



**Qualcomm**  
ATHEROS

REDEFINING MOBILITY



## QCA WCN36x0 Software Architecture

80-Y0513-1 Rev. G

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®  
www.hymost.com

**NO PUBLIC DISCLOSURE PERMITTED:** Please report postings of this document on public servers or websites to: [DocCtrlAgent@qualcomm.com](mailto:DocCtrlAgent@qualcomm.com).

**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.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Atheros, Inc.

Qualcomm is a registered trademark of QUALCOMM Incorporated. Atheros is a registered trademark of Qualcomm Atheros, Inc. All other registered and unregistered trademarks are the property of QUALCOMM Incorporated, Qualcomm Atheros, Inc., or their respective owners and used with permission. Registered marks owned by QUALCOMM Incorporated and Qualcomm Atheros, Inc., are registered in the United States of America and may be registered in other countries.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

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

© 2012-2013 Qualcomm Atheros, Inc.

# Revision History

Revision	Date	Description
A	Aug 2012	Initial release
B	Nov 2012	Added WCN8974 SW features; updated Android WCN-SS bootup sequence, host DXE driver data path, WLAN SW high-level architecture, SME interface model, PE components and functionalities, control path execution model, data path execution model; added SoftAP features
C	Dec 2012	Added 11ac / BTC / Hotspot 2.0 updates and the new MSM8974 supported features
D	Feb 2013	Added QCMobileAP API slides
E	Feb 2013	Added WCNSS FW architecture slides
F	March 2013	Title change; added 11ac Rx STBC, TxBF and LDPC
G	March 2013	Updated feature slides

# Table of Contents

- WCN36x0 Overview
- WCN36x0 WLAN/BT/FM Features
- WCN36x0 Hardware Architecture
- WCN-SS Bootup Sequence
- WCN-SS Software Architecture
- CoreBSP – DALSYS/OS Services
- CoreBSP – DAL Interrupt Controller
- CoreBSP – DAL Timers and Time Services
- WLAN/BT Data Flow
- Android WLAN Host Software Architecture
- 802.11ac Software Integration
- Google Jelly Bean (JB) WLAN Updates
- Android Bluetooth Host – BlueZ
- Windows WLAN Host Software Architecture
- Windows WLAN Power Save Feature
- Windows Bluetooth Host
- BT/WLAN Coexistence
- WiFi Direct (P2P)
- P2P Discovery
- P2P Group Formation
- P2P Group Operation
- P2P Software Architecture
- P2P Event/Message Sequence
- SoftAP
- AccessControlLists or MACFiltering
- Restricting the Number of Clients
- Auto-Shutdown Mode
- Energy Detect Mode
- Channel Range and Auto Channel Selection
- 32 STAs support using Virtual STAs
- CLI Commands for Testing
- QCMobileAP APIs
- Hotspot 2.0
- TDLS
- References
- Questions?

# WCN36x0 Overview

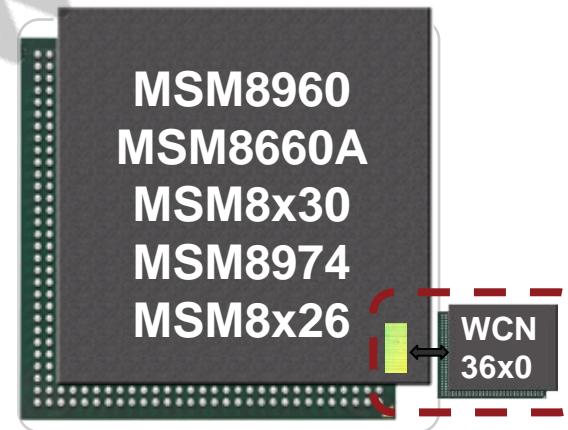
REDEFINING MOBILITY

# Definition/Acronyms

Acronyms	Descriptions
WCN-SS	Wireless Connectivity (WConnect) SubSystem, aka RIVA. It contains all the digital logic for WLAN, BT and FM integrated within the MSM8960 (aka Waverider) device.
cCPU	Connectivity CPU. This is the ARM9 running the WLAN SW on the WConnect Subsystem.
aCPU	Application CPU. This is the Dual-Core processor (Scorpion/Krait) running the WLAN Host SW.
mCPU	Modem Subsystem processor.
CMEM	Connectivity Memory. This is internal SRAM within the WConnect Subsystem. It contains the WLAN HW data structures.
aCPU DDR	Portion of DDR set aside and protected for the Application SS. DXE can move data to and from that memory.
cCPU DDR	Portion of DDR set aside and protected for the cCPU. It contains the WLAN BMU Memory.
Shared DDR	Portion of DDR shared by multiple cores within the MSM™ chipset. The Shared Memory Services are implemented on top of that memory.
RPM	Resource Power Manager. This is a subsystem within the MSM chipset that manages shared resources to optimize power consumption on the MSM chipset. This subsystem will facilitate RIVA power consumption within the MSM chipset.

# WCN36x0 – Third-Generation Connectivity for 28nm MSM™ Chipsets

- BT/FM/WLAN combo RF solution
- Wafer scale package to reduce PCB area
- High-speed, dual-band WLAN solution
  - Ultra-fast wireless data pipe via optimized 433 Mbps WLAN PHY rate and 80 MHz RF bandwidth support
- Integrated BT/WLAN antenna switch
- Optimized LTE BC7, BC38, and BC40 coexistence with 2.4 GHz ISM band (BT/WLAN)
- System memory to replace BT ROM allowing faster new feature additions and enhancements



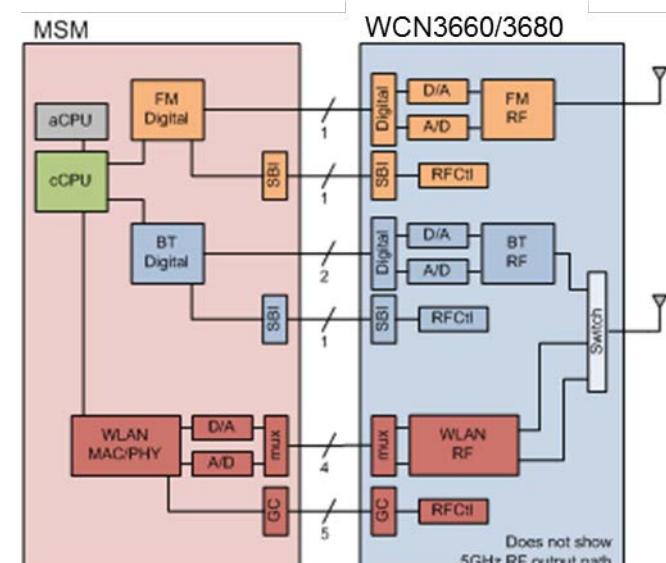
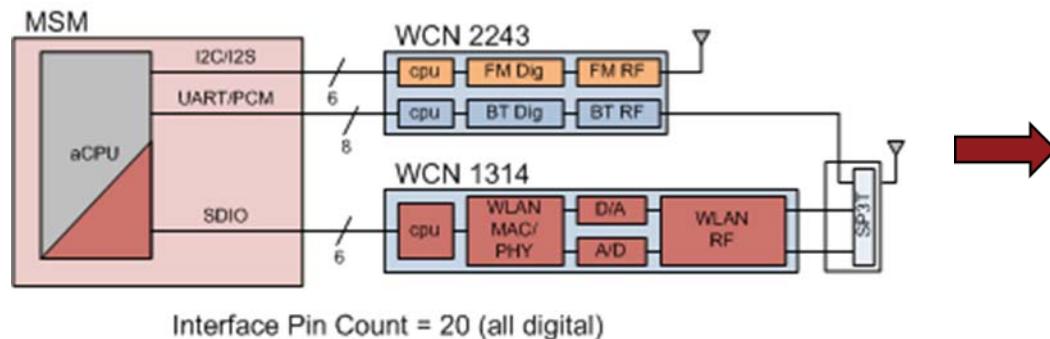
# WCN36x0 – Third-Generation Connectivity for 28nm MSM Chipsets (cont.)

<b>WLAN technology</b>	WCN3660 – 802.11a/b/g/n WCN3680 – 802.11a/b/g/n/ac
<b>BT technology</b>	BT3.x + HS, BT4.0/LE
<b>FM technology</b>	Rx, Tx/RDS, RBDS, RT+
<b>Antenna configuration</b>	Single shared antenna
<b>Interface</b>	High-speed I/F/analog
<b>Package type and size</b>	WLNSP; 14.5 mm <sup>2</sup>
<b>PCB footprint</b>	< 50 mm <sup>2</sup> < 30 external components
<b>WLAN frequency bands</b>	2.4 GHz, 5 GHz
<b>WLAN channel bandwidths</b>	WCN3660 – 20/40 MHz WCN3680 – 20/40/80 MHz
<b>WLAN throughput</b>	WCN3660 – >90 Mbps TCP/IP WCN3680 – >180 Mbps TCP/IP
<b>WLAN power consumption (at V<sub>BATT</sub>)</b>	Rx – 110 mW (20 MHz) Tx – 550 mW (20 MHz, 15 dBm antenna) DTIM=1 – <2 mW
<b>BT Rx sensitivity</b>	-94 dBm GFSK
<b>BT Tx output power</b>	+11 dBm
<b>FM sensitivity</b>	-110 dBm
<b>WLAN max pout (at antenna port)</b>	Up to +19 dBm (11b, 2.4 GHz) Up to +17 dBm (OFDM, 2.4 GHz) Up to +16 dBm (OFDM, 5 GHz)

# WCN36X0 Functional Partitioning

- BT/FM splits the digital processing between MSM and WCN; the digital interface is a 5-pin proprietary interface with data rates <10 Mbps.
- WLAN utilizes an analog differential I/Q interface similar to WAN; it can support high PHY rates –150 Mbps (WCN3660) and 433 Mbps (WCN3680).
- BT/FM/WLAN shares a single processor.
- Software partitioning between the host CPU and WCN CPU remains unchanged.

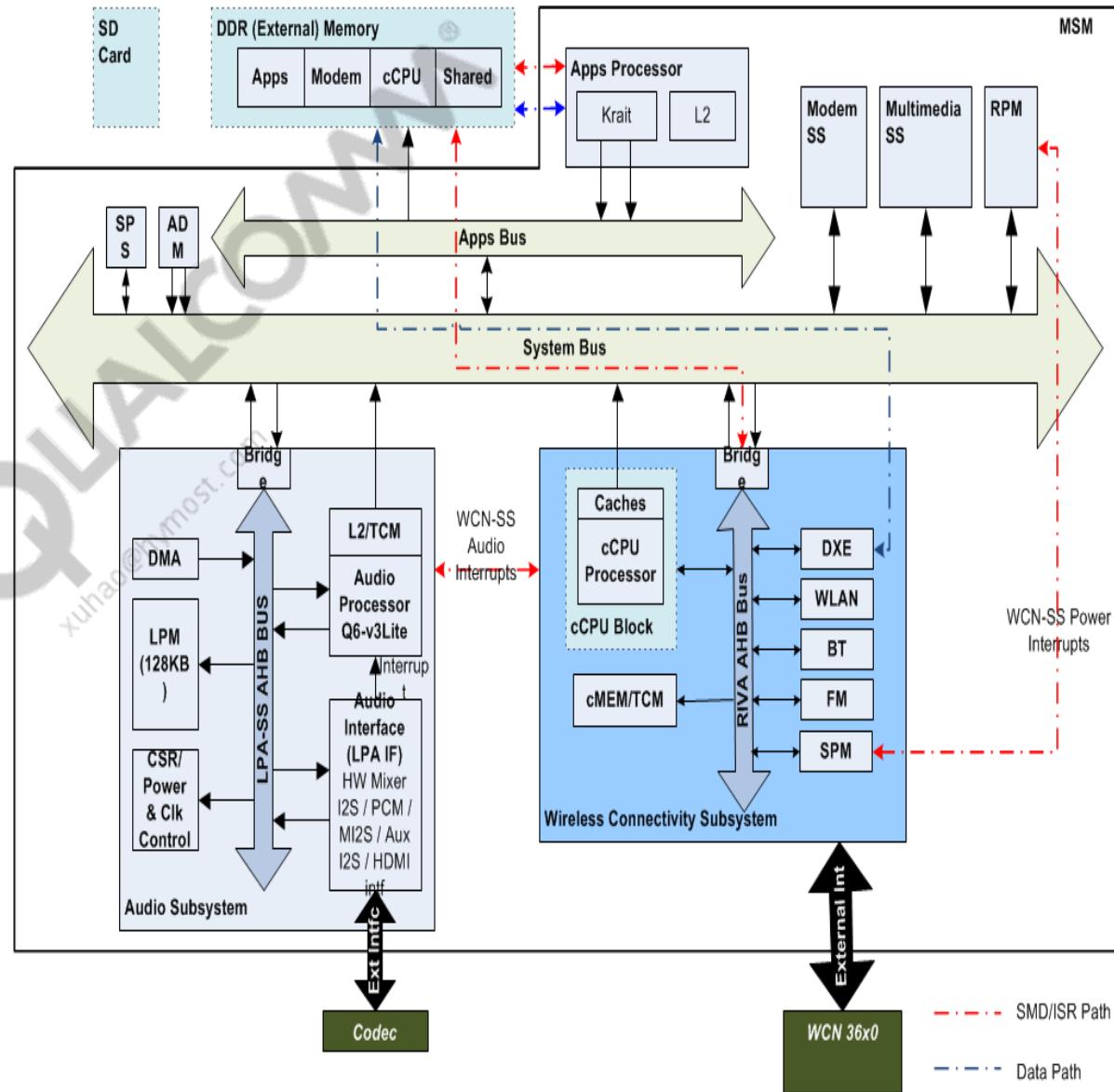
WCN36x0 can be attached to various MSM and APQ processors, such as MSM8960, APQ8060A, and APQ8064 which includes WCN subsystem.



Interface Pin Count = 14 pins  
10 digital pins mux'd behind existing pins  
& 4 new analog pins (I/Q)

# High-Level MSM Architecture

- Shared memory design using high-speed internal bus between aCPU and cCPU
- Enables higher system level throughput (.11ac) – Avoids SDIO bottleneck
- Bluetooth/FM code runs from system memory enabling quick addition of new features and bug fixes
- No PCM buffering allows for low-latency audio path
- WLAN data paths in hardware with small control path in software – Will not tax cCPU
- Small ROM footprint used for audio codecs and algorithms



## WCN36x0 WLAN/BT/FM Features

REDEFINING MOBILITY

# Integrated Connectivity Options for 28nm MSM Platforms

	WCN3660	WCN3660A	WCN3680	WCN3620
<b>CS Date</b>	Jan'12	Sep'12	Sep'12	Jun'13
<b>Die Size</b>	14.5mm <sup>2</sup>	14.5mm <sup>2</sup>	14.5mm <sup>2</sup>	11.5mm <sup>2</sup>
<b>WLAN standards</b>	802.11 a/b/g/n	802.11 a/b/g/n	802.11 a/b/g/n/ac	802.11 b/g/n
<b>WLAN bands</b>	2.4 and 5GHz	2.4 and 5GHz	2.4 and 5GHz	2.4GHz
<b>WLAN channel BW</b>	20 and 40MHz	20 and 40MHz	20, 40 and 80MHz	20MHz
<b>WLAN Peak Data Rate</b>	150Mbps	150Mbps	433Mbps	72Mbps
<b>BT</b>	BT3.x, BT4.0/LE	BT3.x, BT4.x/LE	BT3.x, BT4.x/LE	BT3.x, BT4.x/LE
<b>FM</b>	RX/TX, RDS, RBDS, RT+	RX/TX, RDS, RBDS, RT+	RX/TX RDS, RBDS, RT+	RX, RDS, RBDS, RT+
<b>ANT</b>	No	Yes	Yes	Yes
<b>External coupler support</b>	Yes with external power detector circuitry	Yes	Yes	Yes

# WCN36x0 Chipset Support for Android Release

		Ice Cream Sandwich	JellyBean Sep-Oct 2012	K Release 2Q 2013 <sup>(1)</sup>
8974	-	-	-	WCN3660A <sup>(2)</sup> WCN3680
8064	WCN3660	WCN3660 WCN3660A WCN3680	WCN3660 WCN3660A WCN3680	WCN3660 WCN3660A WCN3680
8960PRO	-	WCN3660A WCN3680	WCN3660A WCN3680	WCN3660A WCN3680
8960/8x60A	WCN3660	WCN3660	WCN3660	WCN3660
8x30/8x27	-	WCN3660	WCN3660	WCN3660

■ Maintenance mode

1 Estimated Google release dates

2 JB-MR1 or K-release

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

# WCN36x0 WLAN Android Software Versions by MSM PL

Product Lines	4.5	4.6	4.7	5.0
MSM8960 LA 2.0		■		
MSM8960 LA 2.2 + QSC1215		■	■	
APQ8064 LA 1.2	■			
APQ8064 LA 1.3	■			
APQ8064 LA 1.6			■	
MSM8930 LA1.0				
MSM8930 LA 1.7		■		
MSM8974 LA 1.0				■
MSM8x26 LA 1.0				■

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

# WCN36x0 Android WLAN Software Versions and Roadmaps

CS Date	Sept'12	Nov'12	Feb'13	April/May'13
WCN SW version	4.5	4.6	4.7	5.0 (Pronto)
802.11 a/b/g/n	■	■	■	■
WiFi Direct (P2P)	■	■	■	■
SoftAP (10/32 clients)	■	■	■ (32)	■ (32)
WPS2.0	■	■	■	■
WPA/WPA2 PSK	■	■	■	■
WPA/WPA2 Enterprise	■	■	■	■
EAP SIM/AKA	■	■	■	■
WAPI	■	■	■	■
BTC	■	■	■	■
WMM	■	■	■	■
WMM-PS	■	■	■	■
Voice Personal	■	■	■	■
RTT support	■	■	■	■
Transmit Power Control	■	■	■	■
Offloads (ARP, PNO, MCBC, Pattern Matching)	■	■	■	■
LFR w/OKC	■	■	■	■
CCXv4 ASD	■	■	■	■
Sigma endpoint	■	■	■	■
WiFi Display	■	■	■	■
LTE coexistence	■	■	■	■ (Advanced)
802.11ac (WCN3680)	■	■	■	■ (LDPC/TxBF)
Concurrency	■(SCC)	■(MCC)	■(BTC/Adv Sch)	■(PSM)
Passpoint		■(Phase 1)	■(Phase 1)	■(Phase 2)
CRDA		■	■	■
TDLS			■(Phase 1)	■(Certification)
Voice Enterprise			■(11r/k)	■(11r/k/v)
PMF				■

# WCN36x0 BT Profile Support by Android Software Release

BT/PN	8960	APQ8064/Fusion3	8x30	8x27
<b>Bluetooth Profiles</b>				
A2DP 1.2	■	■	■	■
AVRCP 1.0	■	■	■	■
AVRCP 1.3	■	■	■	■
BPP 1.2	■	■	■	■
DUN 1.1	■		■	■
FTP 1.1 / 1.2	■	■	■	■
HFP 1.5	■	■	■	■
HFP 1.6	■	■	■	■
HSP 1.1	■	■	■	■
HID 1.0	■	■	■	■
MAP 1.0	■	■	■	■
OPP 1.1 / 1.2	■	■	■	■
PAN 1.0	■	■	■	■
PBAP 1.0 / 1.1	■	■	■	■
SAP 1.1	■		■	■
SDP 1.1	■	■	■	■
SPP 1.1	■	■	■	■
<b>Bluetooth 4.0 Low Energy Profiles</b>				
GATT Client	■	■	■	■
GATT Server	■	■	■	■
Thermometer (Client)	■	■	■	■
Proximity (Client)	■	■	■	■

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

# WCN36x0 FM Android Software Features

Features	
FM Rx RDS	■
FM Rx Over BT	■
FM Rx DSP Routed Audio	■
FM Rx Multimedia Concurrency	■
FM Tx RDS	■
FM Tx Multimedia Concurrency	■
FM FTM Enhancements	■
RSSI Measurement Tool	■
Enhanced RDS	■*
Unicode RDS	■*
FM Concurrency with WAN CDMA 1X	■
FM Concurrency with WAN GSM	■
FM Concurrency with WAN UMTS	■

\*Features provided outside of main PL

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

# WCN3680: 802.11ac features

	Feature	MSM8960 + WCN3680 H2'12 CS	MSM8974 + WCN3680 H1'13 CS
Bandwidth	20MHz	YES	YES
	40MHz contiguous	YES	YES
	40MHz non-contiguous	-	-
	80MHz contiguous	YES	YES
	80MHz non-contiguous	-	-
	160MHz contiguous	-	-
	160MHz non-contiguous	-	-
SS	1 spatial stream	YES	YES
	2 spatial streams	-	-
Rates	MCS 0 - 7 support	YES	YES
	MCS 8 - 9 support	YES	YES
MU-MIMO	11ac MU-MIMO Transmission	-	-
	MU-MIMO Feedback support	-	YES
	MU-MIMO ACK	-	YES
Misc	Multi-channel CCA detection	YES	YES
	STBC (2x1 RX)	YES	YES
	11ac A-MPDU Power Save	YES	YES
	CSI Feedback of 11ac sounding	-	YES
	Advanced coding (LDPC)	-	YES
	SU TX BF Feedback Support		YES

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

# MSM8974 WCN-SS Android Feature Updates

## ■ WLAN feature additions

- MU-MIMO (Receiver mode)
- Advanced 11ac features
  - ◆ Tx Beam Formee
  - ◆ LDPC
- TCP checksum offloading, RTT3

## ■ Power management improvements

- uBSP – Low-power mode for WLAN BMPS / BT LPPS

**Note:** Data in this table can change without notice; for latest version, contact QCA marketing.

## WCN36x0 Windows BT/WLAN Features

REDEFINING MOBILITY

# Windows WLAN Features Supported

Features	GDR1	GDR2	GDR3
11b/g 2.4 GHz/20 MHz	Y	Y	Y
11b/g/n 2.4 GHz/20 MHz	Y	Y	Y
11a/n 5 GHz/20 MHz	Y	Y	Y
11a/n 5 GHz/40 MHz	Y	Y	Y
11ac	N	N	Y (with 3680)
IBSS/ad-hoc	N	N	N
BSS (client)	Y	Y	Y
BSS (AP)/QCMobileAP (8 clients)	Y (Modified P2P GO implementation)	Y (Modified P2P GO implementation)	Y (Modified P2P GO implementation)
Wi-Fi Direct / P2P	N	N	N
Wi-Fi display	N	N	N
TDLS	N	N	N
BT-AMP	N	N	N
BT coexistence (data)	Y	Y	Y
BT coexistence (audio)	Y	Y	Y
Multi-link Concurrency	N	N	N
Multicast support (32 addresses)	Y	Y	Y
WAPI (as extension)	Y	Y	Y
CCX (as extension)	N	N	N
Virtual Wi-Fi support (handling multiple ports)	Y	Y	Y
Native Wi-Fi driver model	Y	Y	Y
ARP and NS Offload	N	Y	Y
Beacon early Termination	N	Y	Y
Nth Beacon Filtering	N	Y	Y
Modulated DTIM	N	Y	Y
NDIS support	6.30	6.30	6.30

# Windows WLAN Features Supported (cont.)

Features	GDR1	GDR2	GDR3
Programmable packet filtering (8 patterns) (D0 packet coalescing)	Y	Y	Y
Wake on WLAN	Y	Y	Y
AOAC	N	N	N
Preferred network list offload	Y	Y	Y
WMM-SA	N	N	N
WMM-PS	Y	Y	Y
WMM	Y	Y	Y
WMM-AC (client only)	Y	Y	Y
WNM and power network save (11v)	N	N	N
Hotspot 2.0 – 11u (client only)	N	N	N
Protected management frames	Y	Y	Y
Voice personal	Y	Y	Y
Voice enterprise (as extension)	N	N	N
WPS 2.0	N	N	N
Open	Y	Y	Y
WEP	Y	Y	Y
WPA-Personal/TKIP	Y	Y	Y
WAP2-Personal/TKIP	Y	Y	Y
WAP2-Enterprise/AES	Y	Y	Y
WAP2-Personal/AES	Y	Y	Y
WPS (QCMobileAP)	N	N	N
WPA2 personal (QCMobileAP)	N	N	N
Wi-Fi position assist	Y	Y	Y

# Windows Bluetooth Features Supported

## ■ WinRT

- Microsoft Bluetooth inbox core-stack
  - ◆ Bluetooth 3.0 (eL2CAP)
  - ◆ Bluetooth 4.0 LE
- Microsoft Bluetooth inbox profiles
  - ◆ HID 1.0, PAN 1.0, OPP 1.2, HCRP (print)
  - ◆ HFP 1.5, A2DP 1.2, AVRCP 1.3
  - ◆ Additional profiles depend on Microsoft

## ■ WP8

- BR/EDR
  - ◆ Depends on Microsoft
- BLE
  - ◆ QCOM provides own BLE SDK
- Check current schedule with Microsoft

# Windows FM Features Supported

- FM Receiver
  - FM Radio Data System (RDS)/ Radio Broadcast Data System (RBDS)

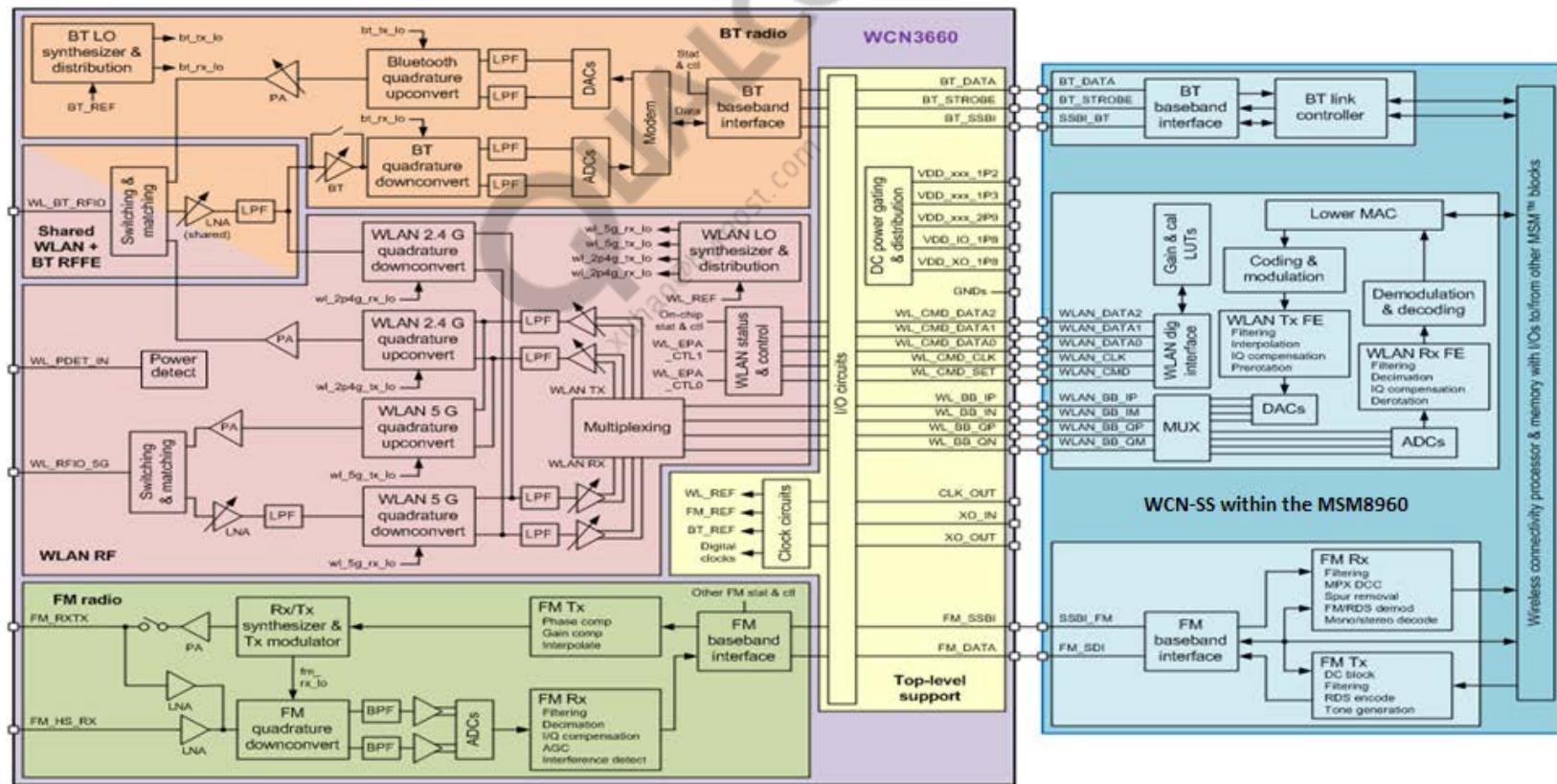
QUALCOMM®  
xuhao@hymost.com

# WCN36x0 Hardware Architecture

REDEFINING MOBILITY

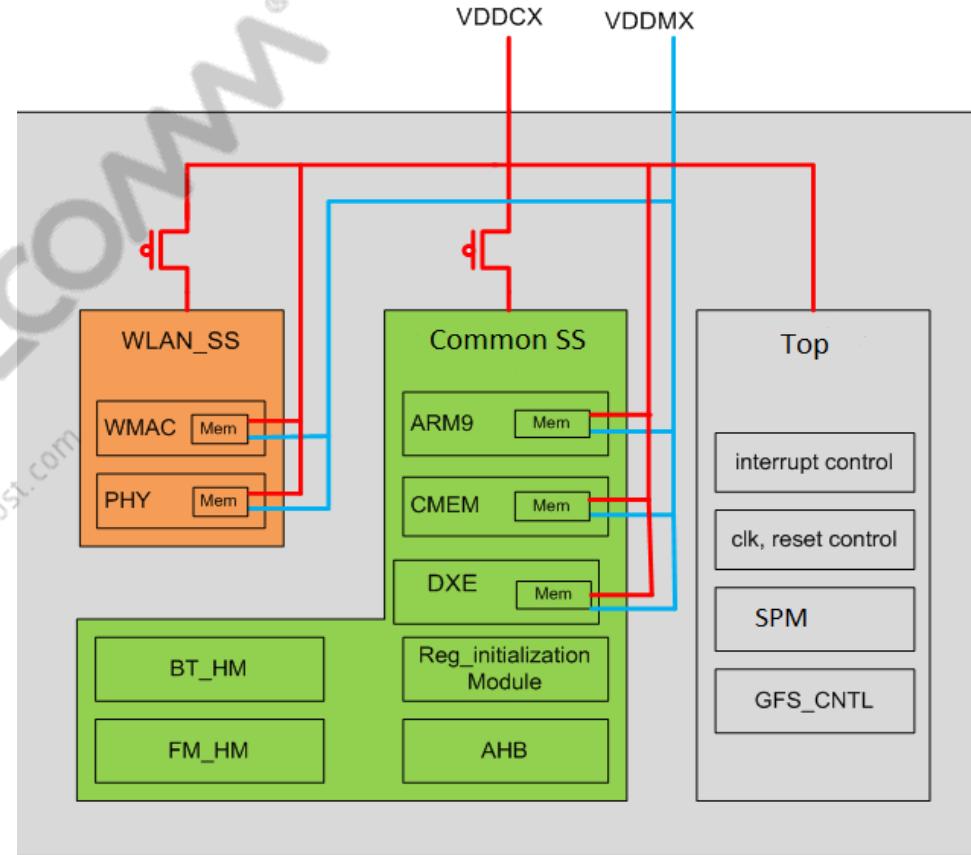
# WCN36x0 System Interconnects

- In WCN36x0 Bluetooth/FM splits the digital processing between MSM and WCN – the digital interface is a 5-pin proprietary interface with data rates <10 Mbps
- WLAN utilizes an analog differential IQ interface similar to WAN to support high data rates (WL\_BB\_XX pins)



