – Background scan, active scan
    - Normal priority – Passive scan
    - High priority – BMISS/Roam scan, low RSSI scan
  - Latency
    - Latency requirement of each virtual interface
    - Ensures that a channel does not remain unallocated for more than its latency value
  - Throughput
    - Throughput requirement of each virtual interface

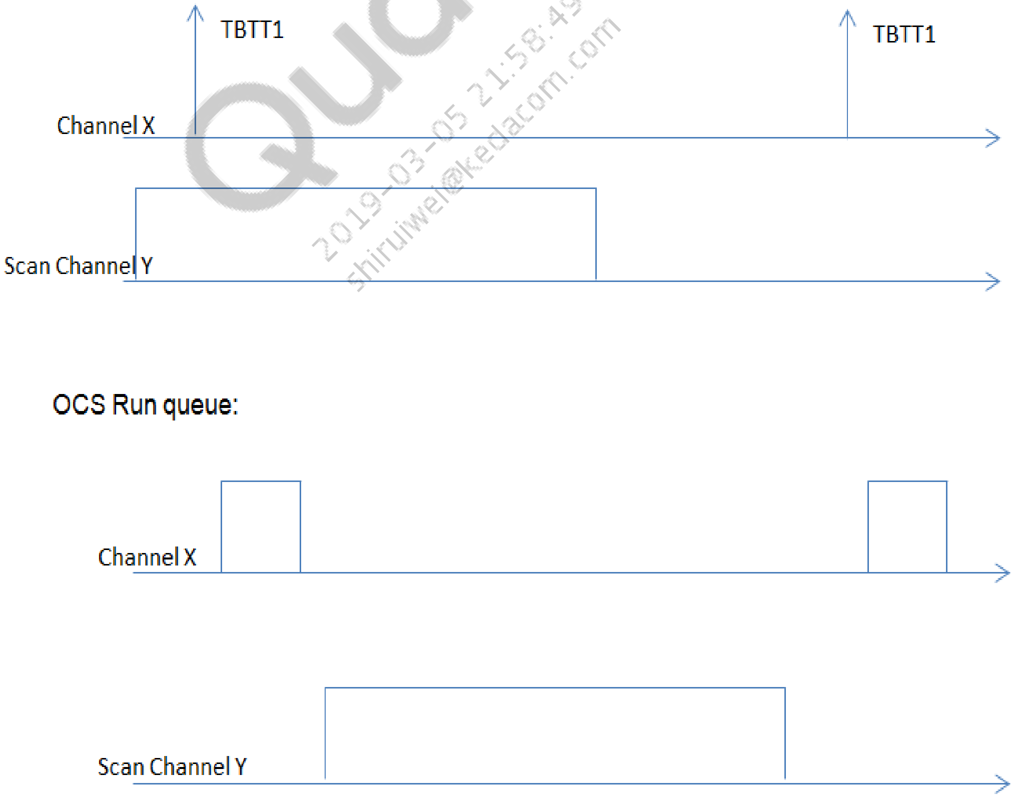
Multi-Channel Concurrency (3 of 7)

- Low priority – Background scan, active scan
  - Non-strict start time, non-strict duration
  - The OCS (Off Channel Scheduler) fits the scan into available slots between the higher priority requests; the scan duration can be shrunk to fit the scans



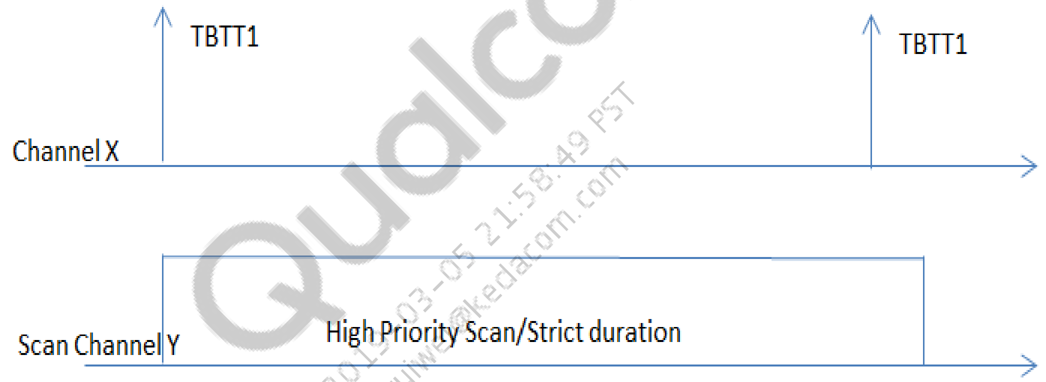
Multi-Channel Concurrency (4 of 7)