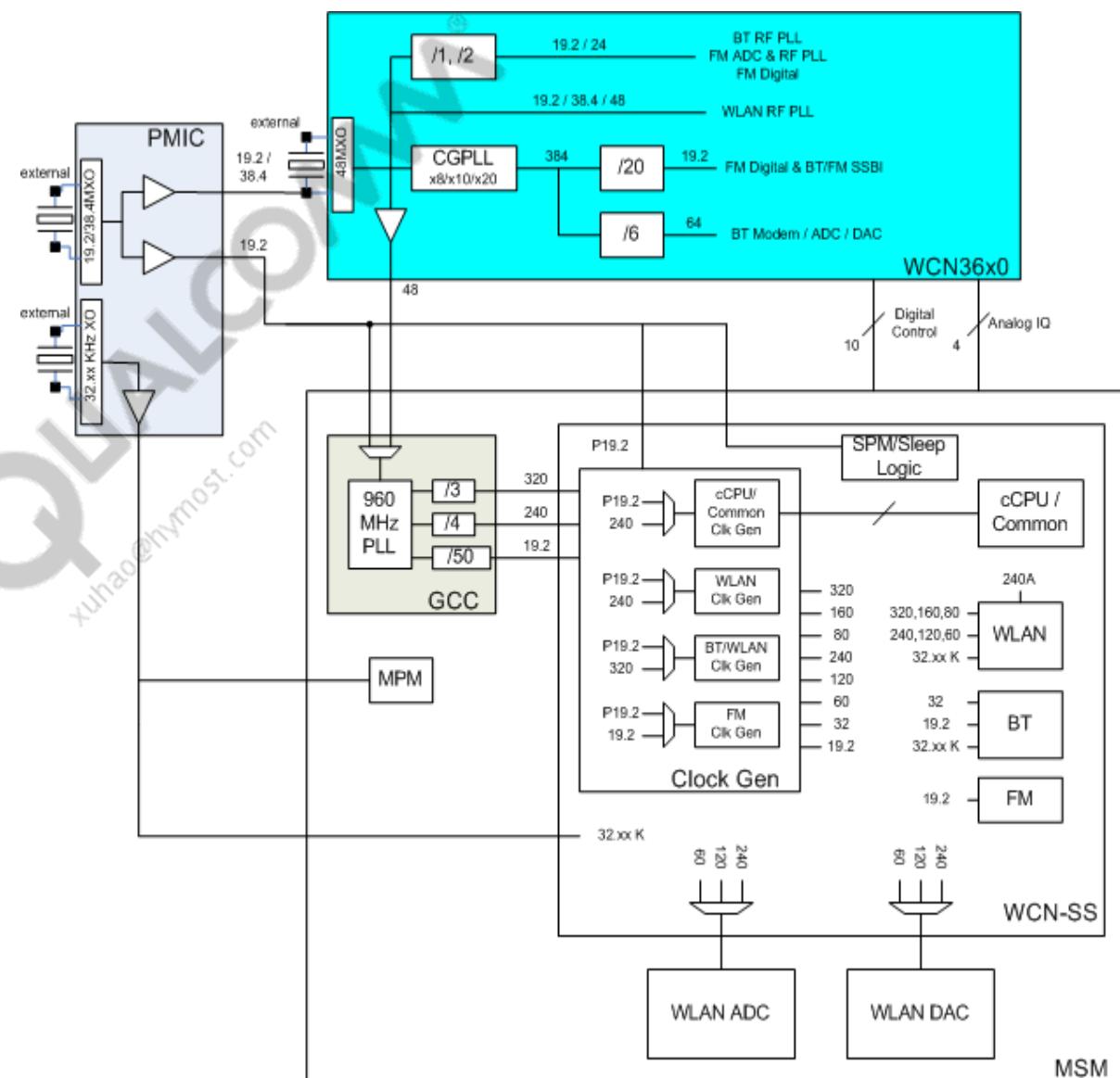
# WCN-SS Power Rails and Headswitches

- VDDmx is the memory voltage and powers RAM blocks
- VDDcx is the chip voltage and powers digital (nonmemory) blocks
- RPM controls the voltage of VDDcx and VDDmx to satisfy the dynamic requirements of all subsystems that share access to these rails
- Three blocks with isolated power control – WLAN, Common, and Top (Always-On); headswitches control power to WLAN and Common
- WLAN software controls the headswitch to WLAN block; shuts it down whenever WLAN is idle
- Subsystem Power Manager (SPM) hardware block controls the headswitch to common block



# WCN36x0 Clock

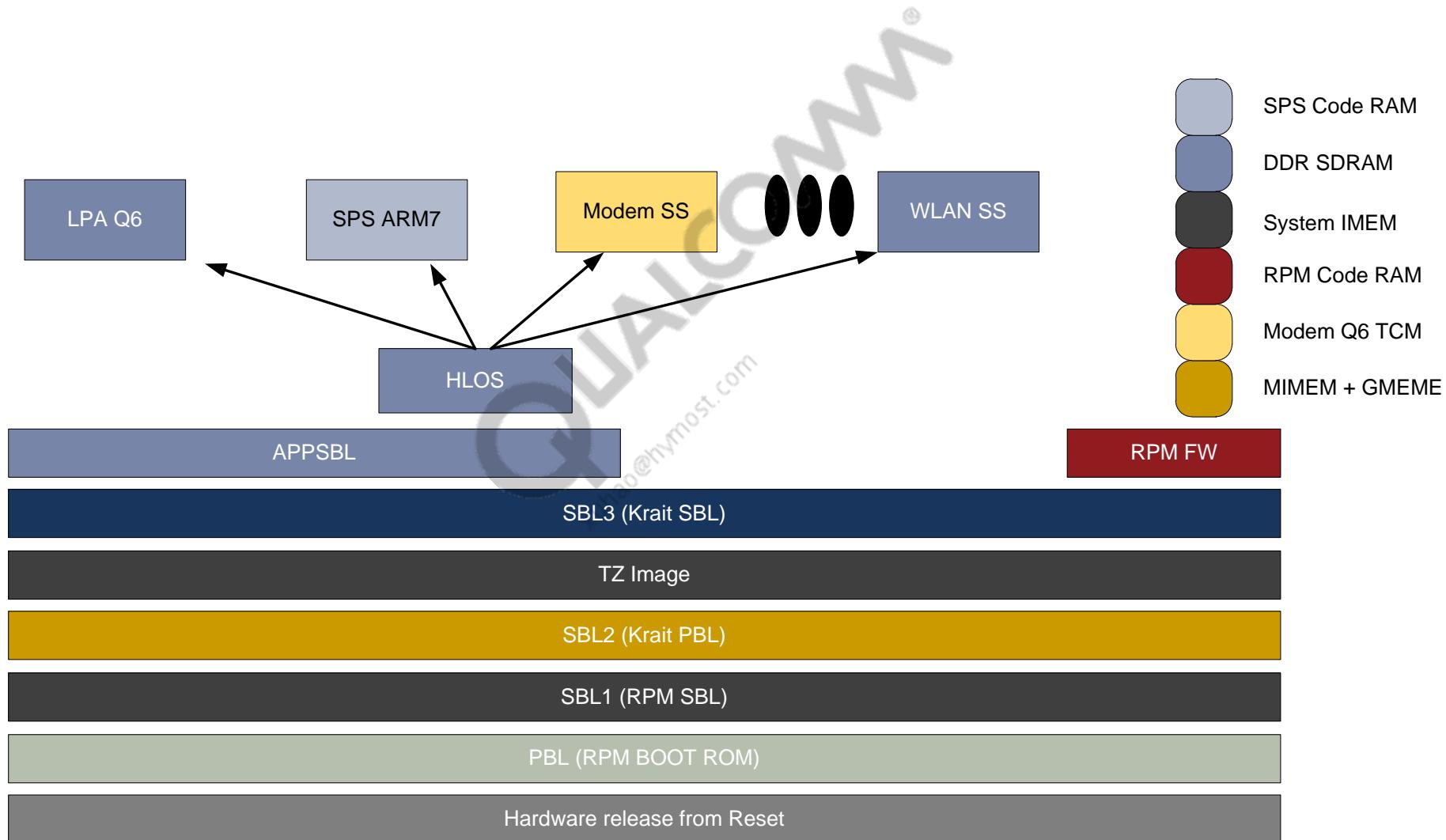
- WCN36x0 has an optional 48 MHz reference clock that is required for WLAN 5 GHz operation
- 960 MHz PLL is programmed by WCNSS software, to be sourced from either 19.2 MHz or 48 MHz reference clock – until PLL output is stable, WCNSS is clocked directly off the 19.2 MHz reference clock
- CGPLL on the WCN3660 device is used to drive modem, ADC/DAC, and digital blocks for Bluetooth and FM; it is controlled by WCNSS software



## WCN-SS Bootup Sequence

REDEFINING MOBILITY

# WCN-SS Bootup Sequence



# WCN-SS Firmware Images in Android

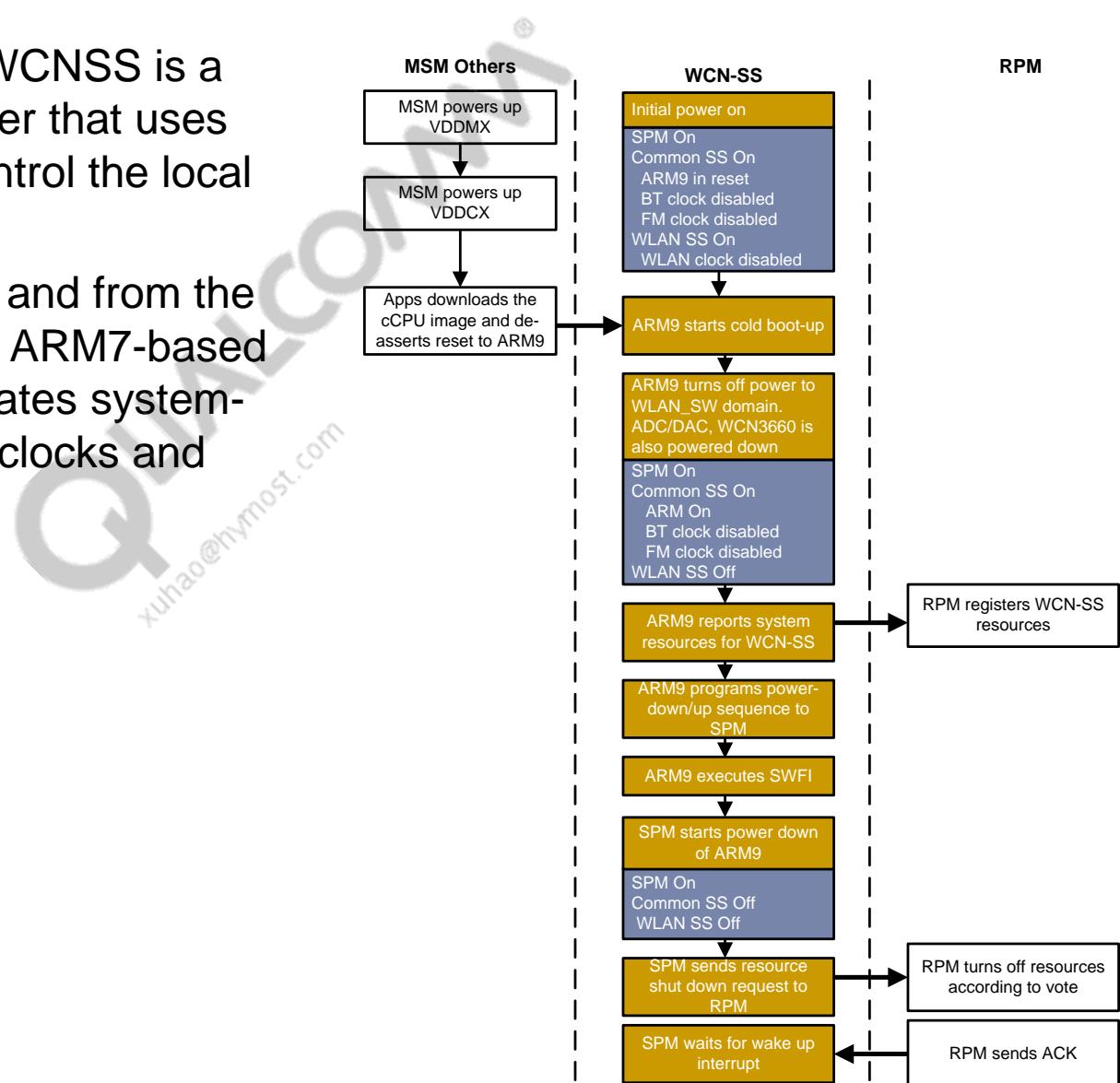
- WCNSS image is split into wcnss.mdt, wcnss.b00, wcnss.b01, wcnss.b02, wcnss.b04 in M8960AAAAANAZW<build id>  
  \wcnss\_proc\build\ms\bin\PIL\_IMAGES\8960\_SPLITBINS\_SCAQBAF
- Split files are combined into NON-HLOS.bin and flashed into modem partition
  - Mounted into /firmware folder when Android brings up  
(android/device/qcom/msm8960/init.target.rc)
  - All wcnss.\* files would be in /firmware/image folder
  - Symbolic link to /system/etc/firmware folder for request\_firmware kernel API to load images files (android/device/qcom/msm8960/init.qcom.modem\_links.sh)
  - Reference Solution 00021245 about update Riva image manually

# WCN-SS Bootup Sequences in Android

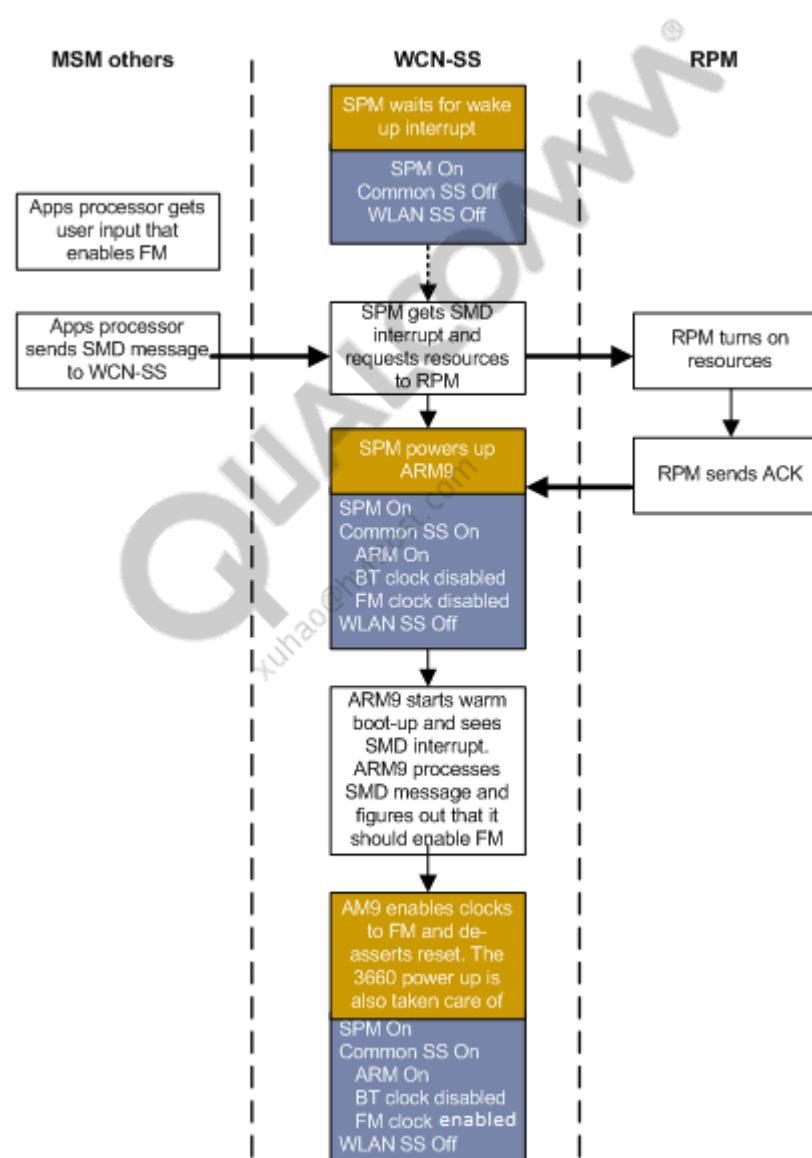
- The wcnss-core driver brings up WCNSS
  - Script /system/etc/init.qcom.wifi.sh executes “echo 1 > /dev/wcnss\_wlan” to let driver know that user-space is ready to handle firmware download requests
    - ◆ init.qcom.rc declares /init.qcom.syspart\_fixup.sh to run on early-boot stage and /system/etc/init.qcom.wifi.sh is executed inside the script.
  - The driver turns on power, configures clock source (19.2 MHz or 48 MHz, depends on has\_48mhz\_xo parameter) for WCN36x0 and then calls pil\_get to initialize WCNSS hardware, loads WCNSS firmware and deasserts cCPU reset to let cCPU start running.
  - To configure clock source as 19.2 MHz, modify init.qcom.wifi.sh to set parameter has\_48mhz\_xo
    - ◆ Add “echo 0 > /sys/module/wcnsscore/parameters/has\_48mhz\_xo” before “echo 1 > \$wcnssnode” in init.qcom.wifi.sh
    - ◆ The default setting is 48 MHz clock.
- Source code
  - wcnss-core driver: android/kernel/drivers/net/wireless/wcnss
  - PIL: android/kernel/arch/arm/mach-msm/peripheral-loader.c, pil-riva.c
- A message “wcnss\_wlan\_ctrl\_probe: SMD ctrl channel up” in kernel log implies that cCPU is up and running

# WCN-SS Bootup Sequence

- The SPM block within WCNSS is a hardware sleep controller that uses signal sequences to control the local sleep/wake transition.
- It has signals that go to and from the RPM block, which is an ARM7-based subsystem that coordinates system-wide power resources (clocks and voltages).



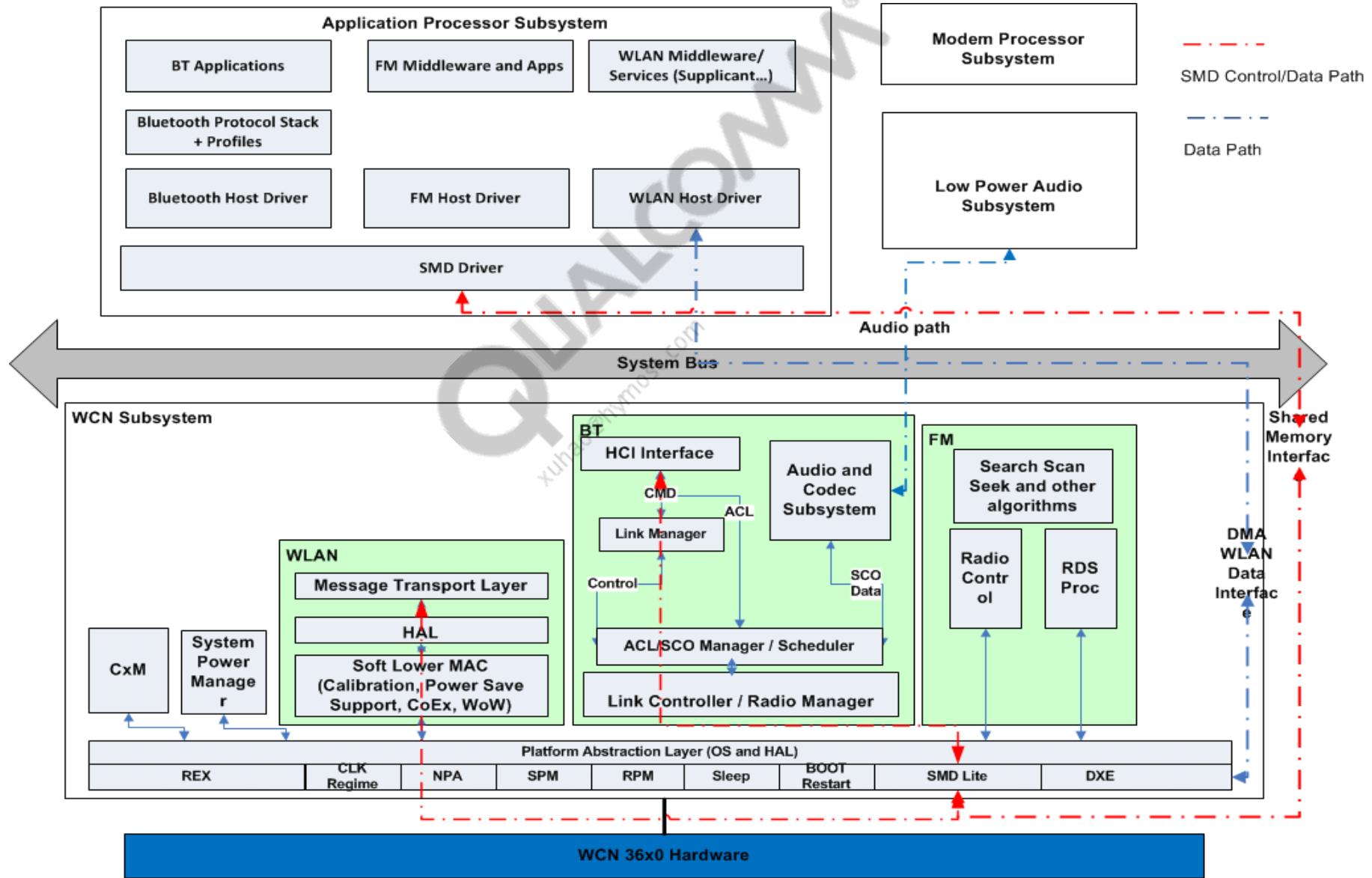
# Example of Enabling WCN-SS FM Module



# WCN-SS Software Architecture

REDEFINING MOBILITY

# WCN-SS High-Level Software Diagram



## WCN-SS Overview

- WCNSS (Wireless Connectivity Sub-System) component and includes BT/FM and WLAN
- WLAN/BT/FM firmware runs on the WCNSS ARM9 Processor
- WLAN MLME S/W (802.11 Protocol Stack) runs on APP-SS
- BT Host Stack runs on APP-SS

## Control Path – SMD Lite

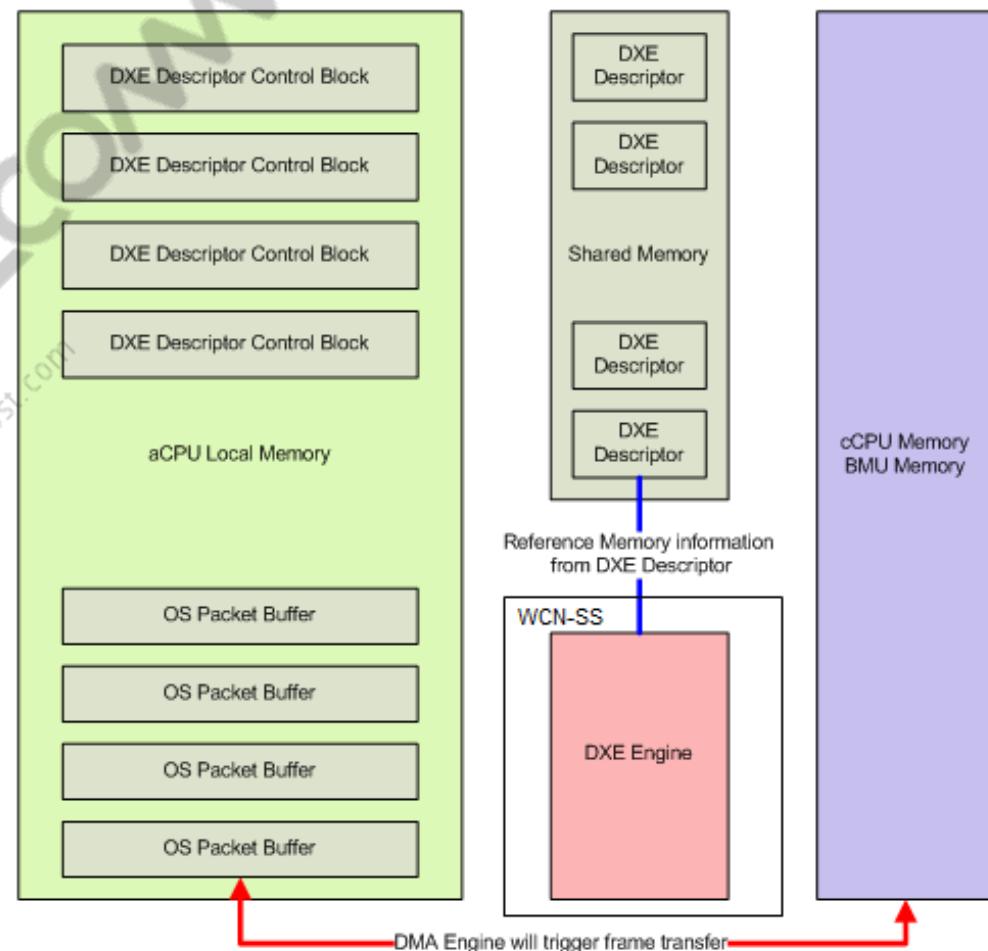
- SMD Lite is a lightweight shared memory driver and is designed to support low-latency control messages using a simple API. It is not suited for data flows where throughput is more important than latency.
- SMD Lite has two key advantages over classic SMD for control messages.
  - A simple read/write API allowing messages to be sent and received immediately in the caller's context without blocking or context switching
  - A lightweight, taskless implementation for cores where all of SMD is not necessary
- Each channel has one Tx FIFO and one Rx FIFO, each end can specify inbound FIFO size (4 K if not specified) when opening the port, each end must open the same named port. Names must be unique across the entire system.
- Ports preserve message boundaries across the interprocessor channel; one write into port exactly matches one read from the port.
- Clients receive notification via a callback when the port's status changes and may call Read or Write functions from the callback or notify their own threads for further processing

# SMD Lite Channels

- Between cCPU and aCPU
  - One for WLAN Upper MAC (UMAC) and Lower MAC (LMAC) messaging
  - One for Bluetooth HCI commands and events
  - One for FM commands and responses
  - One for logging shared by WLAN, Bluetooth, and FM

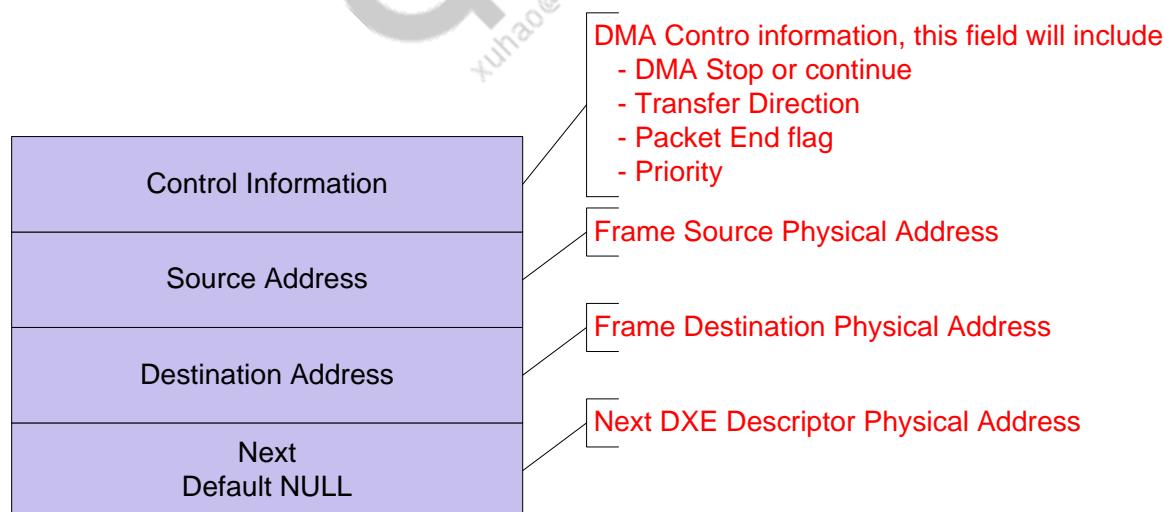
# Data Path - DXE Engine

- The DXE engine is a DMA controller residing in WCN-SS transferring frames between cCPU memory and aCPU memory to transmit and receive packets.
- The DXE descriptor is a circular linked list. The DXE engine reads the DXE descriptor to transfer the data and control the status.
- The RIVA WLAN subsystem has four DMA channels, two Rx channels, and two Tx channels.
  - Rx high-priority channel – For management frame
  - Rx low-priority channel – For data frame
  - Tx high-priority channel – For management frame
  - Tx low-priority channel – For data frame
- Each of the channels will have its own DXE descriptor set and DXE descriptor control block set.



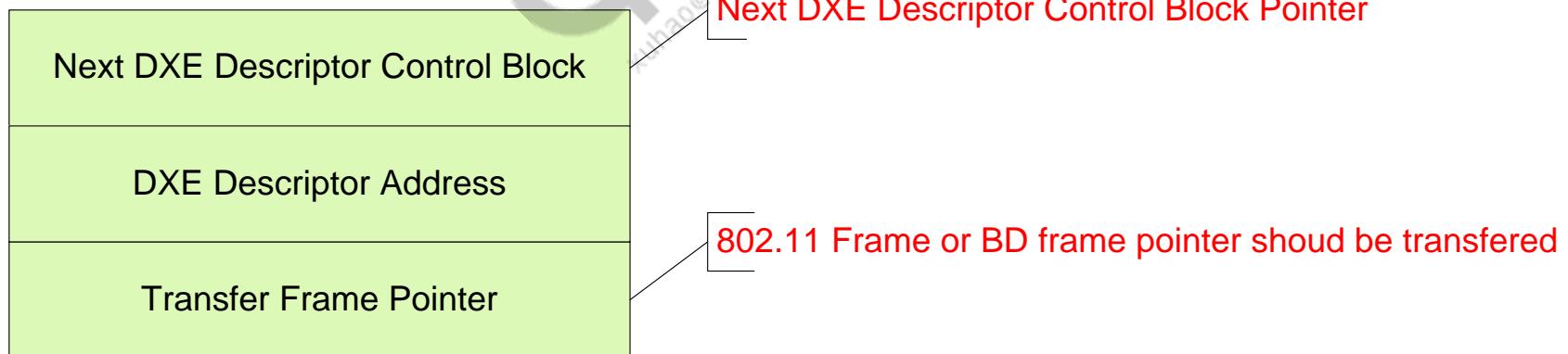
# Data Path - DXE Descriptor

- The DXE descriptor contains
  - Frame source address and frame destination address
  - Link to next descriptor; forms a circular ring of descriptors
  - Control information for frame validity, DMA status, etc.
- The DXE descriptor will be located within a shared memory area, as the host DXE driver and DXE engine must access this common area



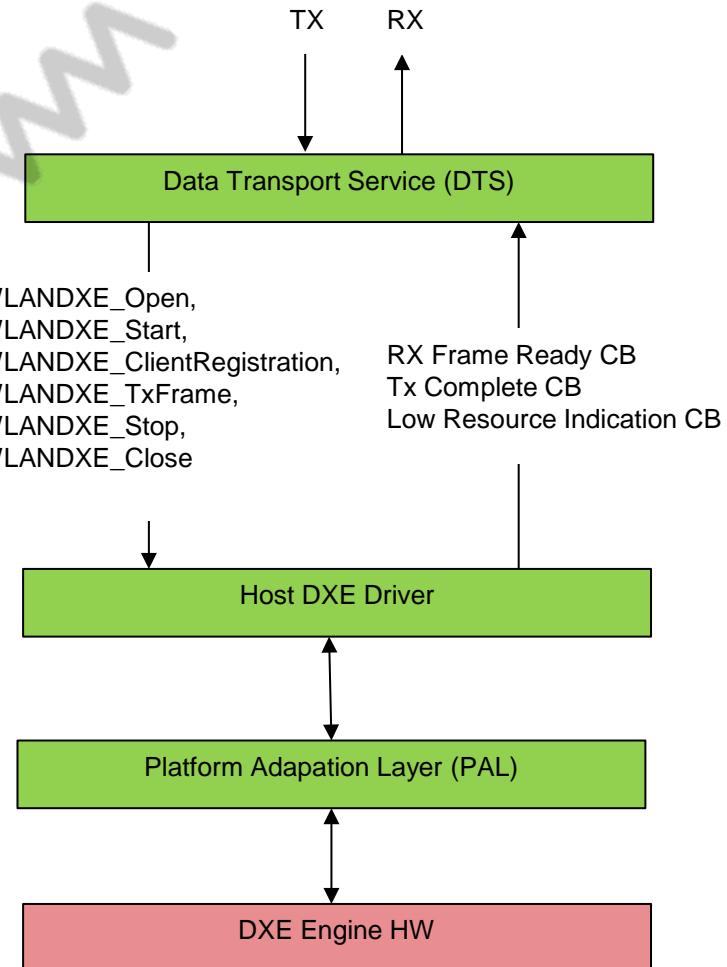
# Data Path - DXE Descriptor Control Block

- The host DXE driver uses DXE the descriptor control block to:
  - Program RIVA DXE engine registers
  - Maintain the DXE descriptor location and status
  - Know the active DXE descriptor location
- The DXE descriptor controller is mapped one-to-one with the DXE descriptor



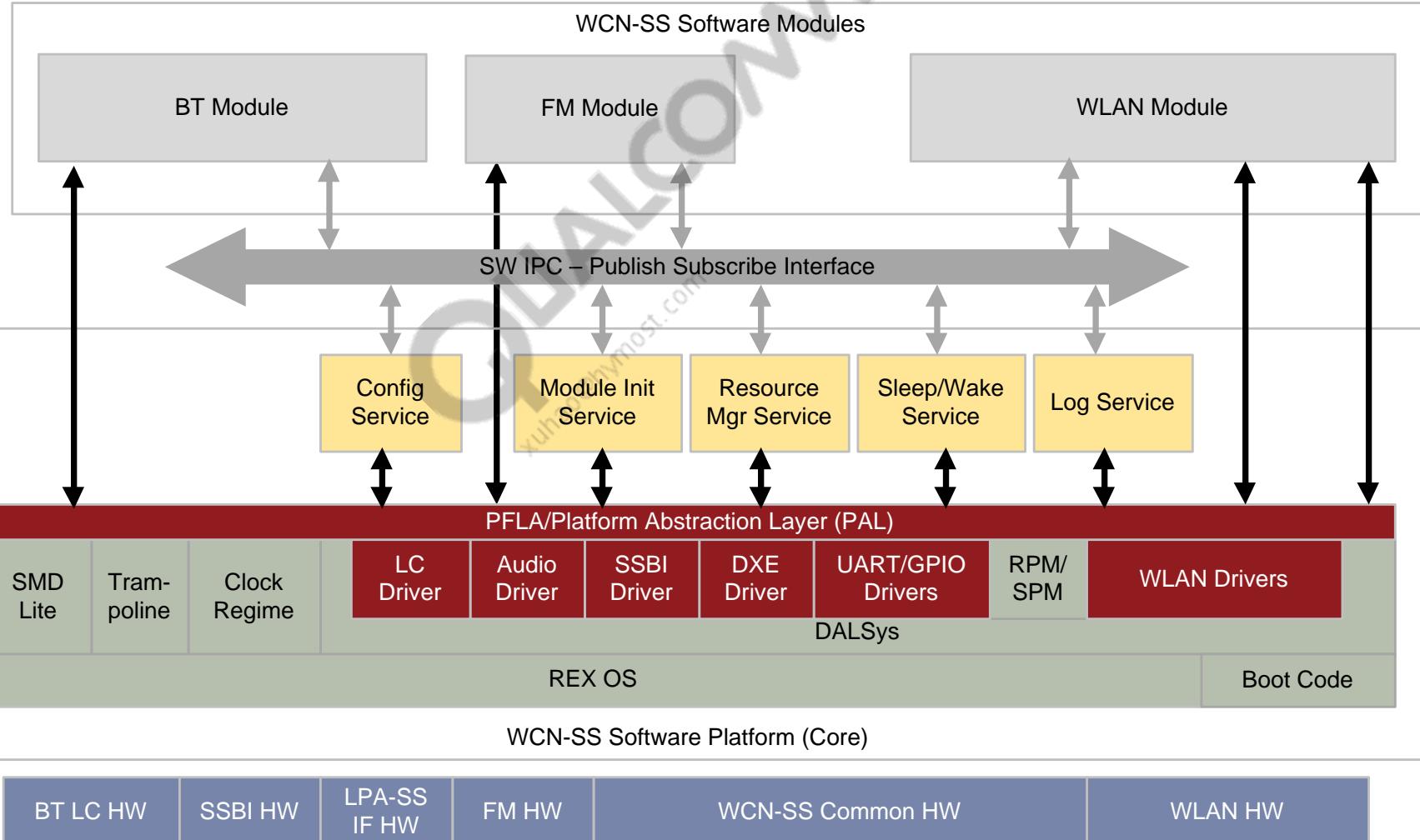
# Data Path – Host DXE Driver

- Data Transport Service (DTS) is the client of the host DXE driver. DTS pushes a frame to the DXE driver to be sent out, and the host DXE driver notifies DTS that the new Rx frame is ready to pick up.
- The Host DXE driver's roles are:
  - Receive and handle DXE hardware interrupts for Tx and Rx
  - Maintain a DXE descriptor control block for each channel
  - Program the DXE descriptor
  - Program the RIVA DXE engine registers
- The Platform Abstraction Layer (PAL) roles related to data transfer are:
  - Packet handle, allocate, free
  - Dispatch packets to Tx and Rx threads
  - Abstract register read, write function interface to DXE
- Source code
  - DTS: CORE/WDI/TRP/DTS/src
  - DXE: CORE/DXE
  - PAL: CORE/WDI/WPAL



# WCN-SS Software Architecture Overview

WCN-SS



# DAL (Device Adaptation Layer)

- DAL is a SW framework to provide the common API for the platform agnostic
- A common shim layer that provides DALs access to the OS services in a way that is OS neutral
  - The implementation of the DALSys is OS dependent however that is transparent to the DALSys users
  - Simple and efficient, scalable to the environment
  - It decouples the stacks from the low level microkernel running on WCNSS cCPU
- DALSys provides the basic necessary OS Services that the connectivity software need
- DAL Interrupt Controller, DAL Timer, DAL Timetick

# CoreBSP – DALSys / REX

## ■ DALSys

- A common shim layer that provides DAL access to the OS services in a way that is OS-neutral
- The implementation of the DALSys is OS-dependent; however, that is transparent to DALSys users
- Decoupling the stacks from the low-level microkernel running on the cCPU can help maximize portability of WLAN, Bluetooth, and FM modules

## ■ Kernel Services provided by DALSys

- Synchronization (interrupt, mutex or critical sections)
- Events and callbacks
- Memory and cache operations
- Physical memory manipulation
- Shared execution contexts Workloops ( shared threads )

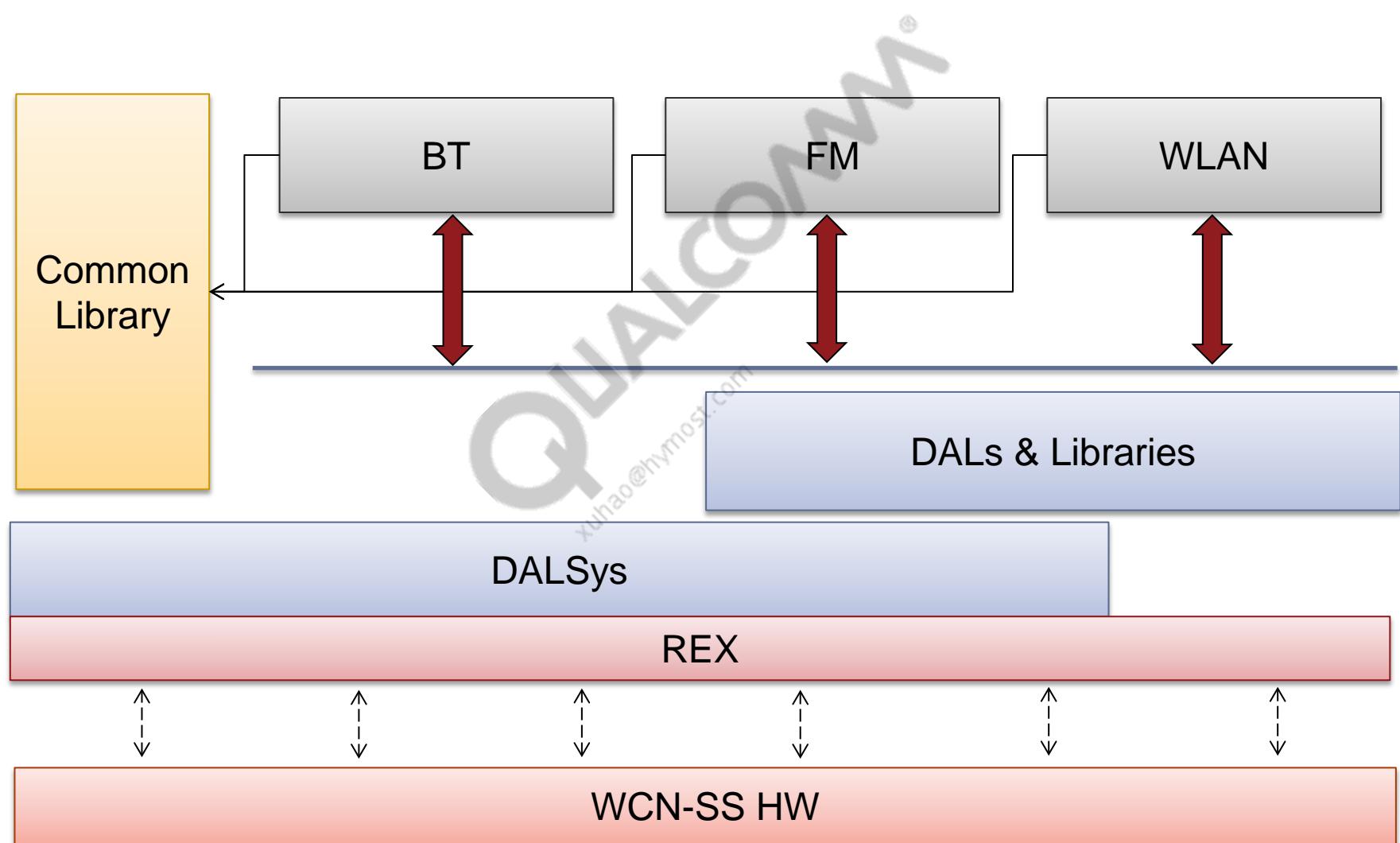
## ■ The underlying RTOS for DALSys is REX

- REX is small(~4K), highly predictable (executed code is well defined), low latency in interrupt processing and can bring all the services that WCNSS needs

REDEFINING MOBILITY

## DALSYS/OS Services

# WCNSS DAL High-Level Diagram



## OS Services – Task and Signal

- DALSys does not provide traditional thread services; however it provides execution contexts in the form of *Workloops*
- A workloop internally is a thread context that processes DALSys Events
- Workloops are created dynamically and use a predefined stack size
- They are scheduled in the same manner the OS would schedule any other task (in case of REX this will be strict priority based)
- In order for a workloop to process events the events must be initially created and added to the workloop
- Once an event is added a function pointer designating the processing function of that event will be supplied
- Multiple clients can add events to a workloop, thus sharing the workloop (advantageous for minimizing resources used by the workloop – its own thread, its own stack, so forth...)
- Events will be posted to the workloop by using the event control APIs offered by DALSys (this would be equivalent to signaling a task)
  - The workloop will be scheduled for execution and will process its event

# OS Services – Tasking and Signaling

Operation	DALSys	REX
Create a thread/task	<p><b>DALSYS_RegisterWorkLoopEx</b></p> <p>*DAL operates with work loops Registers a work loop function (executed in the context of a new thread)</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><b>name</b> – Unique Workloop name</li> <li><b>dwPriority</b> - Priority of this execution context</li> <li><b>dwMaxNumEvents</b> - Maximum number of events that can be added</li> <li><b>phWorkLoop</b> - Return parameter to this Work Loop Handle</li> <li><b>pObj</b> - Pointer to Work Loop Object (if statically alloc-ed)</li> </ul>	<p><b>rex_def_task(*rex_create_task)</b></p> <p>Defines a new rex task that will be placed in the task list. Creates a new thread and an event for the new task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><b>p_tcb</b> - valid tcb for new task</li> <li><b>p_stack</b> – stack for the new task</li> <li><b>p_stksiz</b> - stack size for new task</li> <li><b>p_pri</b> - priority for new task</li> <li><b>p_task</b> - task startup function</li> <li><b>p_param</b> - parameter for new task</li> </ul>
Signal a thread/task	<p><b>DALSYS_EventCreate</b></p> <p>Construct a DALSYS Event Object.</p> <p><b>dwAttrbs</b>: Event Attribs</p> <p><b>phEvent</b> : Return parameter, upon successful invocation, points to the DALSYS Event Handle <b>pEventObj</b> : Pointer to the Event Object if user doesn't desire dynamic memory allocation for the Event Object. NULL Otherwise</p> <p><b>DALSYS_AddEventToWorkLoop</b></p> <p>Add events to the work loop.</p> <p><b>hWorkLoop</b> - Handle to the registered work loop</p> <p><b>pfnWorkLoopExecute</b>: Function to be invoked when the associated event is triggered.</p> <p><b>pArg</b> - Arg to be passed to the work loop function <b>hEvent</b> - Handle to desired add event <b>h</b></p> <p><b>Sync</b> - Handle to synchronization object to be associated with this event</p> <p><b>DALSYS_EventCtrlEx</b></p> <p>DALSYS Event control API</p> <p><b>hEvent</b> : DALSYS Event Handle</p> <p><b>dwCtrl</b> : DALSYS Event control command (see common defs)</p> <p><b>dwParam</b> : parameter to be passed to callback <b>pPayload</b> : payload message</p> <p><b>dwPayloadSize</b>: payload size</p>	<p><b>rex_set_sigs</b></p> <p>Sets signals for specified TCB, causes REX to reschedule system</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li><b>tcb_ptr</b> - tcb for which the sigs will be set</li> <li><b>sigs</b> - the sigs to set</li> </ul>

# OS Services – Synchronization

- DALSys provides a DALSysSyncObj as a generic synchronization mechanism
- A very versatile object that can build various types of synchronization mechanism based on the attributes that are given:
  - A simple resource synchronization (critical section/mutex or semaphore)
  - Resource Interrupt or IRQ Level synchronization for resources shared between the DAL and DAL's ISR, or the DAL interrupt context and the DAL
  - Non preemptive synchronization – equivalent to a task lock (interrupt disable)

Operation	DALSys	REX
Semaphore -init -acquire -release	<u>DALSYS_SyncCreate</u> <u>DALSYS_SyncEnter</u> <u>DALSYS_SyncLeave</u>	<u>rex_enter_crit_sect / rex_exit_crit_sect</u>
Disabled Interrupts	<u>DALSYS_SyncEnter</u> with non-preemption attributes.	<u>rex_int_lock</u> <i>Lock interrupt</i>
Enable Interrupts	<u>DALSYS_SyncLeave</u> used with non-preemption attributes.	<u>rex_int_free</u> <i>free interrupts</i>

# OS Services – Memory Management

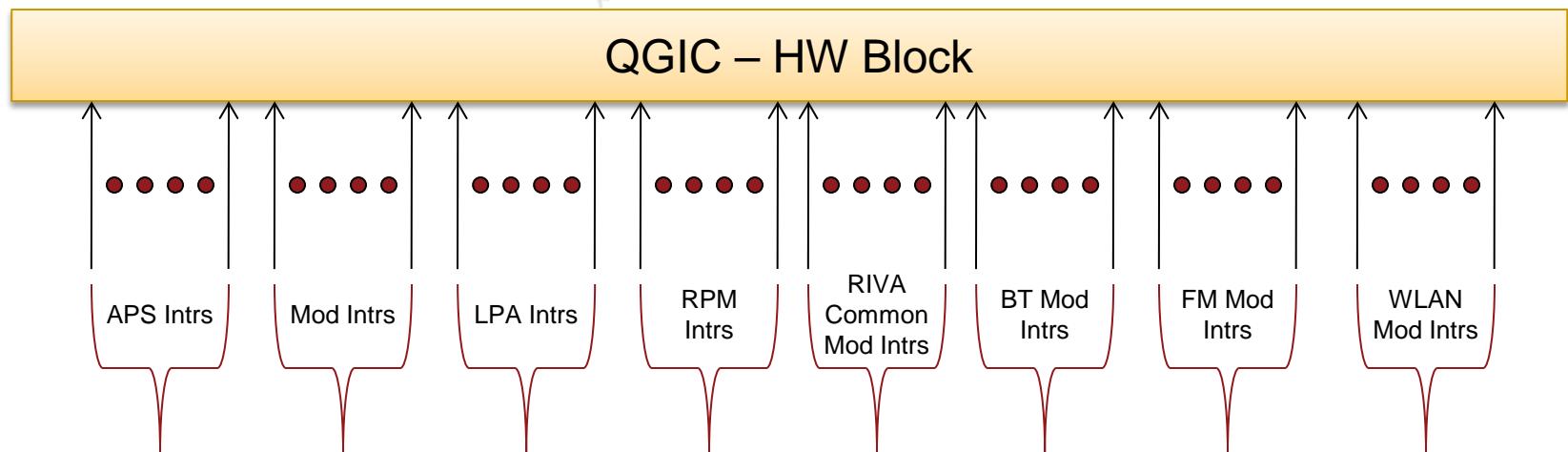
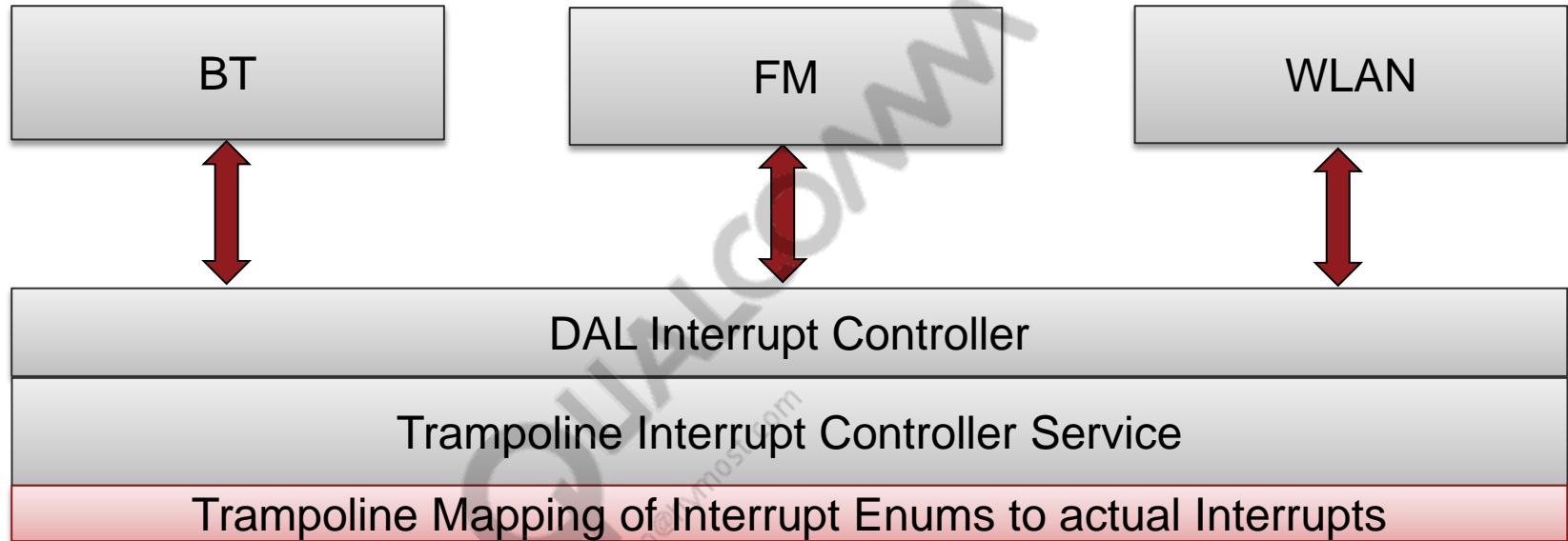
- DALSys provides corresponding API for allocating and freeing memory:
  - DALSys\_Malloc
  - DALSys\_Free
- Cache management is provided through the DALSys\_CacheCommand API
- DALSYS\_MemRegionAlloc and DALSysMemObj can be used to manage physical memory

Operation	DALSys	REX
Allocation	<b>DALSYS_Malloc</b> <i>Allocate memory from local heap</i> <b>dwSize</b> - Requested size of memory to be allocoed <b>ppMem</b> - Return parameter	<b>malloc</b> <i>Allocates a specified number of bytes and returns a pointer to this chunk of memory.</i>
Free	<b>DALSYS_Free</b> <i>Free allocated memory from local heap</i> <b>pMem</b> - Pointer to block of memory allocoed through <b>DALSYS_Malloc</b>	<b>free</b> <i>Frees a chunk of memory allocated</i>
Invalidate data cache line	<b>DALSYS_CacheCommand</b> Parameters: <b>CacheCmd</b> - Invalidate, Flush or clean ( see cache macro defs) <b>VirtualAddr</b> - Virtual Address <b>dwLen</b> - Length of the mem region	<b>mmu_dcache_inval_region</b> <b>mmu_dcache_flush_region</b> <b>mmu_dcache_clean_region</b> Invalidate MMU data cache

## DAL Interrupt Controller

REDEFINING MOBILITY

# DAL Interrupt Controller



# DAL Interrupt Controller – API Mapping

Operation	DALSys	REX
Registration	<p><u><a href="#">DallInterruptController_RegisterISR</a></u></p> <p>Parameters:</p> <p><i>h</i> - DalDeviceHandle  <i>intrID</i> - DALInterruptID  <i>isr</i> - DALISR  <i>ctx</i> - DALISRCtx  <i>bEnable</i> – should interrupt be enabled upon registration</p>	<p><u><a href="#">tramp_register_isr</a></u></p> <p>Parameters:</p> <p><i>irq</i> - which interrupt request line  <i>isr</i> - the handler, which takes "param" as an argument  <i>param</i> - the argument to pass to the ISR</p>
Enabling a specific interrupt	<p><u><a href="#">DallInterruptController_InterruptEnable</a></u></p> <p>Parameters:</p> <p><i>h</i> - DalDeviceHandle  <i>intrID</i> - DALInterruptID</p>	<p><u><a href="#">rex_enable_interrupt</a></u></p> <p>Tramp: in case this is used, the interrupt is enabled automatically at the time of registration</p>
Disabling a specific interrupt	<p><u><a href="#">DallInterruptController_InterruptDisable</a></u></p> <p>Parameters:</p> <p><i>h</i> - DalDeviceHandle  <i>intrID</i> - DALInterruptID</p>	<p><u><a href="#">rex_disable_interrupt</a></u></p> <p>Tramp: in case this is used, the interrupt is disabled automatically at the time of deregistration</p>
Clear an interrupt	<p><u><a href="#">DallInterruptController_InterruptClear</a></u></p> <p>Parameters:</p> <p><i>h</i> - DalDeviceHandle  <i>intrID</i> - DALInterruptID</p>	<p><u><a href="#">tramp_clear_interrupt</a></u></p> <p>Clears an interrupt. Should only be used if an interrupt must be cleared outside of the "tramp_isr" context.  <i>irq</i> - interrupt to clear</p>
Test an interrupt	<p><u><a href="#">DallInterruptController_InterruptStatus</a></u></p> <p>Parameters:</p> <p><i>h</i> - DalDeviceHandle  <i>intrID</i> - DALInterruptID</p>	<p><u><a href="#">tramp_is_interrupt_enabled</a></u>  <i>-checks if interrupt is enabled on the PIC</i></p> <p><u><a href="#">tramp_is_interrupt_pending</a></u>  <i>-checks to see pending to be serviced, in other words is set and enabled.</i></p>