- Normal priority – passive scan
  - Non-strict start time, strict duration
  - Strict duration is required for passive scans so that the OCS does not shrink the scan request duration to accommodate the available time slot between two higher priority requests

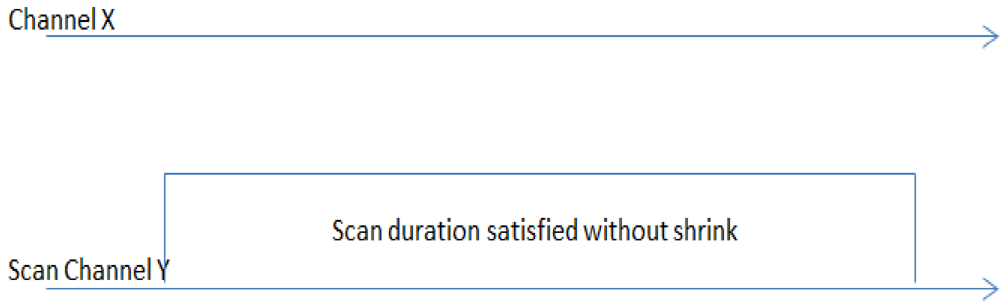


Multi-Channel Concurrency (5 of 7)

- High priority – BMISS/Roam scan, low RSSI scan
  - Strict start time, strict duration



OCS Run queue:



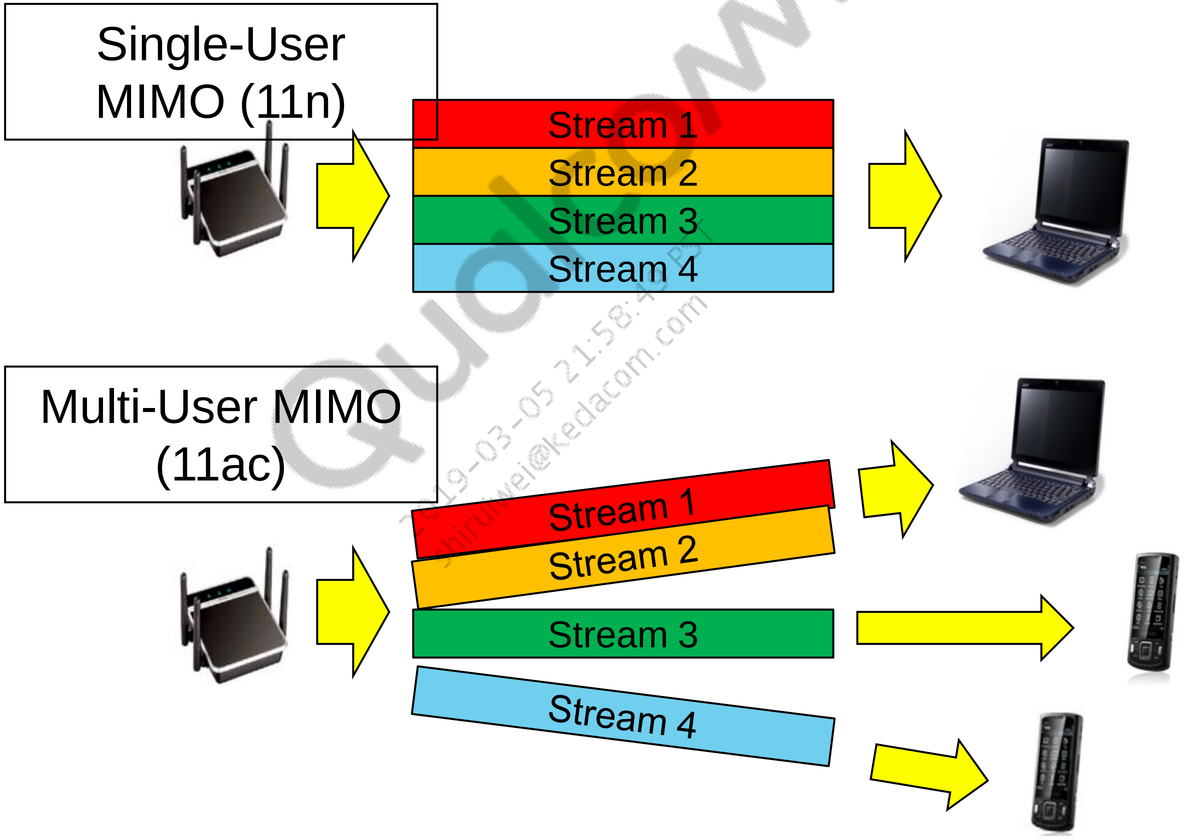
Multi-Channel Concurrency (6 of 7)