## DAL Timers and Time Services

REDEFINING MOBILITY

# DAL Timer and DAL Timetick

## ■ DAL Timer

- Offers an interface that enable configuration and utilization of the Timer services independent from the low-level hardware details
- The API offers microsecond granularity but its precision depends on the underlying environment
- The timers can be run once or periodically depending on the event type associated with the timer
- The event attributes will also specify how this will be posted back to the calling workloop

## ■ DAL Timetick

- Offers a target independent API that clients may use to get access to Timetick services
- It can offer access to multiple system timers of various precision depending on HW availability
- The desired system timers will be identified by the client during the attach call

# Current usage of time API and mapping

Operation	DALSys	REX
Current time in us	<u><a href="#">DalTimetick_GetMs</a></u> Returns a monotonically increasing millisecond count <u><a href="#">DalTimetick_Get</a></u> <u><a href="#">DalTimetick_SclkToPrecUs</a></u>	<u><a href="#">timetick_get_safe</a></u>
Passed time in us	<u><a href="#">DalTimetick_GetElapsed</a></u> Compute the time elapsed from a previous timetick value  Option for us,ms,slck,min,hour	<u><a href="#">timetick_get_elapsed</a></u> (*available through the time service)

# BT/WLAN H/W Timers

## ■ BT Timers

- BT Clock Comparators: Interrupt is generated when the running BT clock matches the programmed comparator value.
- System Timer: The timer can be loaded with a count, and interrupt would fire once the timer expires. All the current OS time management is provided by this timer
  - ◆ The OS time management will be provided by DALSys + REX outside the BT stack, for BT usage the DAL Timer will be available

## ■ WLAN Timer

- WLAN Firmware is running off WLAN MTU H/W timers for SoftMAC/P2P/AP :
  - ◆ WLAN P2P Client Timer
  - ◆ WLAN P2P GO Timer
  - ◆ WLAN AP Beacon Timer
  - ◆ WLAN SLM Timer

# WCN-SS WLAN Software Diagram

## ■ Message Transport Layer

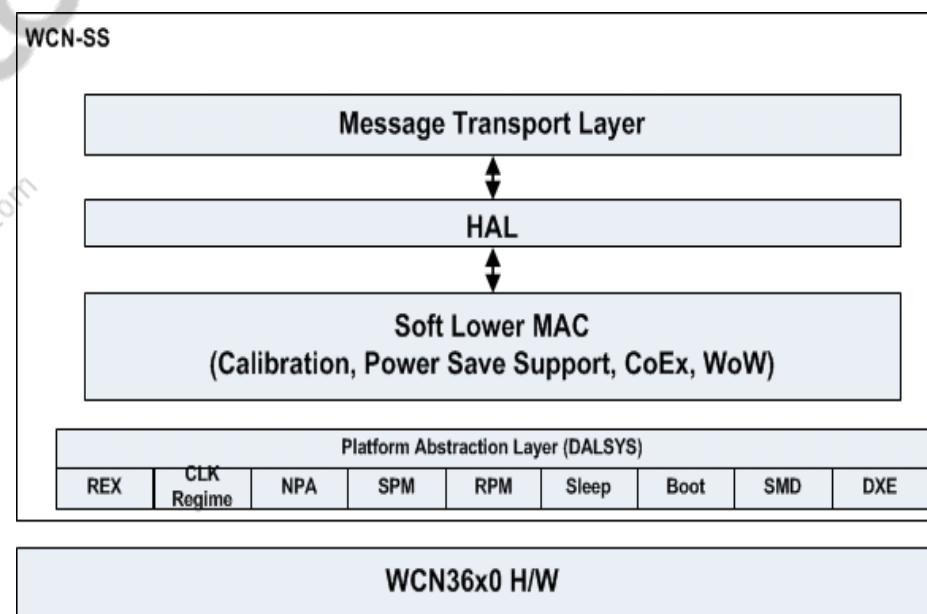
- Interfaces with SMD-Lite for message transport between host (WDI) and HAL
- Provides the execution context needed by HAL to run

## ■ HAL (Hardware Abstraction Layer)

- Provides a message-based interface to UMAC/MLMS SW to configure 802.11 MAC/PHY capabilities of WLAN ASIC
- To function, HAL requires:
  - ◆ At its northbound, a messaging interface with the host
  - ◆ At its southbound, a messaging interface to the SLM, a mechanism to access HW registers, as well as a mechanism for HW interrupt notification

## ■ SLM (Software Lower MAC)

- Maintains a pure messaging interface with HAL
- Implements key (time-critical) features like Beacon Mode Power Save, BTC, RA, etc.

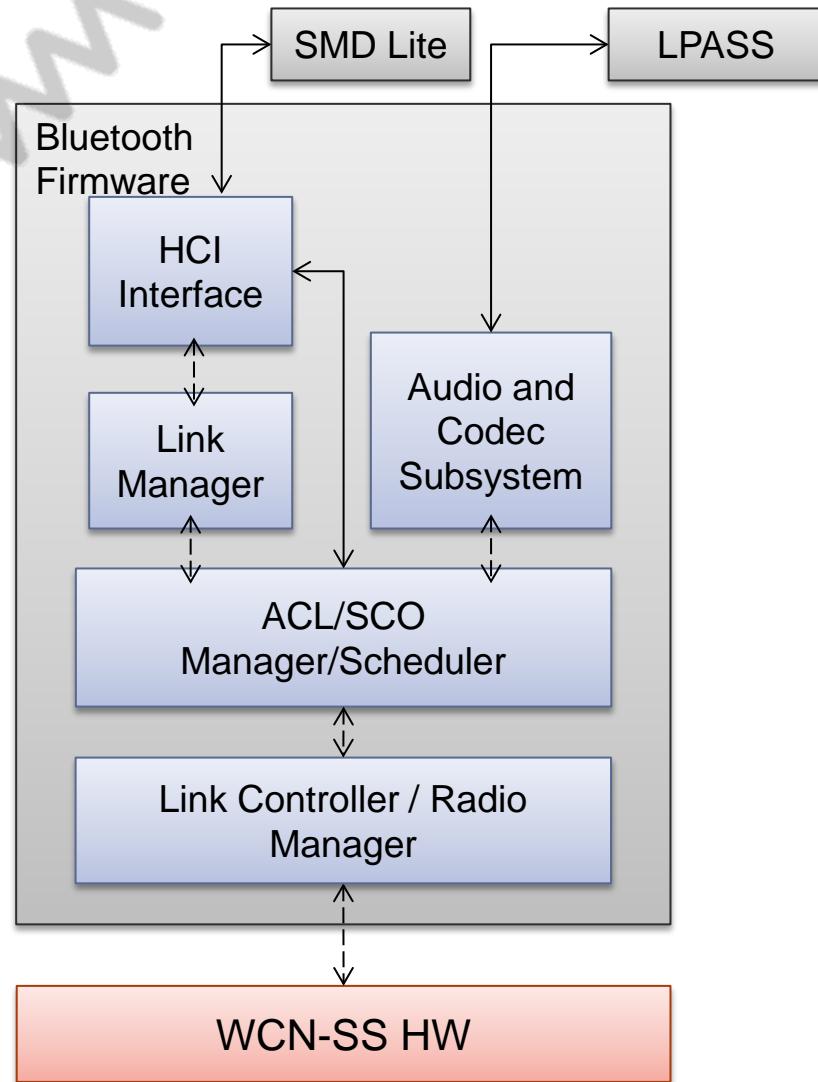


# WCN-SS WLAN HAL

- Exposes the interfaces to the upper MAC running on the host CPU
- Functionality
  - Provides a uniform approach to interfacing with the HW
  - Responsible for
    - ◆ ASIC HW reset and initialization
    - ◆ Configuration and setup of individual WLAN H/W modules (TPE, RPE, BMU, ADU, DXE, PHY, RF)
    - ◆ Interrupt handling
    - ◆ Firmware interface and management
    - ◆ Statistics gathering and update
  - Calibrations

# WCN-SS Bluetooth Software Diagram

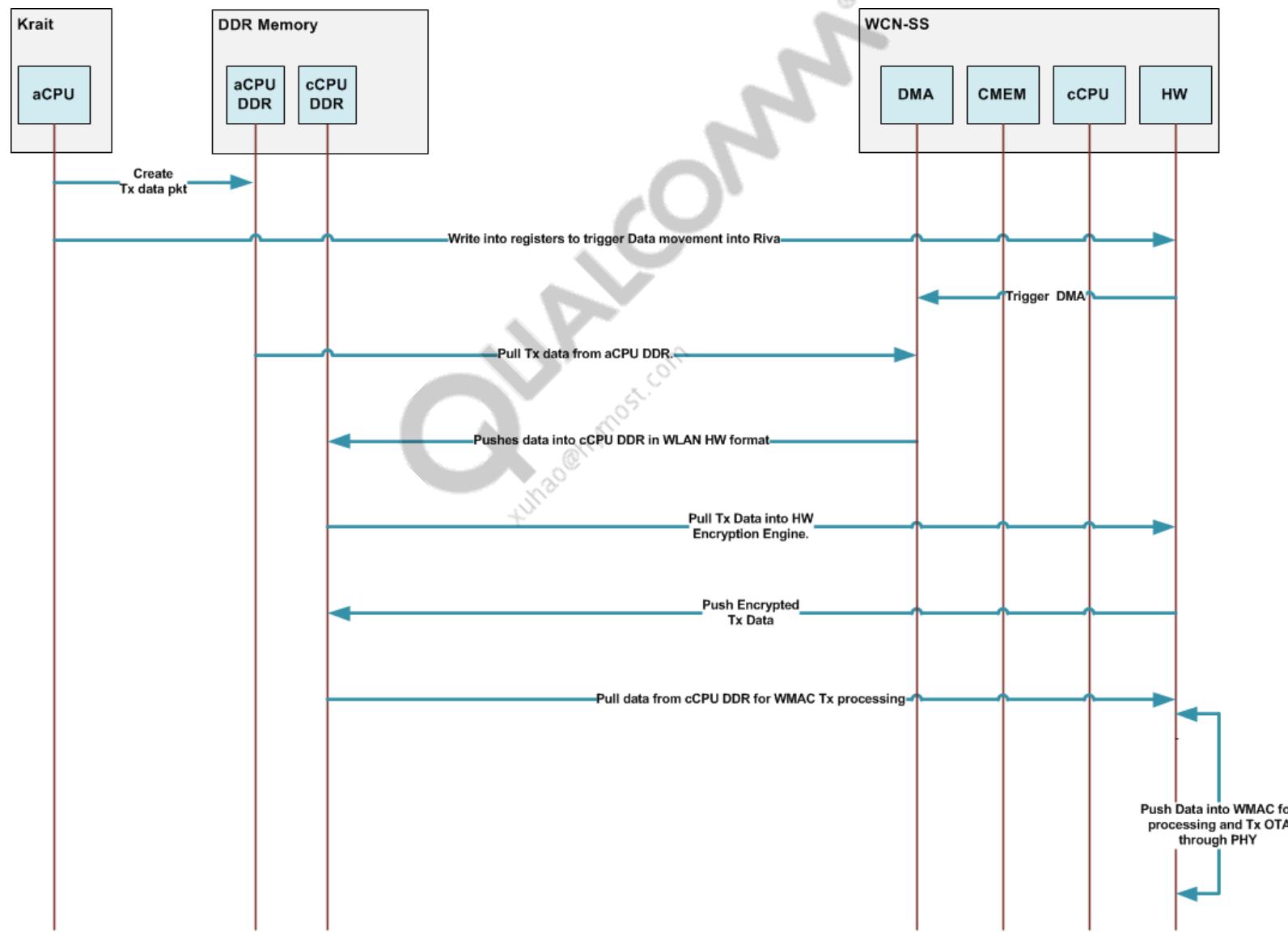
- HCI Interface
  - Interfaces with SMD-Lite for message transport between host (BT Stack) and HCI
- Link Manager
  - Implemented LLM(Logical Link Manager) protocol from BT Spec
  - Security, Authentication handlers
- Audio and Codec SubSystem
  - CVSD codec supported
  - Interfaces with LPASS module to retrieve PCM audio data
  - CMEM is buffer to encode/decode audio data
- Scheduler
  - Maintains all Bluetooth traffic
    - ◆ Page/Inquiry scans
    - ◆ ACL / SCO / BLE packets
- Link Controller / Radio Manager
  - Implement HW dependent codes



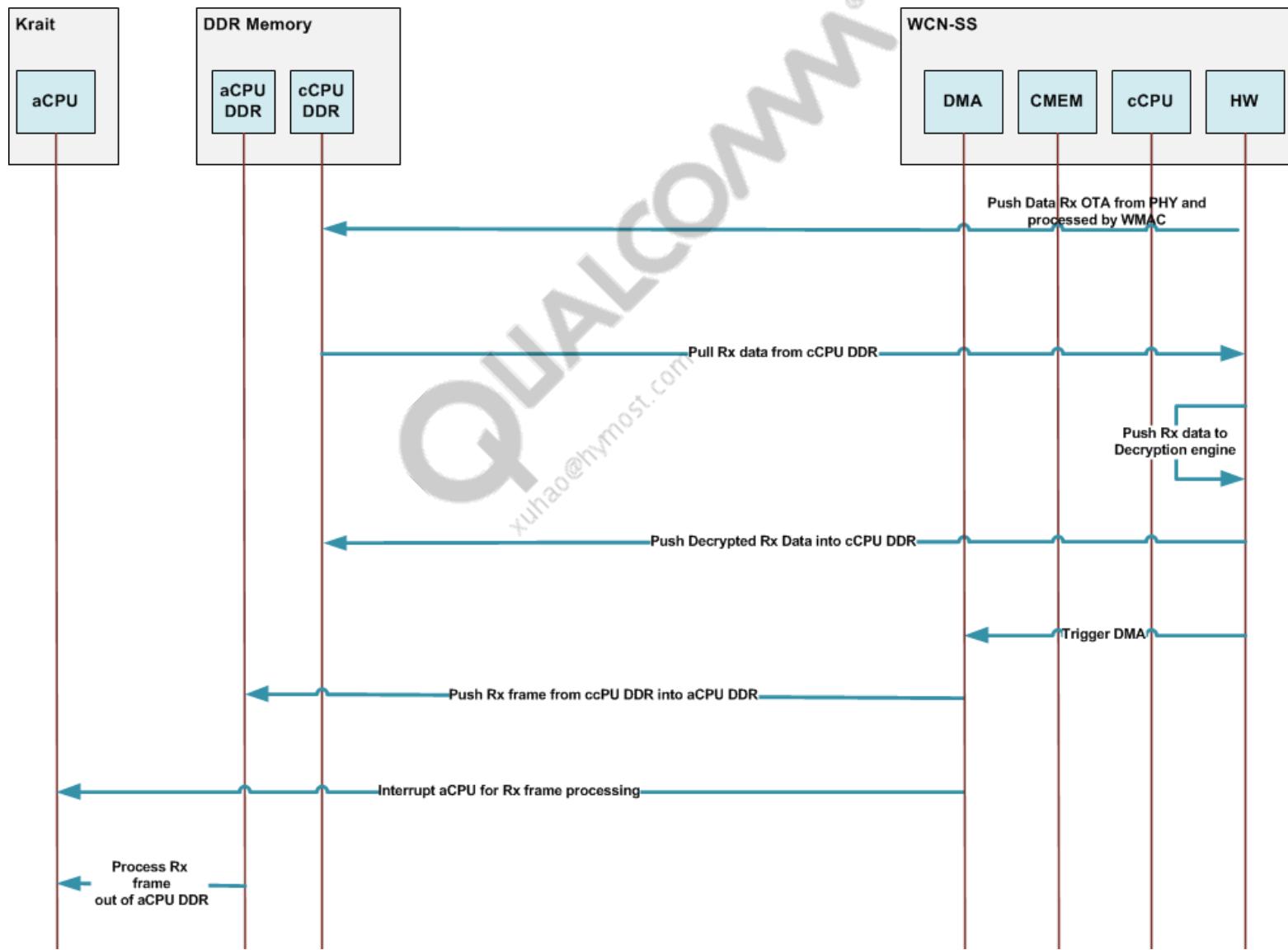
REDEFINING MOBILITY

## Data Flow

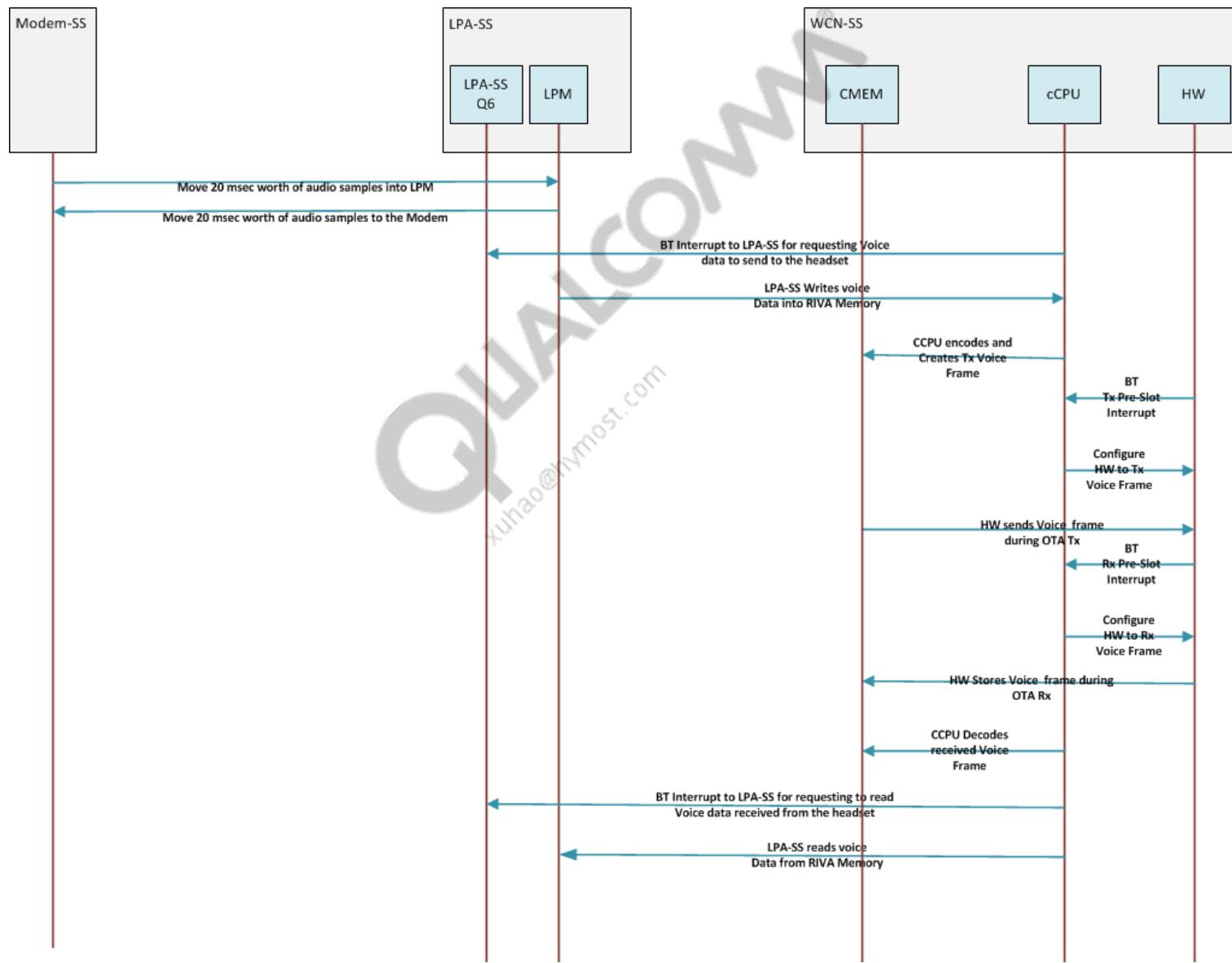
# WLAN – Tx Data



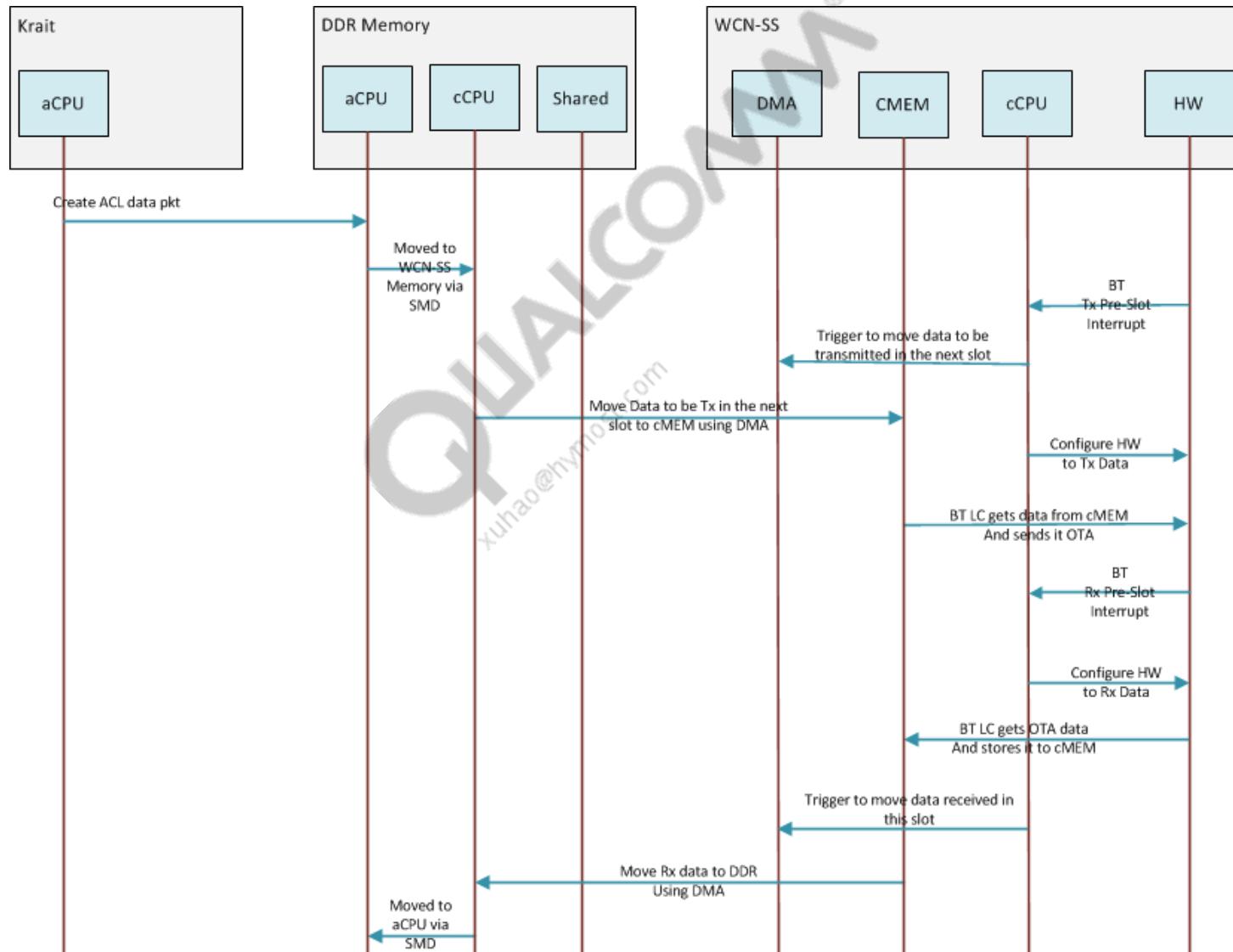
# WLAN – Rx Data



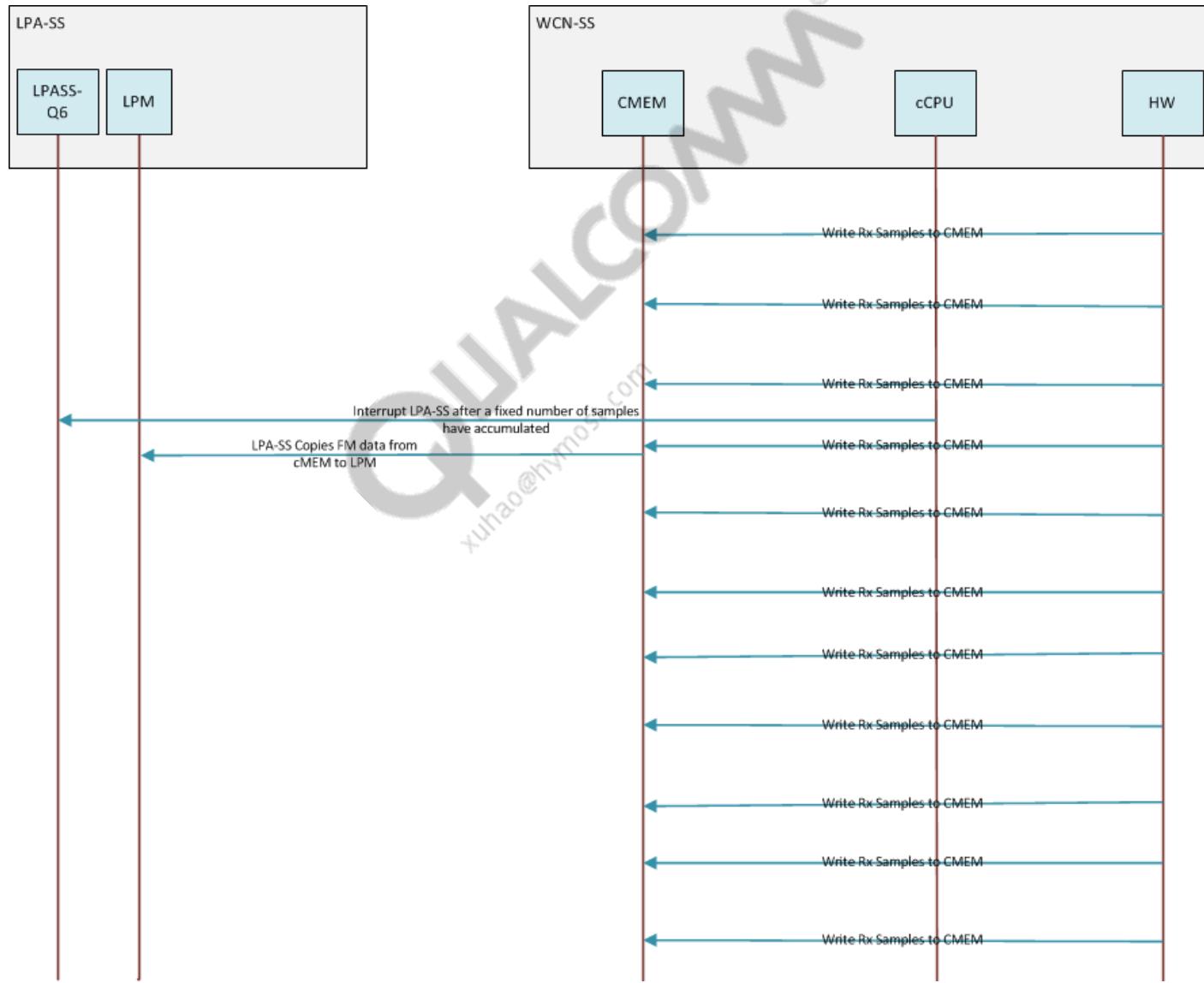
# BT – Voice Call



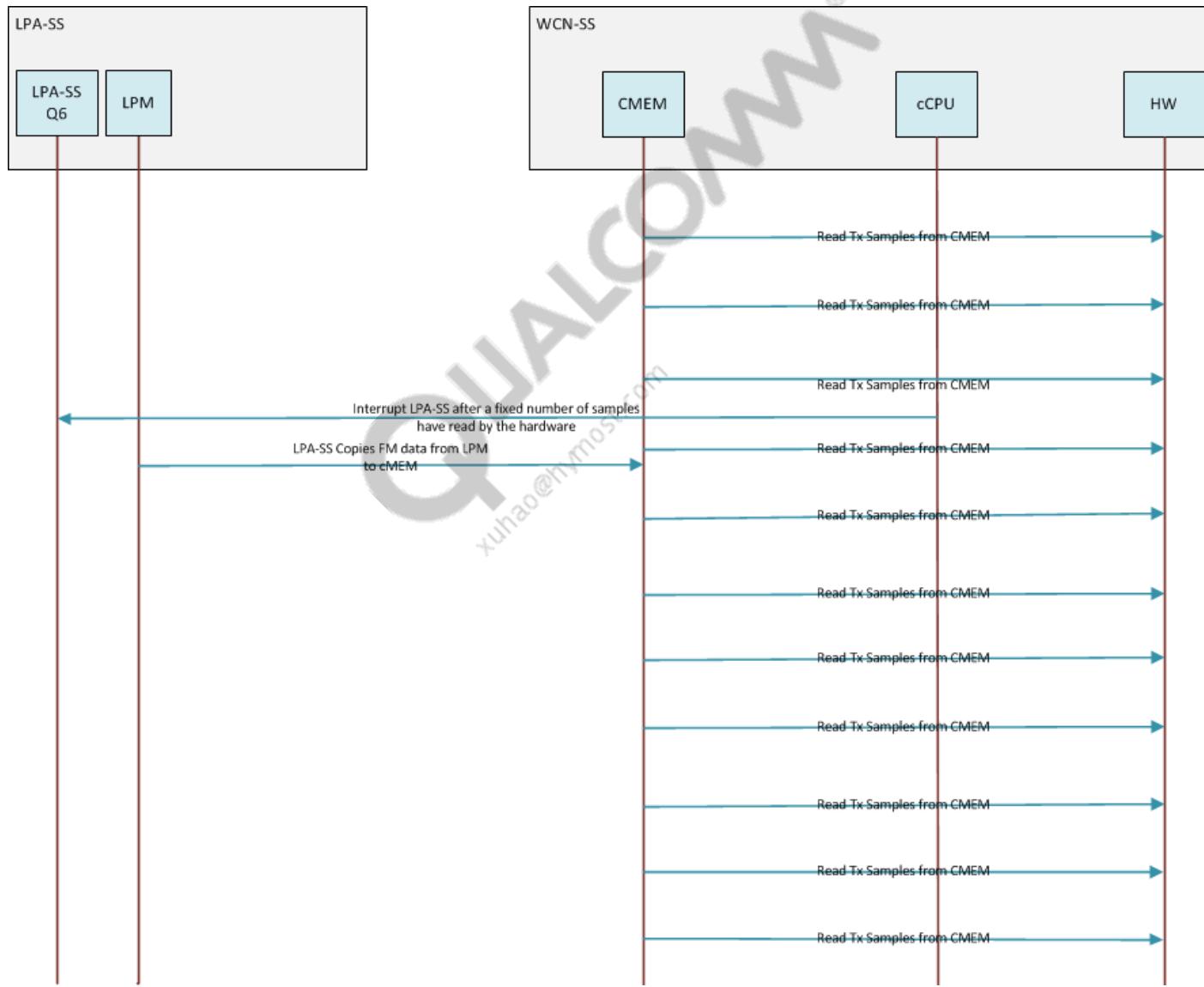
# BT – ACL



## FM Rx Use Case



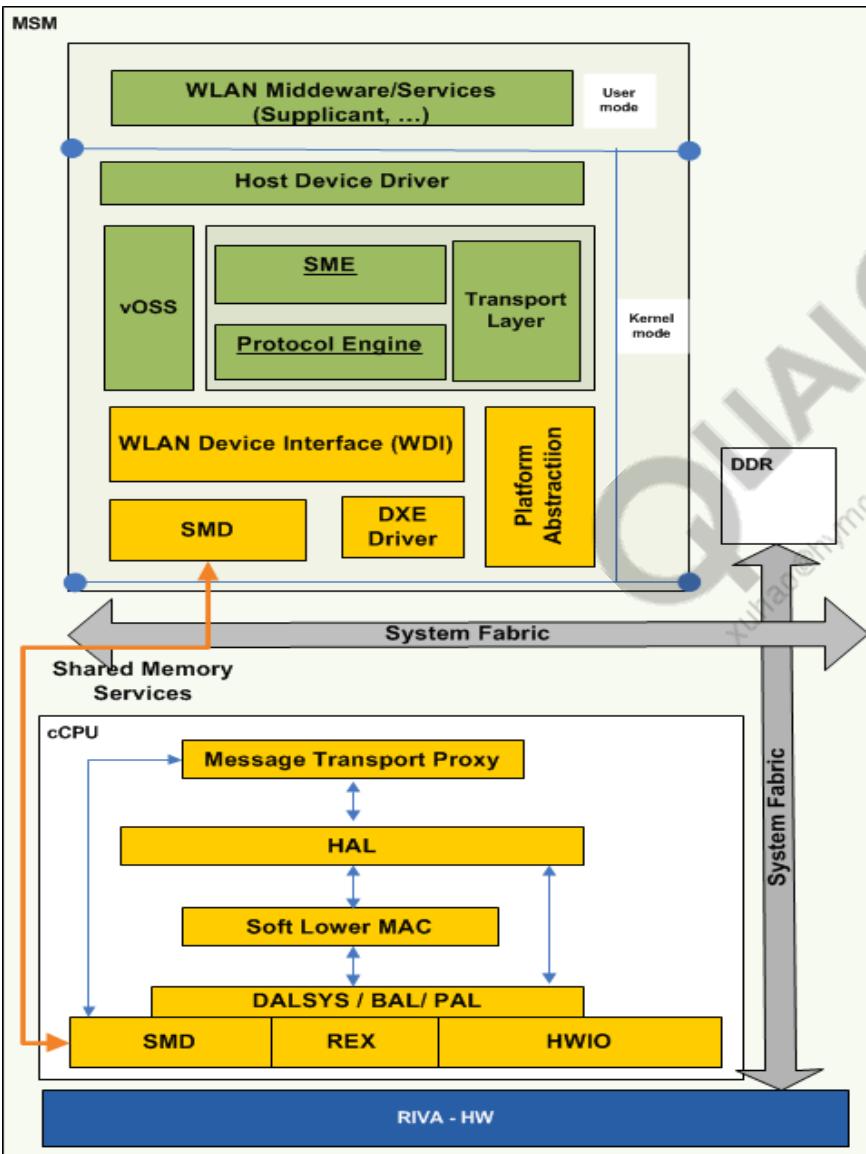
# FM Tx Use Case



# Android WLAN Host Software Architecture

REDEFINING MOBILITY

# WLAN SW High-Level Architecture



- Host device driver
  - Shim layer that interfaces HOST SW to the HLOS specific network driver interface
- Station management entity
  - SW layer responsible for 802.11 SME functionality
- Protocol engine
  - SW layer responsible for 802.11 MLME functionalities
- Transport layer
  - SW that processes data frames that are received and to be transmitted
- DMA driver
  - SW that moves data between system memory and WLAN HW
- HAL
  - SW layer that does all HW control functionalities
- Soft lower MAC
  - SW that implements low-level MAC functionalities – e.g., logic for power save, co-existence, RF cal etc.
- Three major threads
  - Management and Control (MC) Thread: serve messages from modules
  - Tx Thread: handle Tx packets
  - Rx Thread: handle Rx packets

# Host Device Driver - HDD

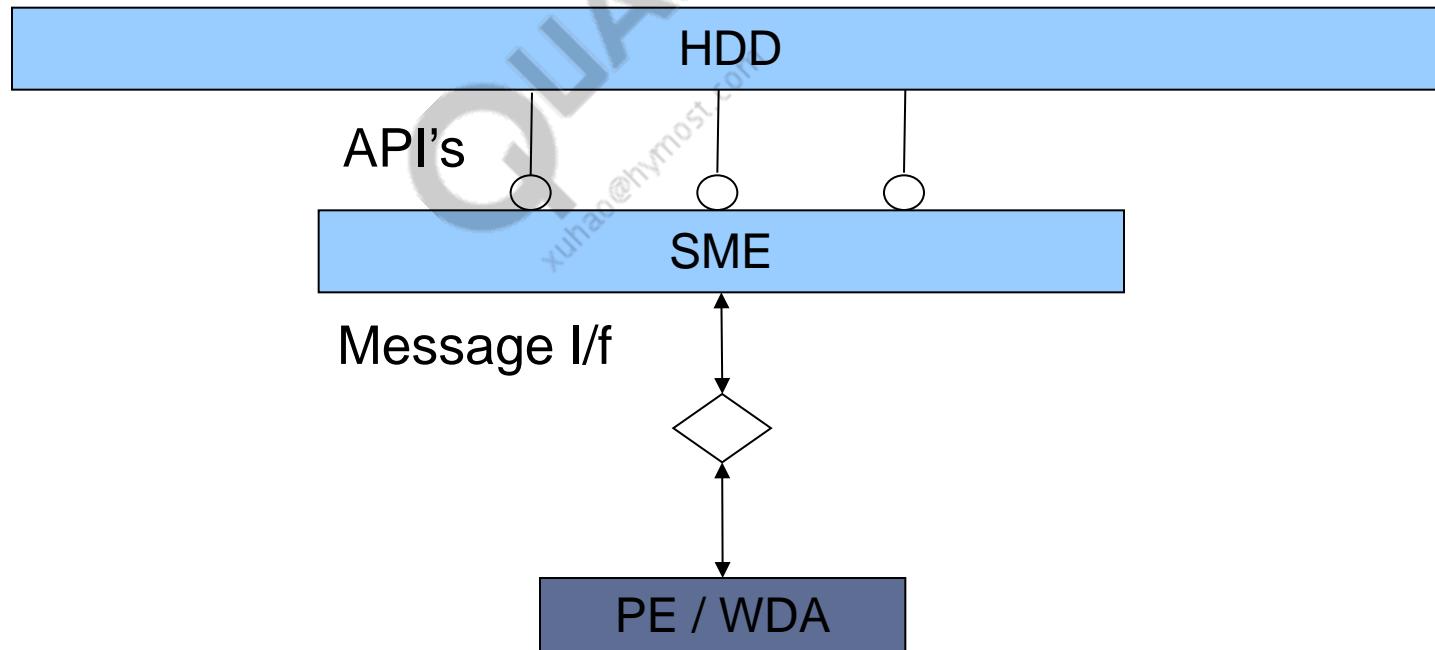
- Driver initialization
  - module\_init, module\_exit, register net device and power manangement callback: wlan\_hdd\_main.c
- Interface with wpa\_supplicant over
  - cfg80211 interface for normal operation: wlan\_hdd\_cfg80211.c
  - wext interface for private IOCTL commands: wlan\_hdd\_wext.c
- Interface with kernel for
  - Tx and Rx over socket buffer: wlan\_hdd\_tx\_rx.c, wlan\_hdd\_softap\_tx\_rx.c
  - Power management handling on early suspend, suspend, resume and late resume: wlan\_hdd\_dev\_pwr.c, wlan\_hdd\_early\_suspend.c
- Read configuration setting from WCNSS\_qcom\_cfg.ini file
  - wlan\_hdd\_cfg.c
- FTM
  - Route RF FTM commands to RIVA or handle NV operation commands: wlan\_hdd\_ftm.c

# Station Management Entity – SME

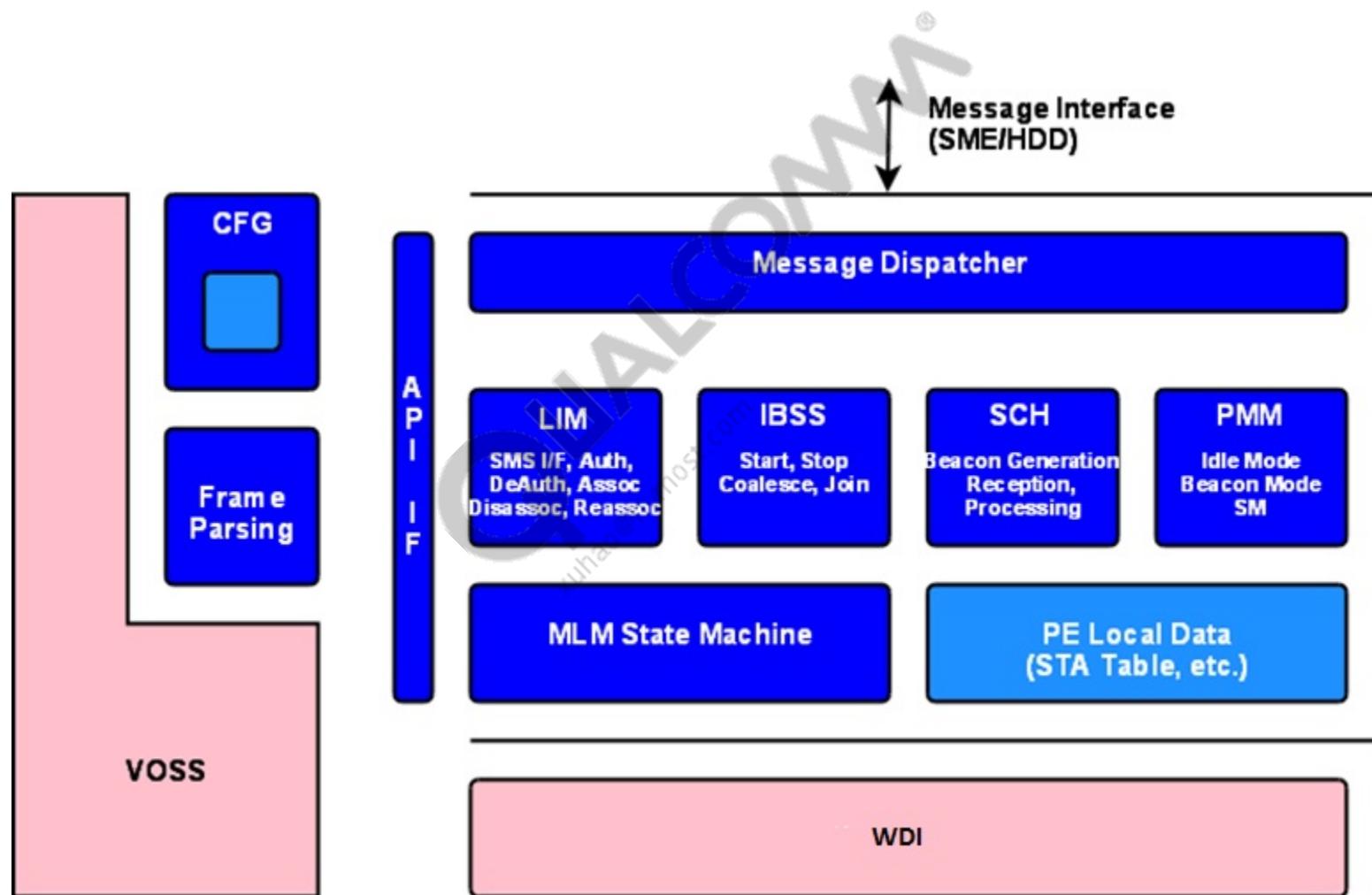
- Station Management Entity
  - Independent of the underlying OS
  - Independent of the HW target/device
- Provides an abstraction layer to access the features supported by the MAC layer
- Primarily services the HDD
- Services currently identified are
  - CSR (Common Scan and Roam)
  - PMC (Power Management Control)
  - BT-AMP (Bluetooth AMP)
  - QoS

# SME Interface Model

- Exposed interfaces are
  - Northbound → APIs/callback from/to HDD, runs in the context of the caller
  - Southbound → Message-based, receive/post message from/to related modules



# PE Architecture



# PE – Components and Functionalities

- Exposes the following interfaces
  - Interfaces are message-based
  - Northbound interface – messages from Host Device Driver (HDD) and Station Management Entity (SME)
  - Southbound interface – messages to WLAN Device Adapter (WDA)
- Link Initialization Module (LIM)
  - Management Frame Processing
  - Station Management – Handling of Authentication, (Re)Association, Disassociation, and Deauthentication
  - Link Monitoring
  - Scan - Handling of Probe Request / Response and Beacon, Scan channels and channel dwell time, Maintain scan hash table
- Power Management Module (PMM)
  - Support for various power save mode: IMPS, BMPS, UAPSD

# PE – Components and Functionalities (cont.)

- Configuration Management (CFG)
  - Contains all configuration values set by user via registry settings, or statically initialized (WCNSS\_cfg.dat)
- Scheduler Module (SCH)
- Automation of Frame Generation and Parsing
  - 802.11 frames generation and parsing
  - Generates code from frame specification text file
  - Takes care of endianess
- IBSS
  - Start, Stop, Coalesce

# WDA (WLAN Device Adaptation)

- Glue between WDI and UMAC
  - UMAC (e.g. mac802.11) will not directly plug into WDI Core
  - WDA adapts UMAC to underlying WDI interfaces/APIs
  - UMAC posts message to WDA for requests
  - WDA posts message to UMAC for response/indication
- Control Request (WDA → WDI)
  - WDA marshals request from UMAC and hands it to WDI using procedural (function) calls
  - WDI APIs will execute in same context in which WDA invokes the WDI APIs
- Control Response/Indication (WDI → WDA)
  - WDA receives callbacks from WDI

# WDI (WLAN Device Interface)

- The main goal of the WDI is to advertise a common interface to access the Prima WLAN device in an Upper MAC agnostic way; a different UMAC should be able to adapt to the WDI interface
- WDI effectively shields the UMAC from the underlying bus used to access the WLAN HW
- WDI encapsulates all processing that is HW-aware, but does not access HW registers (BD Processing, QoS)
- A major design goal for the WDI is to provide on its southbound an interface to adapt to the mechanism used for control messages and data frames transport
  - The **Control transport** layer bridges the WDI into the low level mechanism to transport control messages to HAL
    - ◆ On Prima, the Control Transport Layer implementation interfaces with SMD-Lite to carry message to the HAL
  - The **Data transport layer** bridges the WDI into the low level mechanism to transport data frames to the WLAN BB
    - ◆ On Prima, the Data transport implementation interfaces with the DXE driver

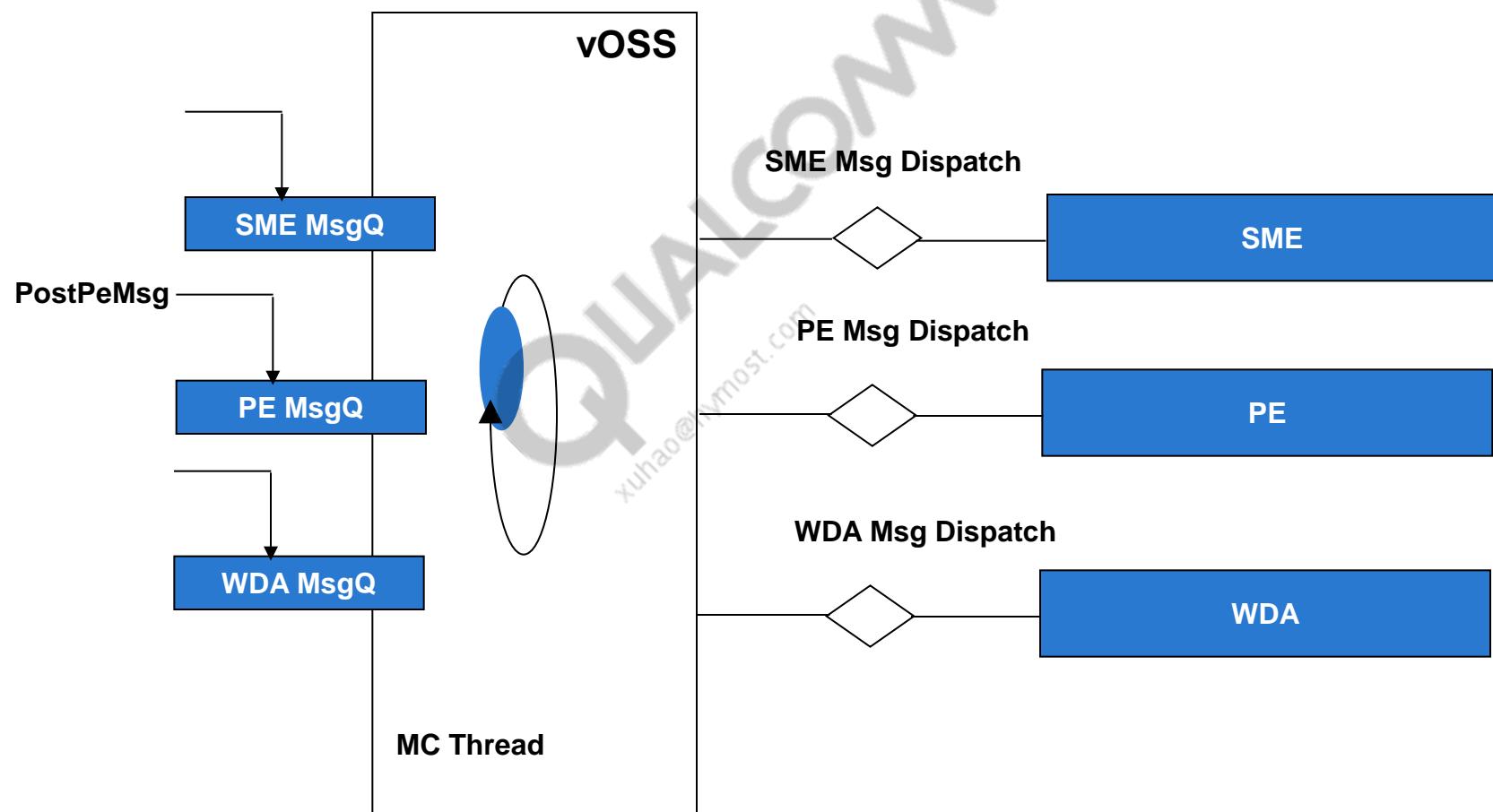
# Transport Layer (TL)

- Provides transport services between MSM device and WLAN module
  - Supports both data and management frames
- Support multiple instances of STA
  - Concurrent infra and P2P link for instance
- Main functionality
  - Rx packet routing to PE, SME and HDD
  - MSDU reordering on the receive path
  - Per-flow tx packet scheduling (optional: QoS flow prioritization)
  - Path control based on the connectivity state of an STA

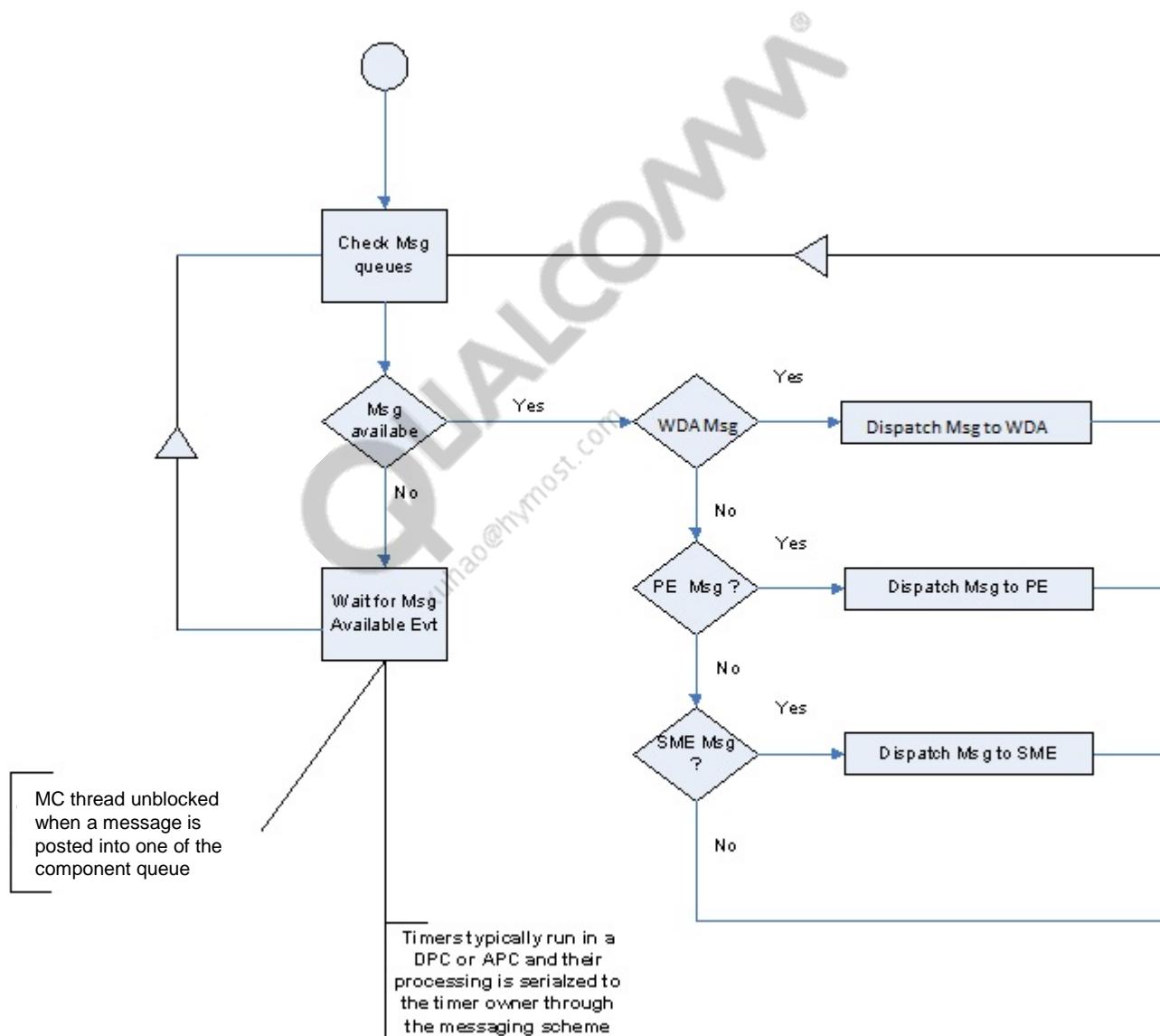
# Virtual Operating System Services (vOSS)

- API providing abstraction for operating system services
  - Allows other components to write portable source code
    - ◆ vOSS specific implementation for Windows Mobile, Linux
  - APIs derived from the POSIX standard, where appropriate
- vOSS APIs provide
  - Basic data types
  - Threading and synchronization (event, mutex, semaphore)
  - Intermodule messaging (message queues)
  - Memory management, clocks and timers
  - Queues, linked lists
  - Network protocol buffer management (transfer buffers)
  - Firmware and ‘option’ (configuration) retrieval
  - Tracing (debug and logging services)