- Latency and throughput
- Off channel scheduler needs the following parameters to handle the latency and throughput for each channel
  - Latency requirement of each channel
  - Percentage time quota for the channel
- Quota tracking
  - OCS decides the tracking interval of quota based on the lowest priority request interval on all channels (typically a beacon interval)
  - Minimum interval is chosen and then a multiple of that is used as quota tracking interval
  - A multiple is used to ensure that tracking interval is not too short which could result in more channel switches
  - The total time allocated for a channel = quota in fraction * tracking interval
  - At the start, the quota granted for each channel is set to zero and then OCS keeps adding the total duration of a channel allocation (at each channel switch) to quota granted for that channel
  - Total quota time already granted and quota time allocated together gives the quota time remaining for each channel
- Latency tracking
  - Based on the programmed latency for a channel, the OCS finds the nearest lower factor of the (lowest) low-priority request interval
  - This programmed latency is used in the OCS scheduling decisions to try to ensure that a channel does not remain unallocated for more than its latency value

Multi-Channel Concurrency (7 of 7)

- Scheduling
- The quota and latency requirements are applied only on low-priority requests which are started for data transfer
- After the higher priority request and non-colliding low-priority requests have been resolved and granted, OCS resolves the low-priority requests; OCS simultaneously iterates over run queues for two different channels and finds a winner
- The order of priority for the low-latency logic is as follows (in decreasing priorities)
  - Strict time requests (beacon receive)
  - Latency
  - Quota

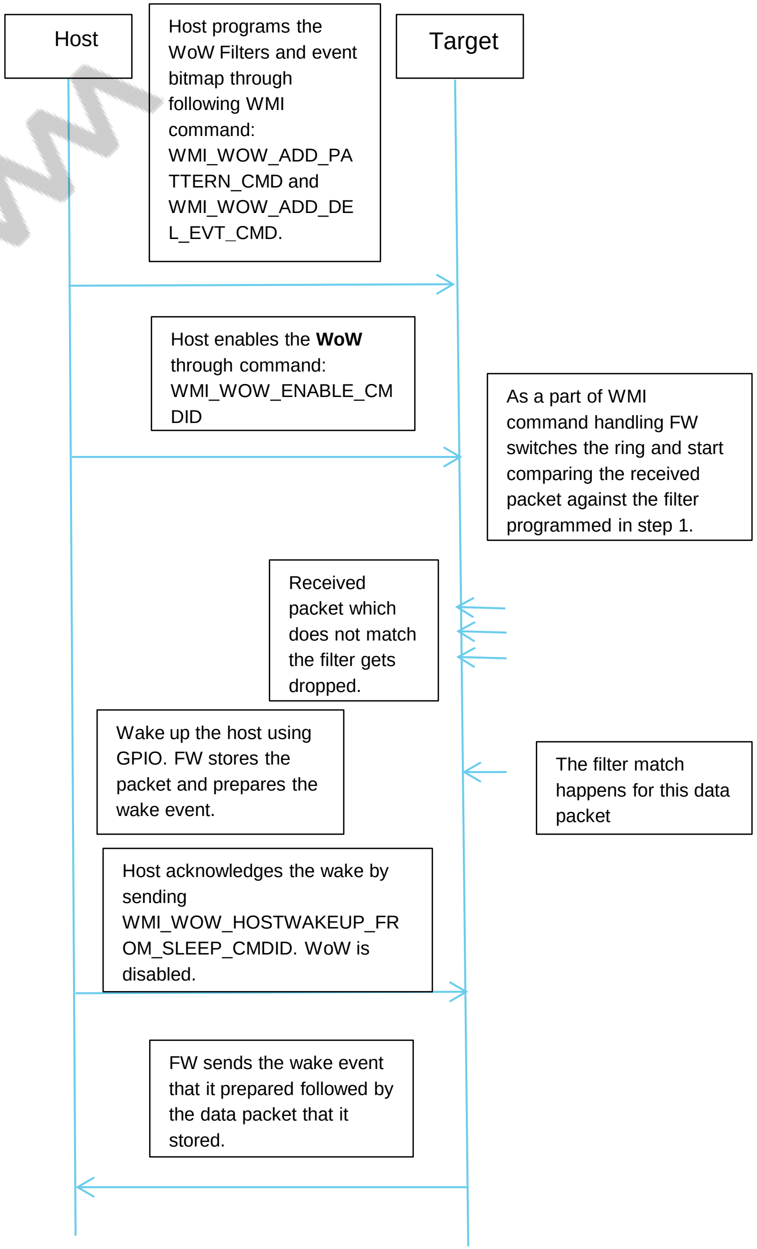
MU-MIMO



- AP transmits to multiple client devices *simultaneously*
- QCA6174 has hardware support for MU-MIMO client