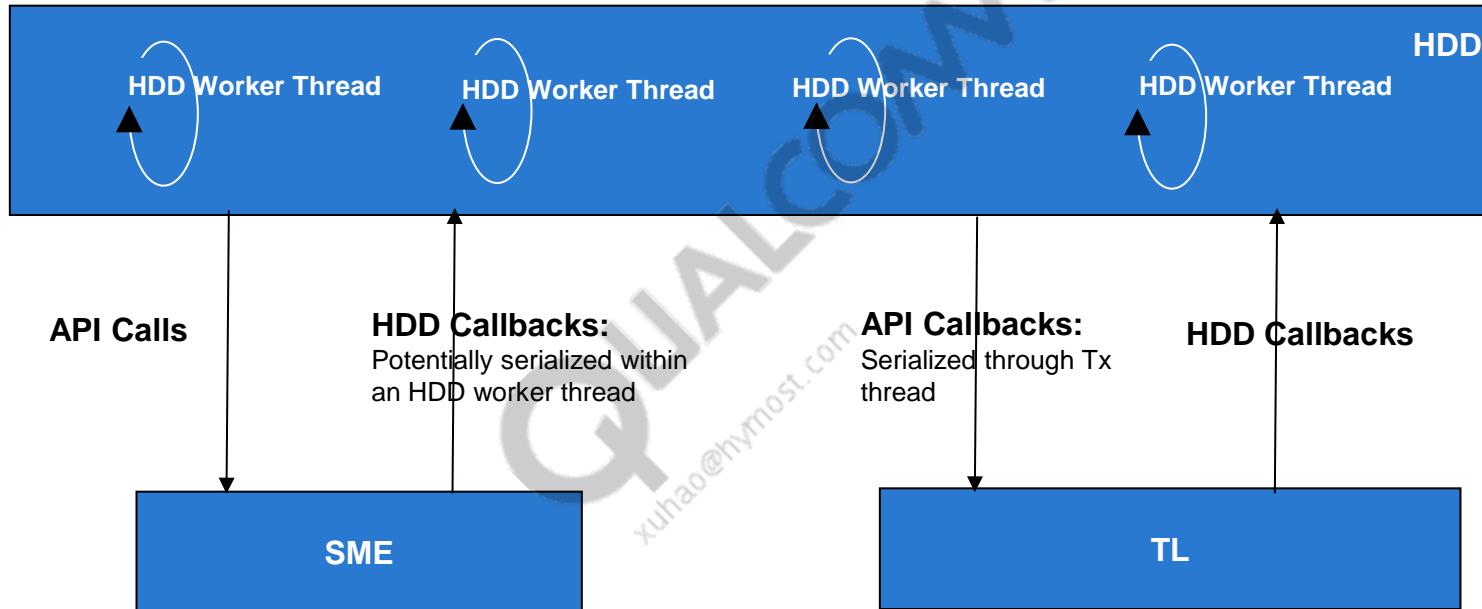
# Execution Model – Control Path



# MC Thread - Message Dispatching



# Interaction with HDD



## TL/HDD Interaction:

**Rx path:** TL executes in Rx Thread context and delivers Rx buffers to HDD in the same execution flow.

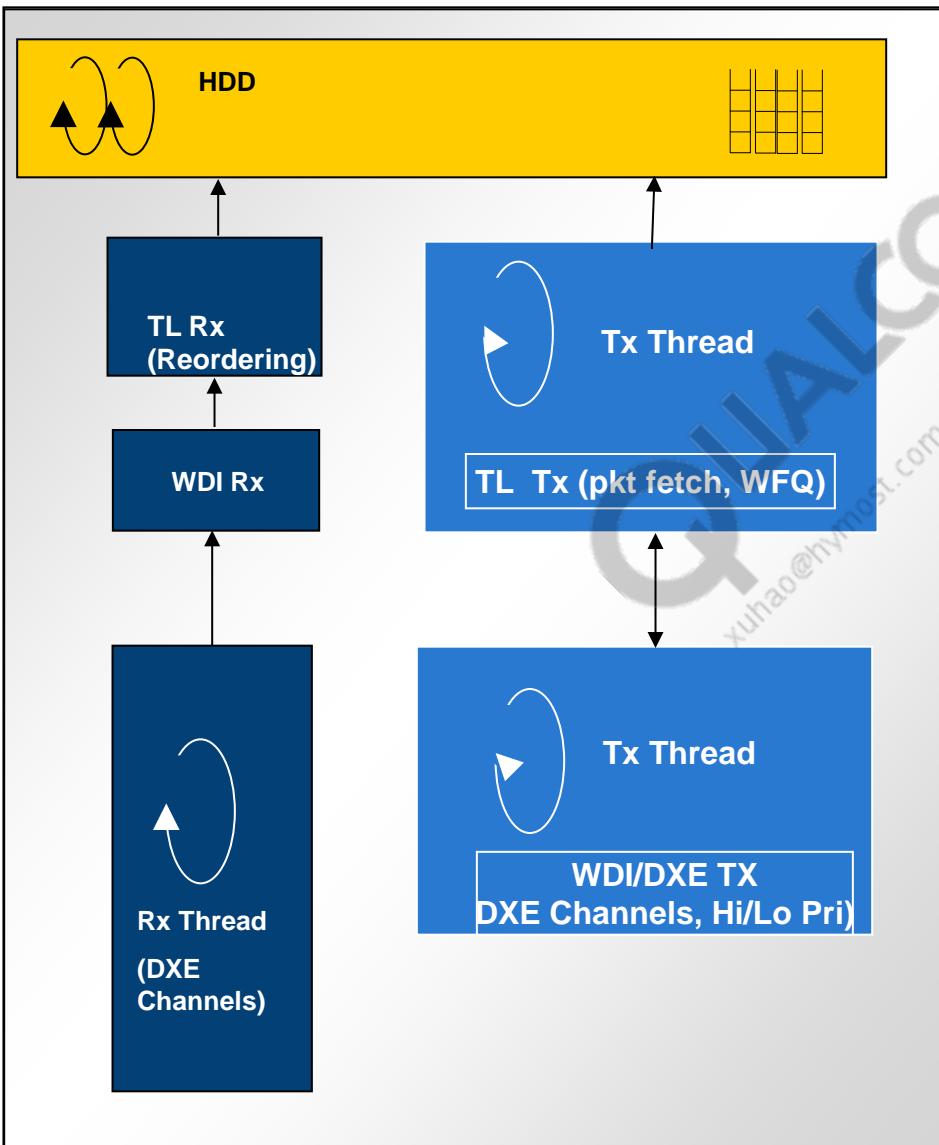
### Tx path:

HDD maintains packets queues

A data Tx request is serialized to TL through the Tx thread

HDD registers a callback with TL to fetch Tx packets executes

# Data Path Execution Model



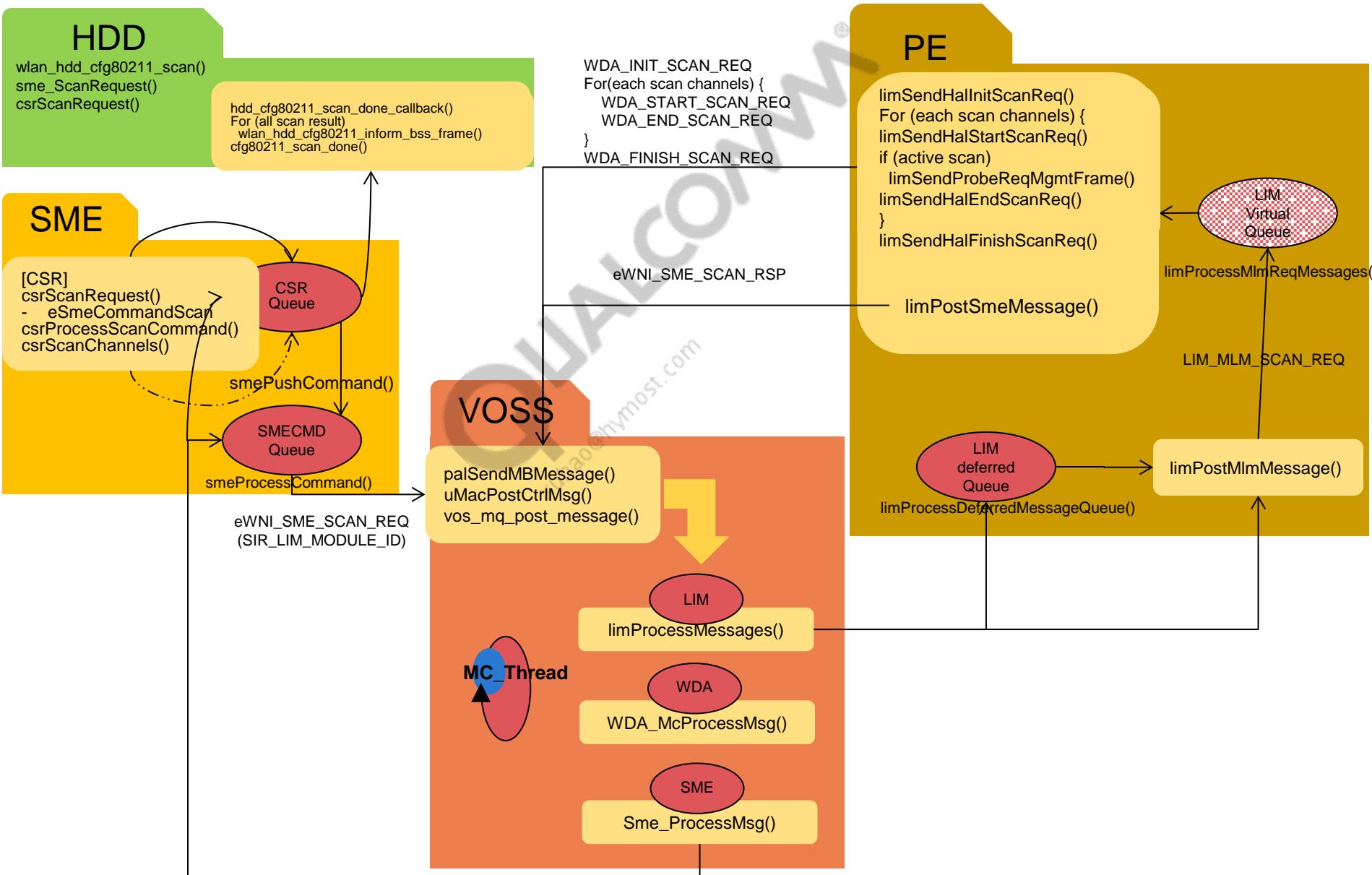
## ■ Tx Path

- HDD maintains per STA/per AC queues
- Notifies TL when pkts are available in the context of OS threads
- TL fetches packets in TX thread context after applying WFQ algo
- DXE driver puts the packet in the DXE ring for DMA
- Mgmt pkts go to Hi Pri channel, data pkts go to Lo Pri Channel

## ■ Rx Path

- DXE Interrupt handler notifies RX thread
- RX thread pulls the packet from DXE rings and hands it over to WDI/TL
- TL does BA reordering before pushing the packets to HDD/OS

# Scan Control Path Execution Flow (No Concurrency)

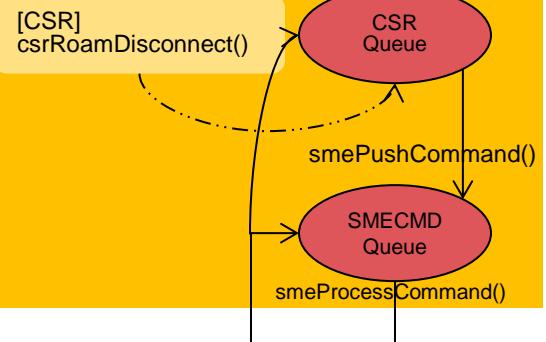


# Disconnect Control Path Execution Flow

HDD

```
wlan_hdd_cfg80211_disconnect()
sme_RoamDisconnect()
csrRoamDisconnect()
```

SME



VOSS

```
palSendMBMessage()
uMacPostCtrlMsg()
vos_mq_post_message()
```

MC Thread

LIM  
limProcessMessages()

WDA  
WDA\_McProcessMsg()

SME  
Sme\_ProcessMsg()

PE

limDelBss()

limPostSmeMessage()

LIM  
Virtual  
Queue

limProcessMlmReqMessages()

LIM\_MLM\_DISASSOC\_REQ

LIM  
deferred  
Queue

limPostMlmMessage()

limProcessDeferredMessageQueue()

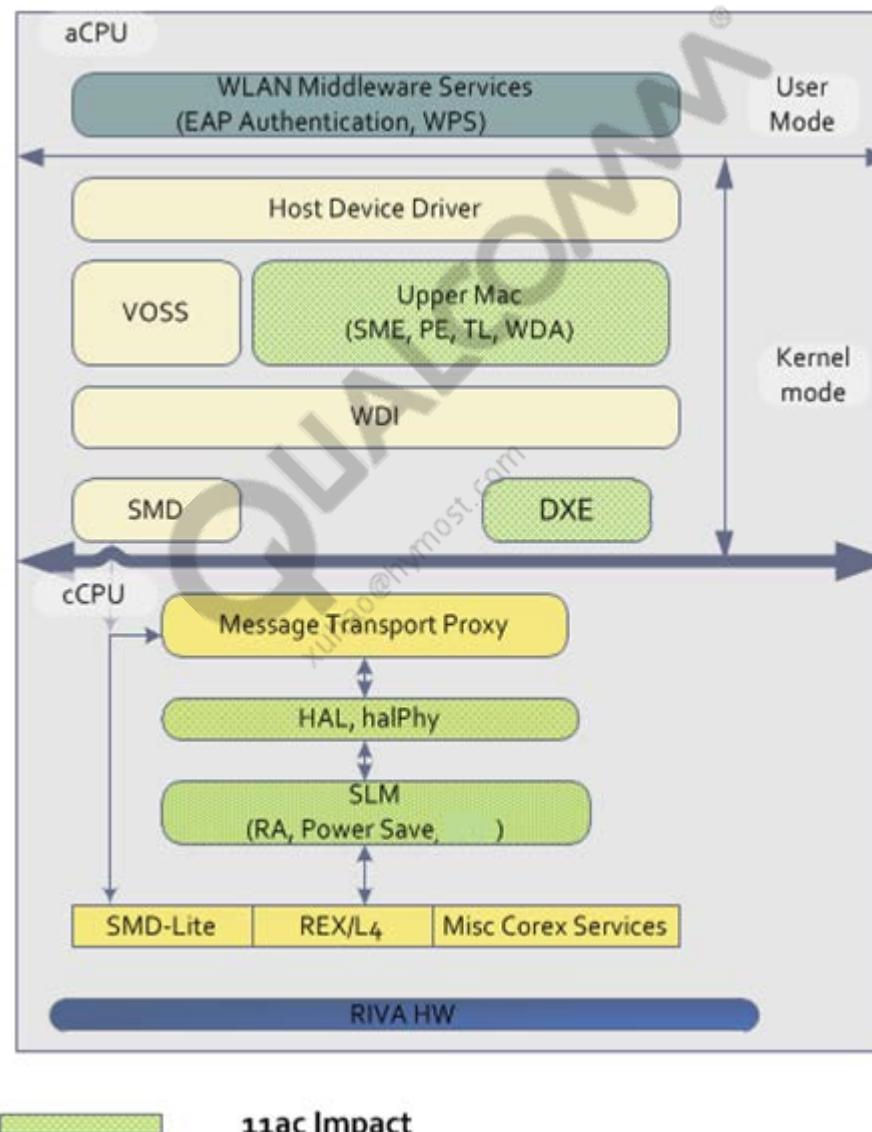
WDA\_DELETE\_BSS\_REQ

LIM\_MLM\_DISASSOC\_CNF

## 802.11ac Software Integration

REDEFINING MOBILITY

# SW Modules Impacted by 11ac



## 11ac Impact – Upper MAC

- Added support for VHT IEs in the mgmt frames parser
- Added VHT Capability IE and VHT Operation IE support in all management frames e.g., Probe Req/Rsp, Assoc Req/Rsp, Beacons, etc.
- VHT capability negotiation with the peer during association
- Updated CFG module with VHT-related configs
- Interface changes between SME→PE→WDA→WDI→HAL to pass additional parameters related to VHT
- Periodic Beacon Parsing to detect dynamic channel bonding mode change or channel switch in VHT IEs

# 11ac Impact – DXE Driver

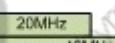
- Rx side
  - Reap DXE descriptor ring before processing
    - ◆ Allows hardware and software to operate concurrently
  - Pull packets from DXE descriptor ring on a per packet interrupt basis (ED interrupt) rather than when the ‘DONE’ interrupt is received
    - ◆ dxeRXEventHandler add ED (external descriptor) interrupt handling to allow host SW to pull packets faster
    - ◆ DONE interrupt indicates the DMA channel has finished performing all the programmed transfers
    - ◆ ED interrupt indicates DXE has finished descriptor transfer, per packet basis. (defined in WLАНDXE\_DescType, bit 17, intr :1; //Interrupt on Descriptor Done)
  - Increase ring size – 11ac has higher performance requirement
    - ◆ Allow more head room for software to process (packet/sec is higher)
    - ◆ chanRXLowPriConfig from 40 to 256, chanRXHighPriConfig from 10 to 40

## 11ac Impact – Host (Misc.)

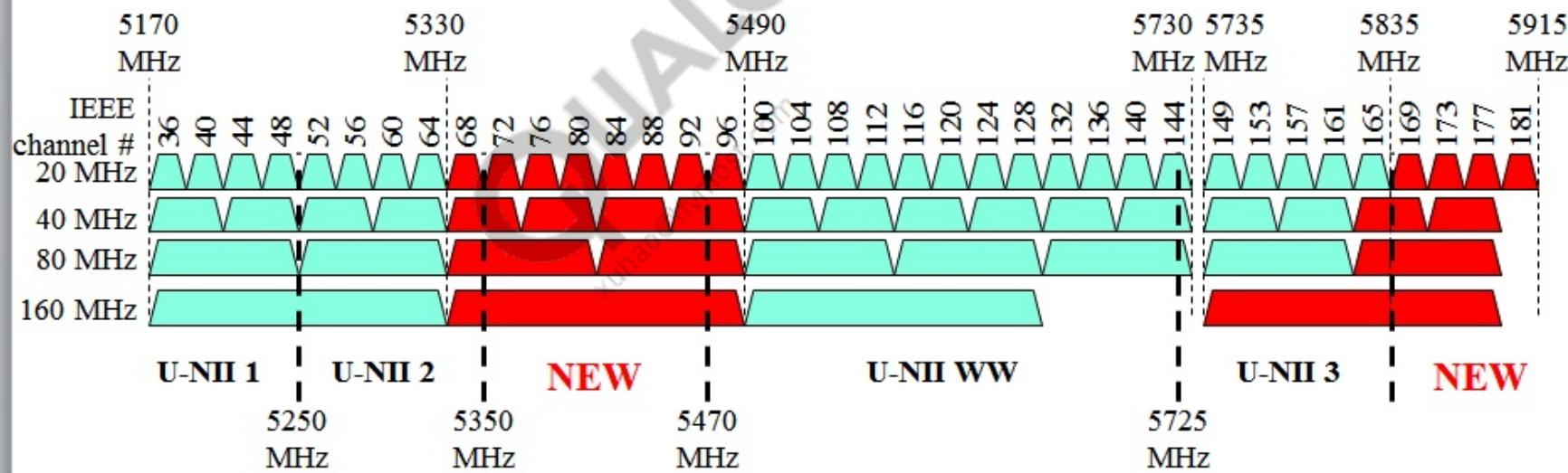
- NV.bin is changed due to additional VHT rates
- WCNSS\_qcom\_cfg.ini has new parameters
  - VHT channel width – gVhtChannelWidth
    - ◆ 1 to 20 and 40 MHz
    - ◆ 2 to 80 MHz (default)
  - VHT Tx/Rx MCS values – gVhtTxMCS, gVhtRxMCS
    - ◆ 0 – MCS0-7
    - ◆ 1 – MCS0-8
    - ◆ 2 – MCS0-9 (default)
  - Channel Bonding Mode – gChannelBondingMode5GHz
    - ◆ In STA mode
      - ▶ 0 – disable channel bonding
      - ▶ 1 – enable channel bonding capability
    - ◆ In SoftAP mode
      - ▶ 0 – disable channel bonding
      - ▶ 1 – enable dynamic channel bonding which driver would automatically decide what channel bonding mode to use based on primary channel information

# 11ac Impact – New CB Modes

- In SoftAP mode, driver would automatically select right channel bonding mode based on the primary channel information if gChannelBondingMode5GHz is set to 1 to enable dynamic channel bonding

Channel Bandwidth	Subband Location	Channel Set
20Mhz		All 5Ghz Channels
40Mhz		40, 48, 56, 64, 104, 112, 120, 128, 136
40Mhz		36, 44, 52, 60, 100, 108, 116, 124, 132
80Mhz		36, 52, 100, 116, 149
80Mhz		40, 56, 104, 120, 153
80Mhz		44, 60, 108, 124, 157
80Mhz		48, 64, 112, 128, 161

# 11ac Impact – Available 80 MHz Channels



# 11ac Impact – FTM

- QRCT Tool
  - QRCT tool has been updated to support the following
    - ◆ New 11ac mode (in addition to 11b/g/n)
    - ◆ New 11ac rates
    - ◆ 80 Mhz modes
- halPhy module
  - halPhy module has been updated to transmit AMPDUs (instead of MPDUs for 11n)

# 11ac Impact – WCN-SS in RIVA

- Interface change between host and FW due to additional VHT parameters
- Added support for VHT configs to MAC and PHY modules
- Added support for 80 Mhz modes during channel change
- HAL internal STA and BSS tables updated to store additional VHT parameters
- HAL-RA interfaces updated to pass additional VHT rates
- Added support for 11ac rates in TPE Rate Table

# 11ac Impact – SLM

- Power Save
  - ADU list has been updated with additional 11ac specific MAC/PHY registers so these 11ac configs get restored after wakup
- Rate adaptation enhancements
  - Leverage existing infrastructure from RIVA rate adaptation
  - No change in fundamental algorithms
  - Extend the HAL rate table to support 11ac rates.
  - Extended the HAL rate table to support 11ac 80Mhz duplicate rates for control packets
  - Update rate properties and extend sampling rate table for VHT rates
  - Use only 11ac rates if peer supports 11ac rates
    - ◆ Every 11n rate has an equivalent 11ac rate at same modulation class, so no need to use 11n rates on Tx side
    - ◆ VHT preamble already has HT-SIG to shut out 11n devices
    - ◆ The only additional rate which 11a can offer is 6Mbps vs 6.5 Mbps for VHT20 MCS0
  - Ensure 11n/11ac rates are not mixed with 11ac rates as sampling candidate
  - Ensure 11n/11ac rates are not mixed with 11ac rates as retry rates
  - New field in VHT cap to advertise support from MCS 0-7, 0-8 and 0-9 – host does parsing, riva firmware does mapping to construct the bitmap
  - Leverage dynamic switching of HT20/40/80 to take advantage of hardware capabilities – no CCA counter-based coarse switching at present

# 11ac Impact – Additional PHY Rates

	20 Mhz (11ac MCS0-7)				40 Mhz (11ac MCS0-9)				80 Mhz (11ac MCS0-9)			
	Normal GI		Short GI		Normal GI		Short GI		Normal GI		Short GI	
	Mbps	RateIdx BCC	Mbps	RateIdx BCC	Mbps	RateIdx BCC	Mbps	RateIdx BCC	Mbps	RateIdx BCC	Mbps	RateIdx BCC
BPSK, rate 1/2	6.5	66	7.2	78	13.5	90	15	102	29.25	114	32.5	126
QPSK, rate 1/2	13	67	14.4	79	27	91	30	103	58.5	115	65	127
QPSK, rate 3/4	19.5	68	21.7	80	40.5	92	45	104	87.75	116	97.5	128
16 QAM, rate 1/2	26	69	28.9	81	54	93	60	105	117	117	130	129
16 QAM, rate 3/4	39	70	43.3	82	81	94	90	106	175.5	118	195	130
64 QAM, rate 2/3	52	71	57.8	83	108	95	120	107	234	119	260	131
64 QAM, rate 3/4	58.5	72	65	84	121.5	96	135	108	263.25	120	292.5	132
64 QAM, rate 5/6	65	73	72.2	85	135	97	150	109	292.5	121	325	133
					162	99	180	111	351	123	390	135
					180	100	200	112	390	124	433.33	136

# 11ac Impact – 80 Mhz Duplicate Rates

- Duplicate rates used for transmitting control packets (e.g., Ack, BA, BAR, RTC, CTS etc.) in 80 Mhz mode

802.11 a/g rates (80 Mhz Duplicate)	
Mbps	Rateldx
6	218
9	219
12	220
18	221
24	222
36	223
48	224
54	225

# Space-Time Block Coding (STBC)

- Commonly used in systems where there are multiple transmit chains, but only a single receiver chain, such as a handheld device with a single antenna talking to an Access Point (AP) with multiple antennas
- With STBC, each AP transmitter sends a known linear combination of the other transmitters' signals which the single receiver uses to identify the original data stream
- Both RIVA and Pronto support Rx STBC to improve DL RvR performance
  - No Tx STBC
- Rx STBC capability is conveyed in the HT/VHT Capabilities Info field
  - HT Capability Info Field for Rx STBC – set as 1

Rx STBC	Indicates support for the reception of PPDUs using STBC	Set to 0 for no support Set to 1 for support of one spatial stream Set to 2 for support of one and two spatial streams Set to 3 for support of one, two and three spatial streams
---------	---	--

- VHT Capability Info Field for Rx STBC – set as 1

Rx STBC	Indicates support for the reception of PPDUs using STBC	Set to 0 for no support. Set to 1 for support of one spatial stream. Set to 2 for support of one and two spatial streams. Set to 3 for support of one, two and three spatial streams. Set to 4 for support of one, two, three and four spatial streams. The values 5, 6, 7 are reserved.
---------	---	---

# Space-Time Block Coding (STBC) – Driver Configuration

- gEnableRXSTBC is the configuration in WCNSS\_qcom\_cfg.ini
  - 1 – enabled HT/VHT Rx STBC (default)
  - 0 – disable HT/VHT Rx STBC
  - Host change:  
<https://www.codeaurora.org/gitweb/external/wlan/?p=prima.git;a=commit;h=6bfd14258b44bff103631070c02eadaa774be485>

# Transmit Beamforming (TxBF)

- Tx Beamforming uses sounding techniques to align the transmitter with the receiver; the transmitter sends a signal (“Where are you?”) and listens for a response from the receiver (“I’m right here!”); by changing the characteristics of the transmission, the transmitter can hone in on the receiver’s location to tune the beam to be as narrow as possible
  - Beamforming is also known as beam steering
- A station that transmits a PPDU using a beamforming steering matrix is called Beamformer
  - Not supported in Pronto and RIVA
- A station that receives a PPDU that was transmitted using a beamforming steering matrix is called Beamformee
- A station that receives PPDU that was transmitted using a multi-user beamforming steering matrix is called Multi-user (MU) Beamformee; a station that receives PPDU that was transmitted using a single-user beamforming steering matrix is called Single-user (SU) Beamformee
  - Supported in Pronto but not RIVA
- Currently, TxBF is not yet widely supported by AP in the market

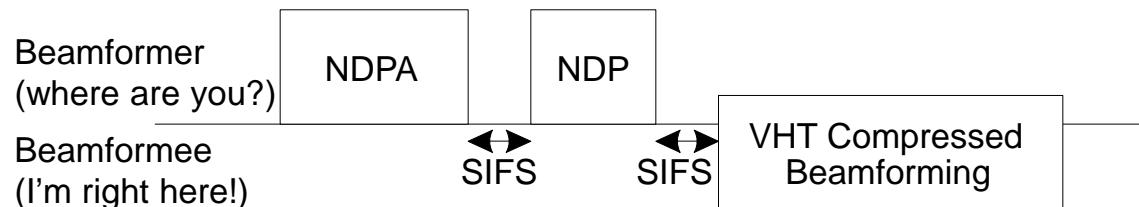
# Transmit Beamforming (TxBF) – VHT Capability Information

- AP shall set the SU/MU beamformer Capable field in the VHT Capabilities element to 1
- STA shall set the SU/MU beamformee Capable field in the VHT Capabilities element to 1
  - As there is limited AP support in TxBF and only SU beamformer is available, Pronto currently enables only SU beamformee; MU beamformee can be enabled in Pronto;

SU Beamformer Capable	Indicates support for operation as an SU beamformer (see 9.31.5 (VHT sounding protocol))	Set to 0 if not supported. Set to 1 if supported.	0: as Pronto does not support beamformer
SU Beamformee Capable	Indicates support for operation as an SU beamformee (see 9.31.5 (VHT sounding protocol))	Set to 0 if not supported. Set to 1 if supported.	1: if feature is enabled by gTxBFEnable in WCNSS_qcom_cfg.ini 0: otherwise
Compressed Steering Number of Beamformer Antennas Supported	The maximum number of space-time streams that the STA can receive in a VHT NDP, the maximum value for $N_{STS, total}$ that can be sent to the STA in a VHT MU PPDU if the STA is MU beamformee capable and the maximum value of $N_r$ that the STA transmits in a VHT Compressed Beamforming frame.	If SU beamformee capable, set to maximum number of space-time streams that the STA can receive in a VHT NDP minus one. Otherwise reserved.	2: if feature is enabled by gTxBFEnable in WCNSS_qcom_cfg.ini. Pronto supports 3 compressed steering of beamformer antennas 0: otherwise
Number of Sounding Dimensions	Beamformer's capability indicating the maximum value of the TXVECTOR parameter NUM_STS for a VHT NDP	If SU beamformer capable, set to the maximum supported value of the TXVECTOR parameter NUM_STS minus 1. Otherwise reserved.	0: if feature is enabled by gTxBFEnable in WCNSS_qcom_cfg.ini. Pronto supports 1 sounding dimension
MU Beamformer Capable	Indicates support for operation as an MU beamformer (see 9.31.5 (VHT sounding protocol))	Set to 0 if not supported or if SU Beamformer Capable is set to 0 or if sent by a non-AP STA. Set to 1 if supported and SU Beamformer Capable is set to 1.	0: as Pronto does not support beamformer
MU Beamformee Capable	Indicates support for operation as an MU beamformee (see 9.31.5 (VHT sounding protocol))	Set to 0 if not supported or if SU Beamformee Capable is set to 0 or if sent by an AP. Set to 1 if supported and SU Beamformee Capable is set to 1.	0: as no MU beamformer AP available

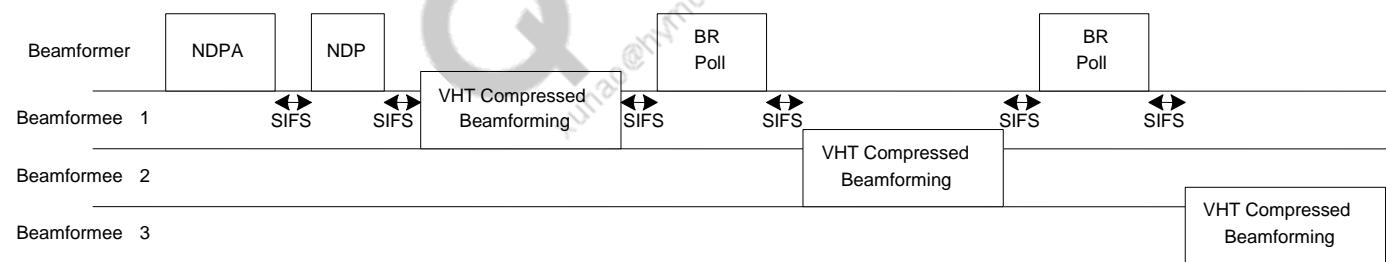
# Transmit Beamforming (TxBF) – Sounding Protocol for Single User