WoW

- Host wakeup during WoW (D3/L2 state) is triggered by
  - Up to 32 possible filters set by host which are matched by firmware before waking host up
  - Various firmware events (de-auth, link-loss, roam)
- Offloading in context of WoW
  - ARP offload
  - GTK offload
- Concurrency mode
  - Magic patterns for multiple virtual interfaces are monitored
  - Packet filters for multiple virtual interfaces are monitored



Host

Target

Host programs the WoW Filters and event bitmap through following WMI command: WMI_WOW_ADD_PATTERN_CMD and WMI_WOW_ADD_DEL_EVT_CMD.

Host enables the **WoW** through command: WMI_WOW_ENABLE_CMDID

As a part of WMI command handling FW switches the ring and start comparing the received packet against the filter programmed in step 1.

Received packet which does not match the filter gets dropped.

Wake up the host using GPIO. FW stores the packet and prepares the wake event.

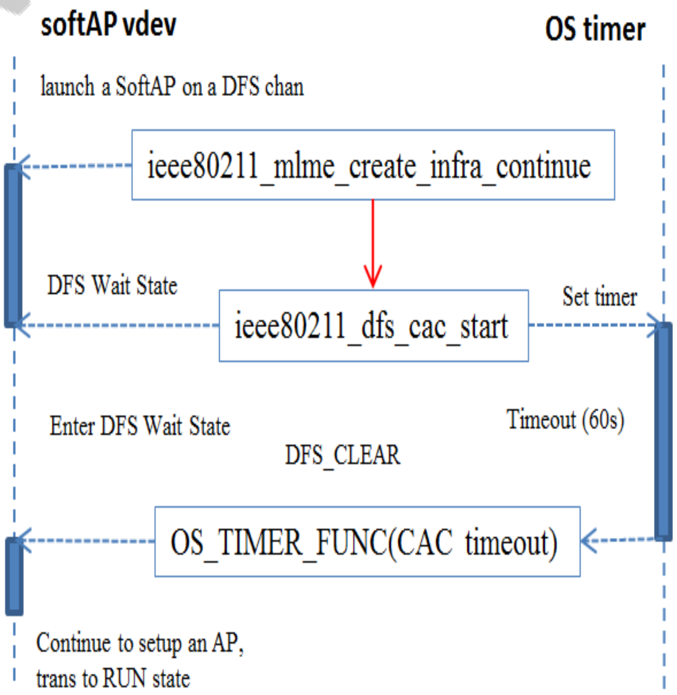
The filter match happens for this data packet

Host acknowledges the wake by sending WMI_WOW_HOSTWAKEUP_FROM_SLEEP_CMDID. WoW is disabled.

FW sends the wake event that it prepared followed by the data packet that it stored.