- A VHT beamformer initiates a sounding feedback sequence by transmitting a VHT Null Data Packet (NDP) Announcement frame followed by a VHT Null Data Packet (NDP) after a SIFS
- VHT NDP shall be transmitted only following a SIFS after a VHT NDP Announcement frame
- Upon receiving the NDPA frame, the beamformee waits for NDP and sends the VHT Compressed Beamforming feedback after the SIFS

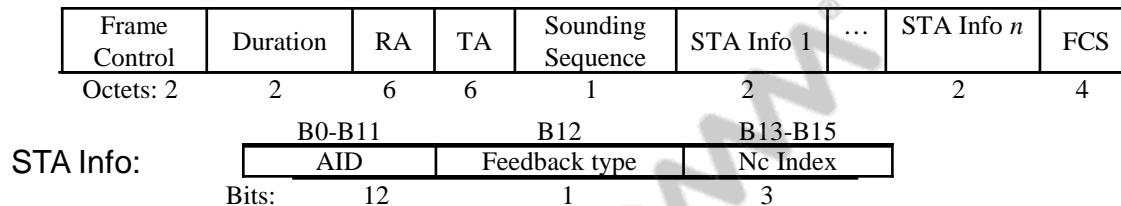


# Transmit Beamforming (TxBF) – Sounding Protocol for Multi-User

- A VHT beamformer that transmits a VHT NDP Announcement frame with more than one station info field should transmit Beamforming Report Poll frame needed to retrieve VHT Compressed Beamforming feedback from the intended VHT beamformees in the same TXOP
- The beamformee sends the VHT Compressed Beamforming feedback after it receives the NDP if the AID matches the first station info; otherwise, the beamformee sends the VHT Compressed Beamforming feedback with the SIFS time after it receives the Beamforming Report Poll from AP



# NDP Announcement Frame



- The VHT NDP Announcement frame includes at least one STA Info field
- The VHT beamformer includes the AID of the beamformee in the AID subfield of the STA info field
- A VHT beamformer that transmits a VHT NDP Announcement frame to a VHT SU-only beamformee includes only one STA Info field in the VHT NDP Announcement frame and sets the Feedback Type subfield of the STA Info field to SU
- If the VHT NDP Announcement frame includes a single STA info field, the RA of this frame is set to the MAC Address of the beamformee
- If the VHT NDP Announcement frame includes more than one STA Info field, the RA of the VHT NDP Announcement frame shall be set to the broadcast address
- A VHT NDP Announcement frame shall not include the same value of the AID subfield

## NDP Frame

- The VHT NDP frame does not contain the MAC header; it includes only the VHT PHY preamble
- The VHT NDP uses the following TXVECTOR parameters
  - APEP\_LENGTH set to 0
  - NUM\_USERS set to 1
  - NUM\_STS indicates two or more spatial streams
  - CH\_BANDWIDTH
  - GROUP\_ID
  - PARTIAL\_AID

# VHT Compressed Beamforming Frame

- The VHT Compressed Beamforming frame is an Action No Ack frame of category VHT

Order	Information
1	Category=21 means VHT action frame
2	Action=0 means VHT Compressed Beamforming
3	VHT MIMO Control
4	VHT Compressed Beamforming Report
5	MU Exclusive Beamforming Report

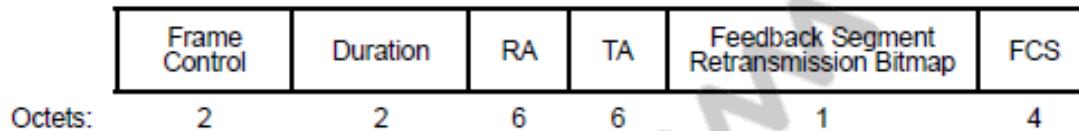
# VHT Compressed Beamforming Frame – VHT MIMO Control

Subfield	Description	Pronto Setting
Nc Index 3 bits	Indicates the number of columns in a matrix minus one: Set to 0 for $Nc=1$ Set to 1 for $Nc=2$ ... Set to 7 for $Nc=8$	0: Nc is always 1 for pronto
Nr Index 3 bits	Indicates the number of rows in a matrix minus one: Set to 0 for $Nr=1$ Set to 1 for $Nr=2$ ... Set to 7 for $Nr=8$	2: for 3 rows in the matrix
Channel Width 2 bits	Indicates the width of the channel in which a measurement was made: Set to 0 for 20 MHz Set to 1 for 40 MHz Set to 2 for 80 MHz Set to 3 for 160 MHz or 80+80 MHz	2: for 80MHz
Grouping 2 bits	Number of carriers grouped into one: Set to 0 for $Ng = 1$ (No grouping) Set to 1 for $Ng = 2$ Set to 2 for $Ng = 4$ The value 3 is reserved	0: for no grouping
Codebook Information 1 bit	Indicates the size of codebook entries: If Feedback Type is set to 0 (SU-BF) Set to 0 for 2 bit for $\psi$ , 4 bits for $\Phi$ Set to 1 for 4 bit for $\psi$ , 6 bits for $\Phi$ If Feedback Type is set to 1 (MU-BF) Set to 0 for 5 bit for $\psi$ , 7 bits for $\Phi$ Set to 1 for 7 bit for $\psi$ , 9 bits for $\Phi$	1: for 4 bit for $\psi$ , 6 bits for $\Phi$
Feedback Type 1 bit	Set to 0 if the feedback report is for SU-BF. If it is set to 0, the feedback report frame shall not include the MU Exclusive Beamforming Report field (see 7.3.1.33) Set to 1 if the feedback report is for MU-BF. If it is set to 1, the feedback report frame shall include the MU Exclusive Beamforming Report field (see 7.3.1.62)	0: for SU
Remaining Feedback Segments 3 bits	Set to zero the Maximum VHT compressed beam forming report size is 1475 we don't fragment the VHT Compressed Beamforming frame.	0
First Feedback Segment 1 bit	Set to 1 as the only feedback segment of an unsegmented report	1
Sounding Sequence 6 bits	Sequence number from the NDPA soliciting feedback	Value from NPD Announcement

# VHT Compressed Beamforming Frame – Report Fields

- The VHT Compressed Beamforming Report field contains the feedback information and is used for carrying explicit feedback information in the form of angles representing Compressed Beamforming feedback matrices  $V$  for use by the transmit beamformer to determine the steering matrices  $Q$  (see 8.4.1.48 in 11ac standard for details)
- The MU Exclusive Beamforming Report field is included if the Feedback Type subfield in the VHT MIMO Control field is set to MU (see 8.4.1.49 in 11ac standard for details)

# Beamforming Report Poll Frame



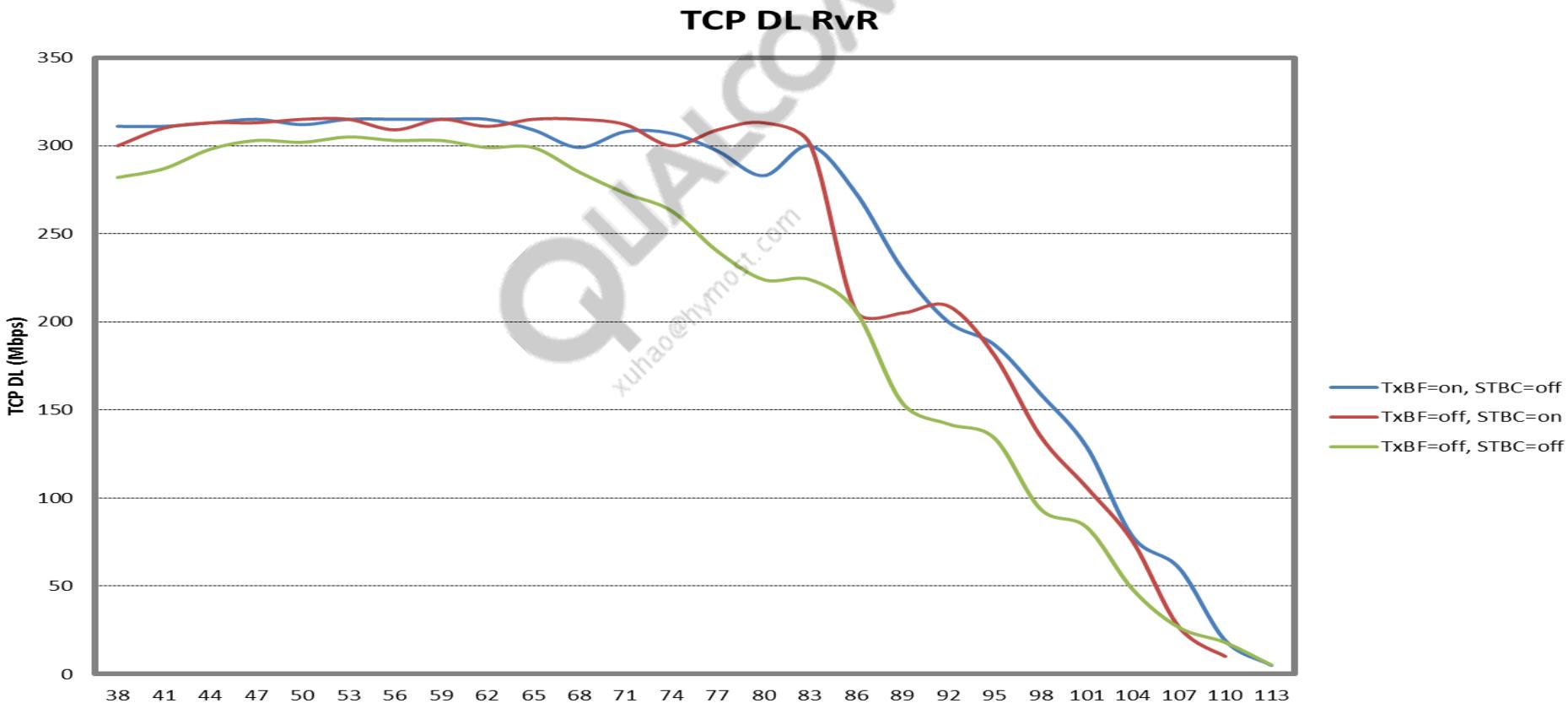
- BR poll is the control frame and it contains the fields as shown above
- RA field is the address of the intended recipient
- TA field is the address of the STA transmitting the Beamforming Report Poll
- The Feedback Segment Retransmission Bitmap field indicates the feedback segments to be polled in a VHT Compressed Beamforming Report, which is contained in one or more VHT Compressed Beamforming frames; the bit in position n is set to 1 when the segment with the Remaining Feedback Segments subfield in VHT MIMO Control field that is set to n is requested

# Transmit Beamformee – Code Change

- TxBF is added starting from driver 3.2.0.51
  - <https://www.codeaurora.org/gitweb/external/wlan/?p=prima.git;a=commit;h=08f87c23add121e1fd168aa2c79597214314cc5d>
  - The driver has to fill out correct VHT Capability Information in management frame and compare capability both on STA and AP and then inform firmware to enable beamformee or not
- gTxBFEnable is the configuration in WCNSS\_qcom\_cfg.ini to enable/disable the feature
- VHT sounding protocol is completely implemented in the hardware; the firmware needs to configure the hardware register according to the capability information from the driver

# TCP DL RvR for Rx STBC and Tx Beamformee

- TCP DL RvR curve from the 8974 FC release test report document (see [Q18])
- Rx STBC and Tx Beamformee offer link improvement at middle range



# Low Density Parity Check (LDPC)

- Low Density Parity Check (LDPC) technology provides improved performance gains compared to standard convolutional coding
  - LDPC's key attraction is that it uses a high-performance error correction mechanism that approaches the theoretical Shannon limit for efficiency
  - LDPC remains an optional feature for 11ac
- Pronto supports LDPC in both HT and VHT modes and for transmission only
  - LDPC is not supported in RIVA
- Rx LDPC capability is conveyed in the HT/VHT Capabilities Info field; all are set as 0 from WCN, but AP has to have these bits as 1 to support LDPC transmission from WCN
  - HT Capability Info Field for receiving LDPC coded packets – set as 0
  - VHT Capability Info Field for Rx LDPC – set as 0

LDPC Coding Capability	Indicates support for receiving LDPC coded packets	Set to 0 if not supported Set to 1 if supported
------------------------	--	--

- VHT Capability Info Field for Rx LDPC – set as 0

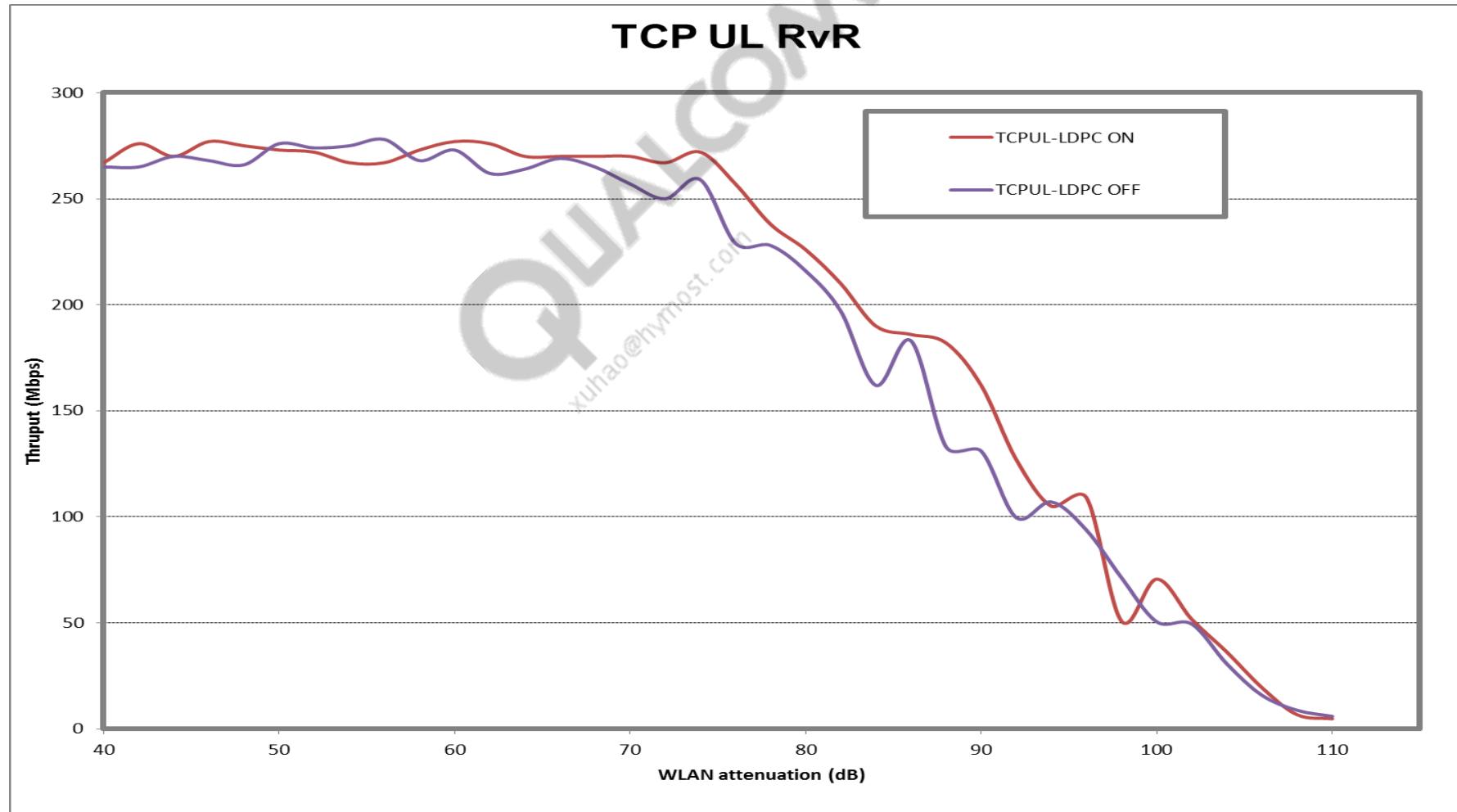
Rx LDPC	Indicates support for receiving LDPC encoded packets	Set to 0 if not supported. Set to 1 if supported.
---------	--	--

# Low Density Parity Check (LDPC) – Code Changes

- Host driver change to support LDPC starting from driver 3.2.0.45
  - <https://www.codeaurora.org/gitweb/external/wlan/?p=prima.git;a=commit;h=b2d2c31cd98ce1fd6ab4d001ed5a65f8a4601d55>
  - Driver has to compare the LDPC capability on STA and AP and inform firmware to enable LDPC or not
- gTxLdpcEnable is the configuration in WCNSS\_qcom\_cfg.ini
  - 0 - disable
  - 1 - HT LDPC enable
  - 2 - VHT LDPC enable
  - 3 - HT & VHT LDPC enable
- Pronto firmware would fill HW register to enable LDPC for HT and VHT rates; depends on configuration settings for gTxLdpcEnable
  - No new rates are introduced for LDPC

# Low Density Parity Check (LDPC) – TCP UL RvR

- TCP UL RvR curve from 8974 FC release test report document (see [[Q18](#)])
- LDPC offers link improvement at middle and higher rates



# 11ac Throughput Test – iperf Command

- iperf version
  - On DUT: iperf version 2.0.4 (7 Apr 2008)
  - On Windows 7 PC/Laptop: iperf version 2.0.2 (03 May 2005)
- iperf commands
  - TCPUL
    - ◆ PC Server: iperf -s -i 1 -w 2M
    - ◆ Phone Client: adb shell iperf -c <server IP address>-i 1 -t 30 -w 2M
  - TCPDL
    - ◆ Phone Server: adb shell iperf -s -i 1 -w 2M
    - ◆ PC Client: iperf -c <server IP address> -i 1 -t 30 -w 2M
  - UDPUL
    - ◆ PC Server: iperf -s -u -i 1 -w 512K
    - ◆ Phone Client: adb shell iperf -c <server IP address> -u -i 1 -t 30 -b 350M -w 512K
  - UDPDL
    - ◆ Phone Server: adb shell iperf -s -u -i 1 -w 512K
    - ◆ PC Client: iperf -c <server IP address> -u -i 1 -t 30 -b 350M -w 512K

# Google Jelly Bean (JB) WLAN Updates

REDEFINING MOBILITY

## ■ Wi-Fi Direct Service Discovery

- Wi-Fi Direct APIs are enhanced in Android 4.1 to support pre-association service discovery in WifiP2pManager
- This allows user to discover and filter nearby devices by services using Wi-Fi Direct before connecting to one, while Network Service Discovery allows you to discover a service on an existing connected network

## ■ Network Usage

- Network usage allows checking whether the device is currently connected to a metered network
- By checking this state before performing intensive network transactions, one can help manage the data usage that may cost your users money and make informed decisions about whether to perform the transactions now or later

## ■ Wi-Fi Direct Service Discovery

- ***addLocalService***
  - ◆ Registers a local service for service discovery
  - ◆ If a local service is registered, the framework automatically responds to a service discovery request from a peer
- ***setDnsSdResponseListeners***
  - ◆ Registers a callback to be invoked on receiving Bonjour service discovery response
- ***setUpnpServiceResponseListener***
  - ◆ Registers a callback to be invoked on receiving Upnp service discovery response
- ***discoverServices***
  - ◆ Initiates service discovery
  - ◆ A discovery process involves scanning for requested services for the purpose of establishing a connection to a peer that supports an available service

## ■ Network Usage

- ***isActiveNetworkMetered***

- ◆ Allows checking whether the device is currently connected to a metered network
- ◆ Returns whether the currently active data network is metered
- ◆ A network is classified as metered when the user is sensitive to heavy data usage on that connection
- ◆ Needs to be checked before doing large data transfers

# WLAN Changes in Google JB

Component	Description
Framework	<ul style="list-style-type: none"><li>Changes in SOFTAP interface name to wlan0</li><li>Changes in Wifi State Machine</li></ul>
Suplicant	<ul style="list-style-type: none"><li>Enablement of additional P2P flags</li><li>Addition of missing driver commands</li><li>P2P Stability Enhancements</li><li>Changes to support off channel capability</li></ul>
Qcom/msm8960	<ul style="list-style-type: none"><li>Board Config changes (Wifi driver module path, name and arguments)</li><li>Modified ctrl-interface in supplicant conf file.</li></ul>
Qcom/common	<ul style="list-style-type: none"><li>Addition of wpa_supplicant and p2p_supplicant services which will be used for standalone STA and Wifi Direct Modes</li><li>Addition of wifi-sdio-on service</li></ul>
Platform	<ul style="list-style-type: none"><li>Addition of wpa_supplicant status property to avoid ANR.</li><li>Corrected interface directory path for supplicant and framework to attach to the same socket path.</li><li>Changes in load/unload driver routines as per Qcom wifi solution</li><li>NDC support for SoftAP</li></ul>
Persist	<ul style="list-style-type: none"><li>Added support for generating extra images (like persist &amp; NAND generation)</li><li>Add back support for persist partition.</li></ul>
WLAN Driver	<ul style="list-style-type: none"><li>Changes to improve Wifi Direct Connection</li><li>Timer fixes</li><li>P2P: pin from peer device changes</li><li>P2P device interface changes</li></ul>

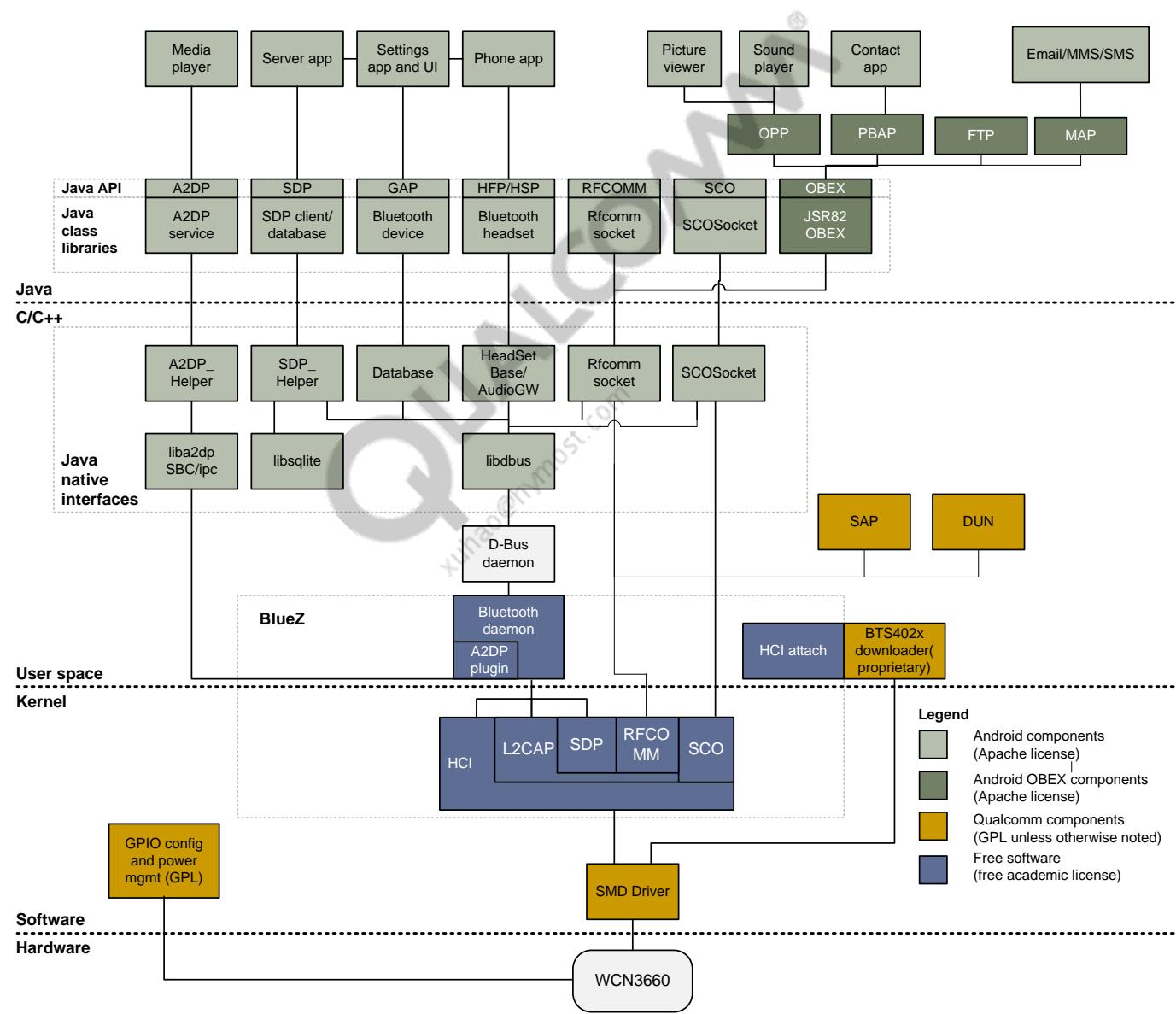
# WLAN Changes in Google JB (cont.)