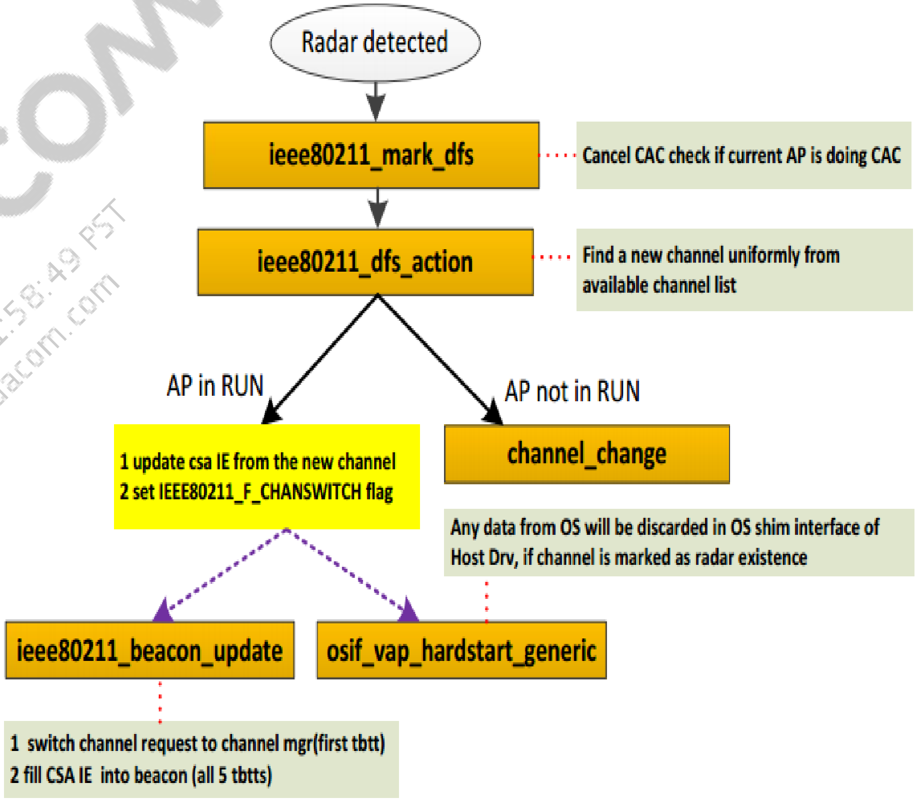
DFS (1 of 2)

- QCA6174 is able to detect radar in the DFS channel under Soft AP mode
- Channel Availability Check (CAC)
  - CAC is a pure software logic that requires AP to wait for 60 sec (10 min for ETSI weather radar) before fully entering the RUN state (start beaconing)
  - If radar is detected during this period, the AP switches to a new channel, and 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, which means we shall not work on this channel during this time
  - After the NOL timeout, the channel is a valid candidate if a new channel should be selected
  - Currently NOL time is 30 min



DFS (2 of 2)

- When radar is detected on a DFS channel, QCA6174 fills the Channel Select Announcement IE in beacon to notify associated STAs
- If QCA6174 is doing a Channel Availability Check, CAC is cancelled because radar is found
- Select a new channel from the available channel list (consider NOL) uniformly
- If AP is not in the RUN state, AP is able to switch channels immediately



Radar detected

ieee80211_mark_dfs

Cancel CAC check if current AP is doing CAC

ieee80211_dfs_action

Find a new channel uniformly from available channel list

AP in RUN

AP not in RUN

1 update csa IE from the new channel
2 set IEEE80211_F_CHANSWITCH flag

channel_change

Any data from OS will be discarded in OS shim interface of Host Drv, if channel is marked as radar existence

ieee80211_beacon_update

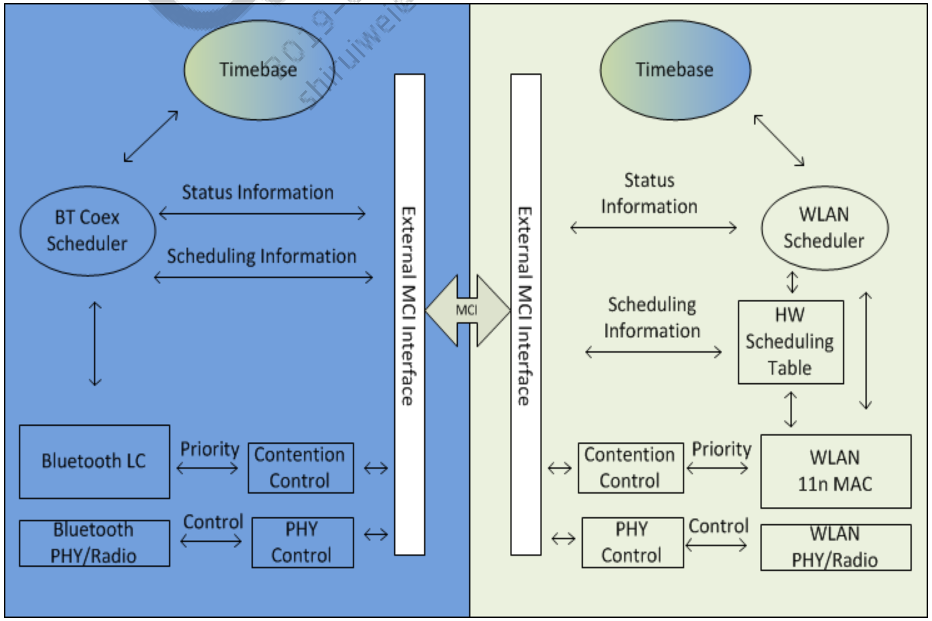
osif_vap_hardstart_generic

1 switch channel request to channel mgr(first tbtt)
2 fill CSA IE into beacon (all 5 tbtts)

Coexistence Architecture on QCA6174

- Message-based coexistence interface
- MCI is a complete coexistence architecture, not just an interface
- MCI is a combination of
  - A message-based coexistence interface
  - Coexistence HW that supports the generation and evaluation of messages
  - Coexistence HW that optimizes frame scheduling and priority setting

- What is MCI?



What are the Benefits of MCI over 3-wire (PTA)?

- WLAN and BT traffic can be better scheduled due to the richer information that is provided over the MCI interface
- MCI allows advance provisioning of BT Tx information
  - 3-wire does not provide BT information in advance leading to suboptimal sizing of A-MPDUs
- Common time reference between WiFi and BT
  - Unlike MCI, in 3-wire PTA, WLAN needs to guess what BT is trying to do
- Does not allow exchanging status information in real time (channel of operation, etc., BT states)
  - Status information needs to be exchanged over the host or sideband (non-real-time)
- PTA supports only 2 BT priorities, MCI supports 256 levels
- MCI is able to distinguish between different BT link types
  - Allows assignment of different BT priorities when multiple profiles are active

New Use Cases with MCI

- Better performance for **multi-profile use cases** because of better scheduling of WLAN and BT
  - Distinction between different HID devices allows for maintaining QoS for each device separately
- BT scheduling information allows **better scheduling of WLAN for MCC**
  - When WLAN gets the information that BT needs the medium, it can switch to 5 GHz and switch back to 2.4 GHz once BT does not need the medium
  - Allows for better combining of free running coexistence with traffic shaping coexistence for **better throughput performance**
  - Free running coexistence does not make use of PS-Poll, NULL frame or CTS to self
  - Priorities distribute the medium only between WLAN and BT
- Avoid prolonged stomping of WLAN (when BT is running inquiry or paging) by prioritizing WLAN traffic

MCI Functionality Details (1 of 3)

- Common time reference
  - WLAN and BT can inform each other of future events
  - WLAN and BT can reference the common time reference to their own internal time reference such as BT ‘s CLKN or WLAN’s TSF timer
- Scheduling information
  - BT sends the expected timing, priority of packet and other information in advance to WLAN
- Contention information
  - BT sends priority information and other information as the BT packet is received/transmitted
  - WLAN can prohibit BT from transmitting
- General purpose information
  - Exchange of status information between BT and WLAN
  - Allows defining of future events
- PHY control information
  - Shared LNA lock information
  - Shared LNA gain information
  - Shared LNA owner information

MCI Functionality Details (2 of 3)

- Interrupt Information
  - Generate an interrupt in the other device
- Sleep / Awake Information
- Scheduling Table
  - Implemented in HW; uses the BT scheduling information
  - Determines if a WLAN transmission is concurrent with BT at some point in the future
  - Determines if WLAN transmission including the received (B-)ACK can be finished by the time BT takes over the medium
  - Allows sizing of an A-MPDU until BT takes the medium
- Adaptive packet priorities
  - WLAN/BT transmit priorities can change depending on the type of transmitted frame
  - WLAN/BT receive priorities can change on the fly depending on the type of received frame
    - Priorities can change depending on the received data, signal strength, packet length, etc.
  - BT priorities can be different for different links
    - Different HID devices can have different priorities
    - Priorities can be changed on a frame-by-frame basis
- Stomp information
  - FW is informed if WLAN prohibited BT from transmitting
  - Contention information is provided

MCI Functionality Details (3 of 3)

- Priority information
  - FW is informed if BT contention information has a specific priority
- Wait for B-ACK
  - Allows transmitting of a B-ACK if a WLAN reception was interrupted by BT and the WLAN A-MPDU was only partially received
  - Allows receiving a B-ACK if a WLAN reception was interrupted by BT and the WLAN A-MPDU was only partially transmitted
- Coexistence transmit statistics
  - WLAN transmission frame errors that are due to BT are counted separately so the rate adaptation is not skewed
  - Statistics are gathered for how much time BT requests the medium
  - Statistics are gathered for how busy the medium is
- Advance PLCP and MPDU header information
  - Provides the PHY rate, RSSI, DA, SA etc. to FW before the complete (A)-MPDU has been received and is independent of the FCS
  - Used for rate fallback detection

Questions?