Component	Description
WiFi Direct	<ul style="list-style-type: none"><li>Supports STA+P2P Client Single Channel Concurrency</li><li>Pin from peer device and Pin from this device options are not supported anymore</li><li>Supports WPS method as PBC by default; user cannot select any other WPS method while initiating a connection</li><li>NO option for Autonomous Group creation</li><li>There is NO separate option for Wifi Direct in Wireless Settings anymore</li><li>Wifi Direct would turn on automatically when Wifi is turned ON</li><li>To turn ON Wifi Display, Wifi needs to be turned ON first and then select Wifi Display option in Wireless Setting</li></ul>
Concurrency	<ul style="list-style-type: none"><li>Supported by default in JB releases</li></ul>

## Android Bluetooth Host – BlueZ

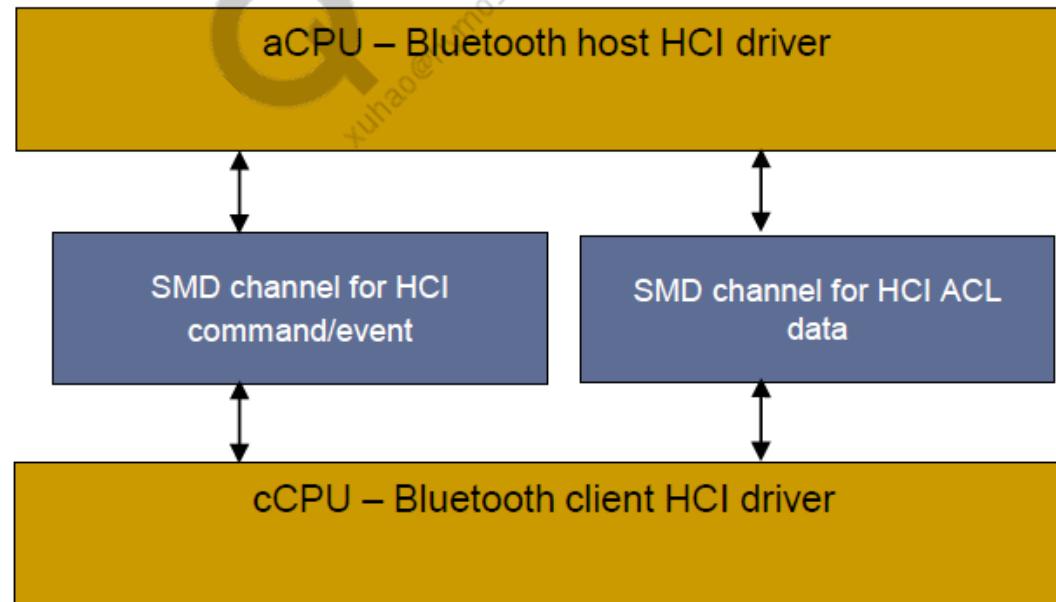
REDEFINING MOBILITY

# Bluetooth SW Architecture on Android (MSM8960/APQ8064/MSM8260A +WCN36x0)



# Software Block Diagram – HCI-SMD Interface

- SMD is a shared memory driver that is used for Inter-Process Communication (IPC) between the different CPU cores
- Between cCPU (WCNSS) and aCPU
  - Two SMD channels are used for Bluetooth HCI commands and events, as well as HCI ACL data transfers
  - One SMD channel for the WCNSS debug logging shared by WLAN, Bluetooth, and FM



# Software Block Diagram – BlueZ Kernel Space

- Socket-based interface – Use to interface between user applications (RFCOMM, Bluetooth daemon) and BlueZ kernel driver
- Host Controller Interface (HCI) – Use SMD hardware interface as transport layer between BlueZ host stack and BT SoC controller

# Software Block Diagram – BlueZ User Space

## ■ Concept

- Bluetooth daemon
  - ◆ Manage SDP database, security manager, authorization agent, HSP/HFP AG, A2DP, and ALSA plug-in
  - ◆ Single-user application
  - ◆ Expose DBUS interface to interface with Android user applications
- DBUS daemon
  - ◆ Interprocess communication (refer to <http://dbus.freedesktop.org>)
  - ◆ Interface between Android user applications and Bluetooth daemon
- Tools – Provide tools (hcitool, hciattach, hciconfig, hcidump, etc.) for debugging and Bluetooth configurations

# BlueZ Initialization

- Startup scripts
  - /etc/init.qcom.rc and /etc/init.qcom.bt.sh are scripts for Android Bluetooth startup procedure
- hci\_qcomm\_init application
  - Configures Bluetooth controller
  - Apply customized bd\_address, clock setting, power class setting through downloading NVM tags via HCI VS commands
- btnvtool application
  - Read/Write bd\_address, clock setting board configuration from BT-NV persist file
- hciattach application
  - Register BlueZ TTY line discipline driver and set SMD interfaces (event channel and data channel) between SMD and BlueZ kernel module

# BlueZ Startup Procedure

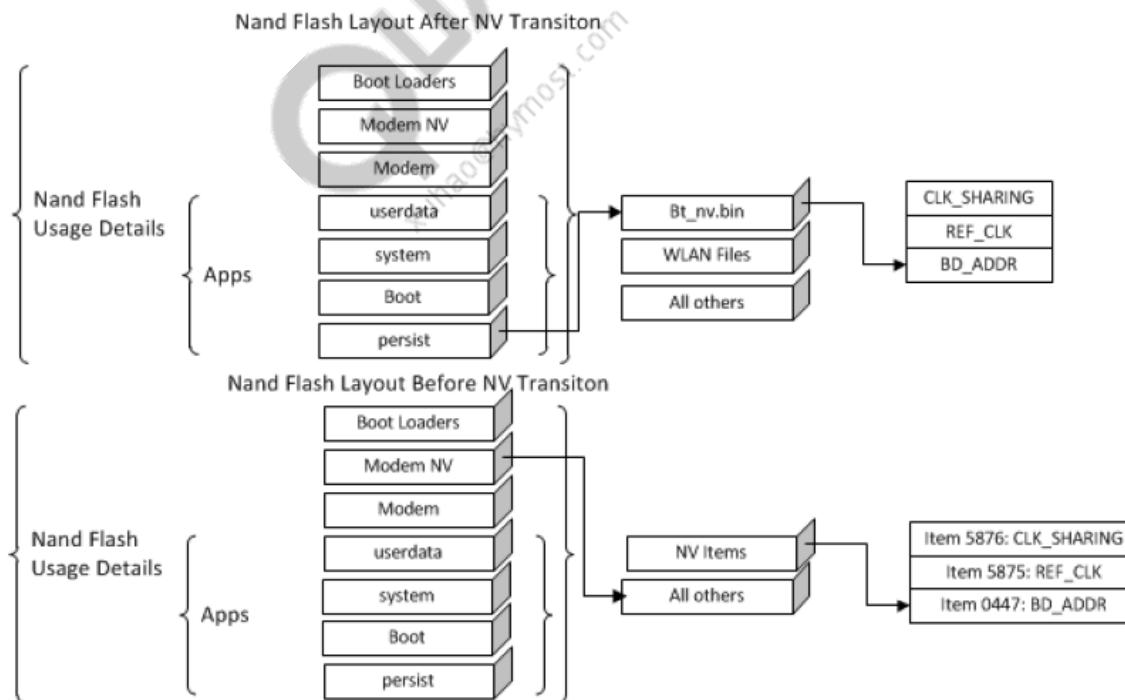
- Call flow on system boot-up (PowerOff→HotOff power state change)
  - Functions bt\_enable, bt\_disable in android/system/bluetooth/bluedroid/bluetooth.c are entry points to enable/disable Bluetooth
  - Related services are declared in android/system/core/rootdir/etc/init.qcom.rc
  - Modify arguments for hci\_qcomm\_init in android/system/core/rootdir/etc/init.qcom.bt.sh for your platform dependency
  - Bttest test tool enables/disables Bluetooth device
    - ◆ bttest enable and bttest disable commands in ADB shell
    - ◆ Reference source code – android/system/bluetooth/bluedroid/bttest.c

# Bluetooth SoC Configuration – hci\_qcomm\_init

- In new MSM8960/APQ8964/MSM8260A integrated WCNSS design, Bluetooth controller does not need to be initialized with NVM tags (already initialized with WCNSS firmware image); however, Qualcomm provides hci\_qcomm\_init tool to configure/customize Bluetooth controller
- Parameters are used to set up different hardware configurations or print debug messages, e.g., hci\_qcomm\_init -r19200000 -c -p0 for Bluetooth SoC, with 19.2 MHz clock source and clock sharing with power class 1

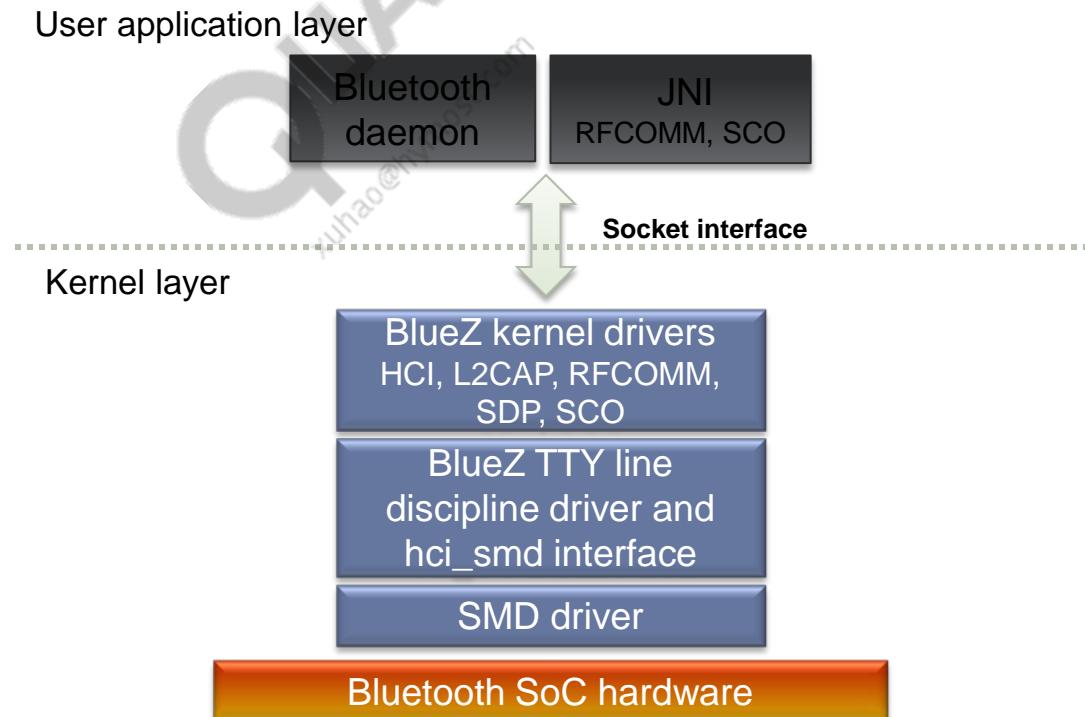
# Read/Write bt nv persist file— btnvtool

- Bd\_address is no longer stored in nv item #447 in current design; instead, bd\_address along with reference clock rate and clock sharing setting are stored in the bt\_nv.bin file under /persist folder
- btnvtool is the tool to read/write these board settings from bt\_nv.bin
  - Ex: Programming BD Address: >> btnvtool –b xx.xx.xx.xx.xx.xx ( where xx.xx.xx.xx.xx.xx is the bd address)

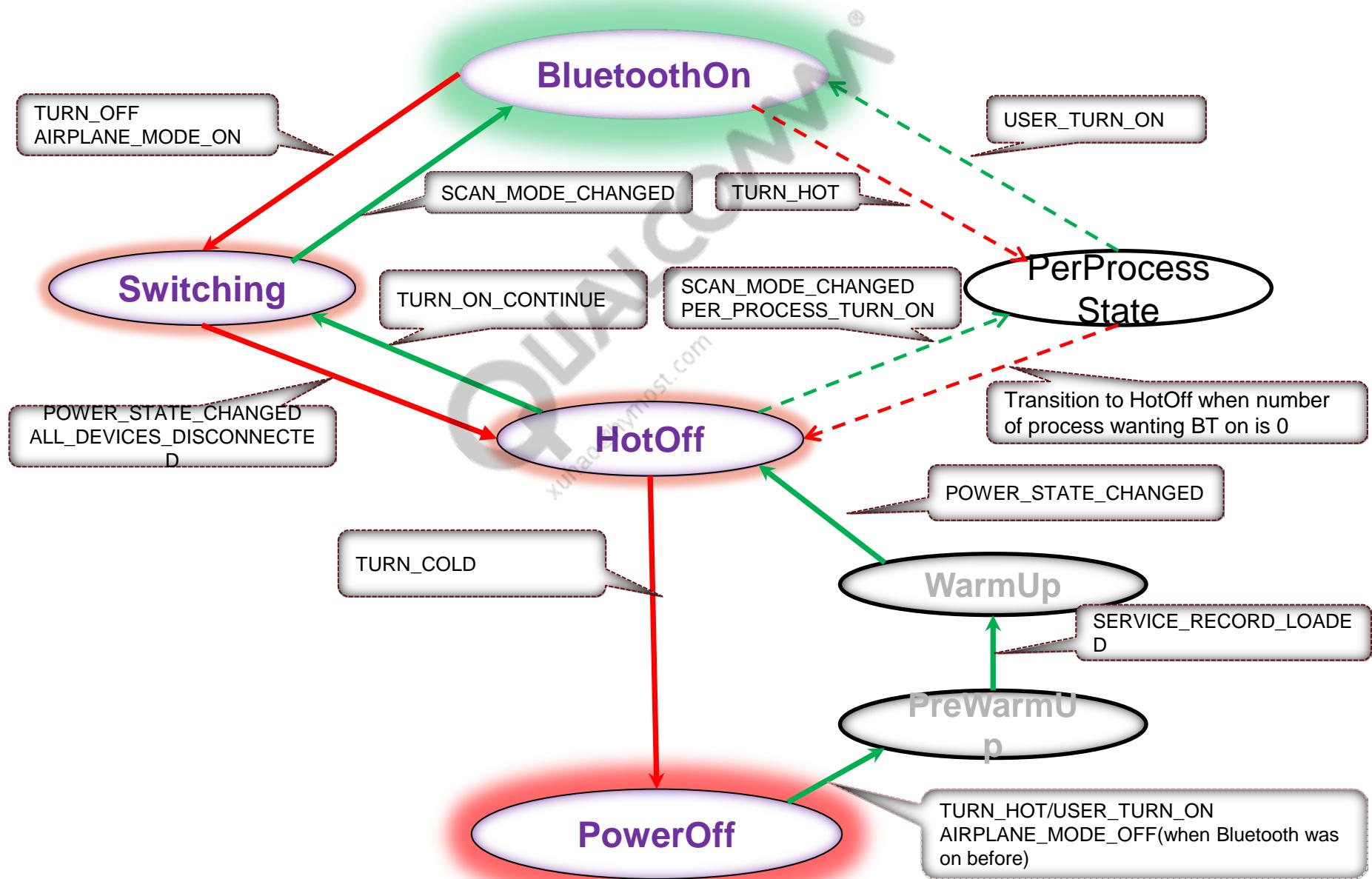


# BlueZ TTY Line Discipline – hciattach

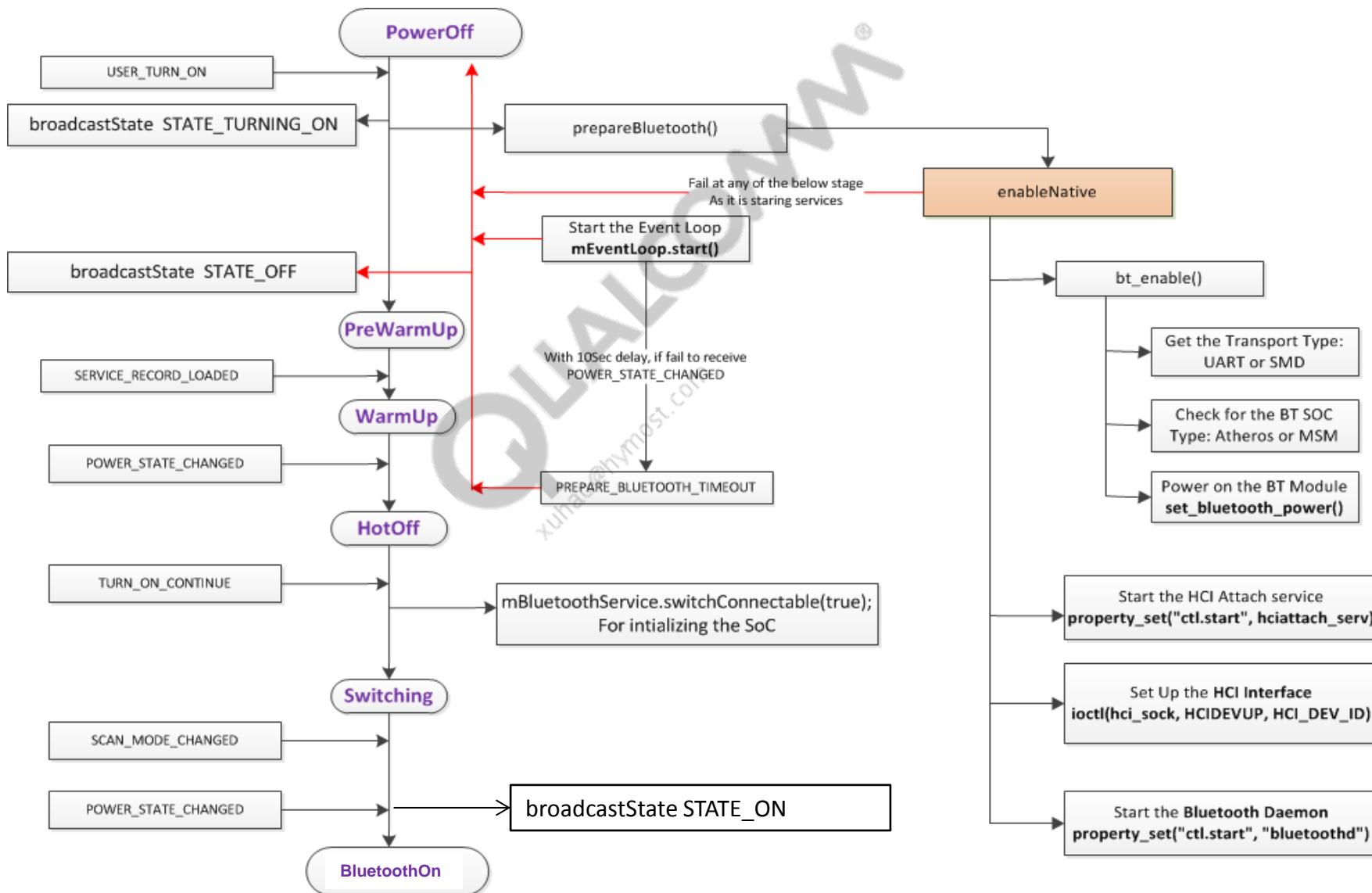
- hciattach – Register BlueZ TTY line discipline driver and set SMD interfaces (event channel and data channel) between SMD and BlueZ kernel module
  - Set up BlueZ TTY line discipline – android/kernel/drivers/bluetooth/hci\_ldisc.c
  - For SMD interface, hci\_smd is used- android/kernel/drivers/hci\_smd.c



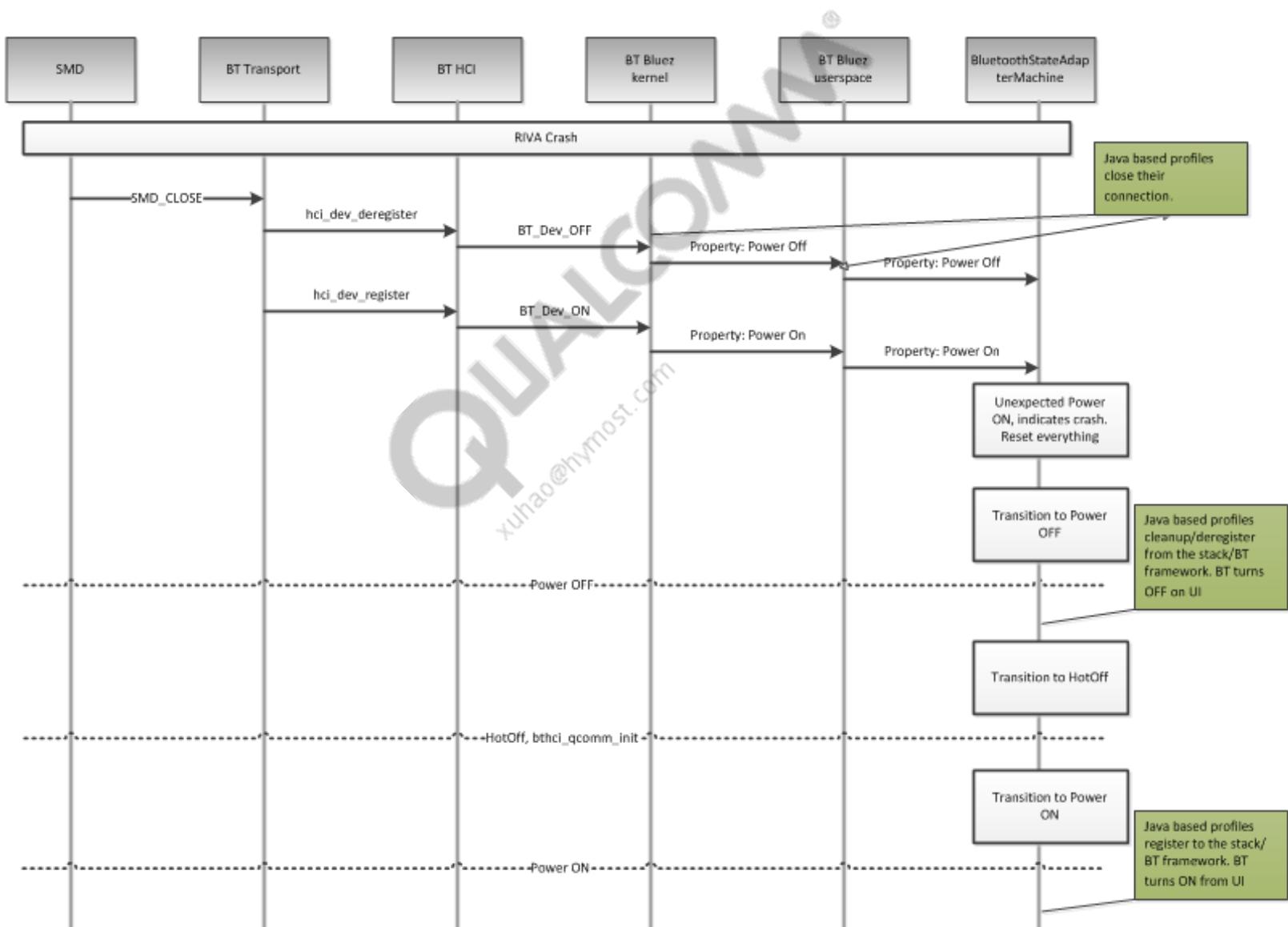
# Adapter State Machine(Complete Flow)



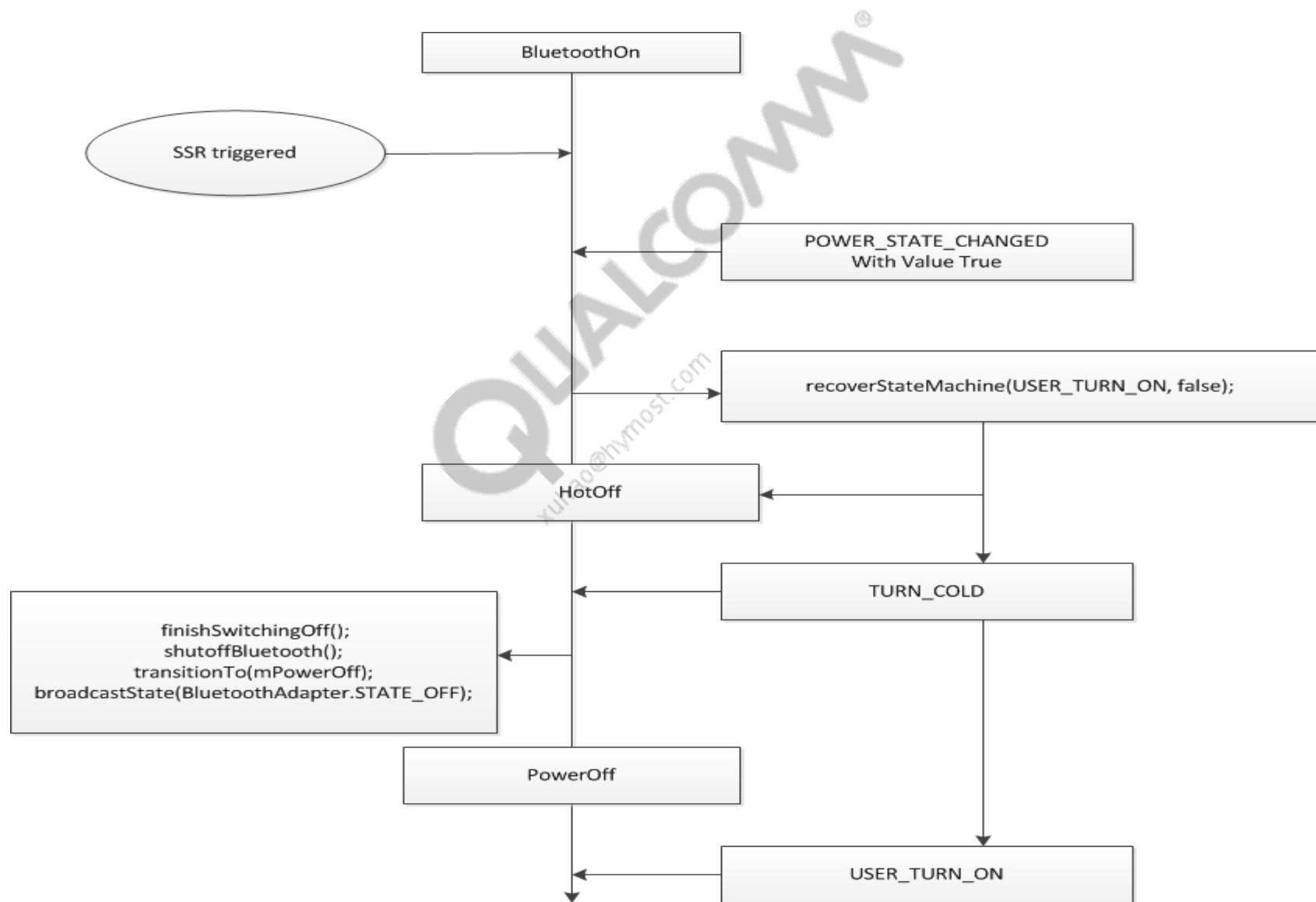
# AdapterStateMachine(USER\_TURN\_ON)- PowerOff → BluetoothOn



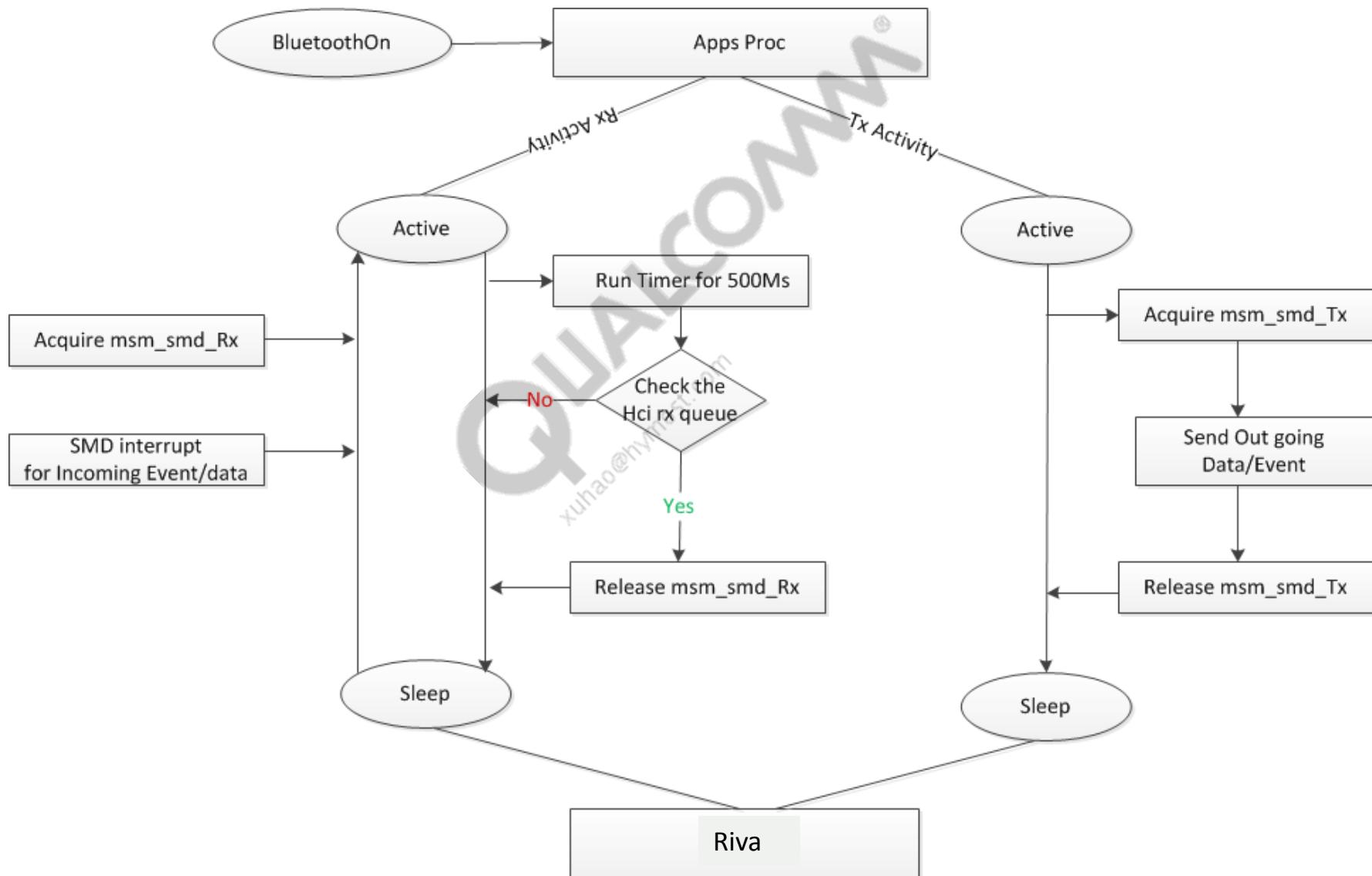
# SSR Sequence



## SSR Sequence (cont.)



# Host Sleep Implementation



# Bluetooth Source Directories (1/2)

- BlueZ kernel drivers
  - android/kernel/net/bluetooth
  - android/kernel/drivers/bluetooth/
- BlueZ user space
  - android/external/bluetooth/bluez
  - android/system/bluetooth/
- JNI APIs (to interface with DBUS and BlueZ Bluetooth daemon)
  - android/frameworks/base/core/jni
- Java APIs
  - android/frameworks/base/core/java/android/bluetooth
  - android/frameworks/base/core/java/android/server

## Bluetooth Source Directories (2/2)

- Bluetooth settings application
  - android/packages/apps/Settings/src/com/android/settings/bluetooth
- Bluetooth profile pack
  - android/packages/apps/Bluetooth/src/com/android/bluetooth
- NVM initialization
  - android/vendor/qcom/proprietary/bt/hci\_qcomm\_init
- BTC
  - android/vendor/qcomopensource/bt-wlan-coex
- MSM™ target related
  - android/kernel/arch/arm/mach-msm
  - android/device/qcom

# OBEX Throughput

- For OPP transfers, we have logs at APKs that can be used to profile the send/receive transfer rates
  - packages/apps/Bluetooth/src/com/android/bluetooth/opp/BluetoothOppObexClientSession.java
  - packages/apps/Bluetooth/src/com/android/bluetooth/opp/BluetoothOppObexServerSession.java
- Look for logcat logs
  - Receiving file completed for Rx rate
  - SendFile finished sending file for Tx rate

# RFCOMM Throughput

- Use the rctest tool to measure the throughput between two targets
  - Run rctest in server mode in target1
    - ◆ *adb shell rctest -r*
      - ▶ This would create a server on a particular <channel> and listen for incoming rfcomm connections
  - Run rctest in Client mode and transfer some data in target2
    - ◆ *adb shell rctest -s -i <BD\_ADDR of target 1> -P <channel> -B <file\_name>*
      - ▶ This would connect to target1's Rfcomm channel <channel> and transfer data packets from the file <file\_name>
  - Displays the Tx/Rx rate at the console
  - Ref: *adb shell rctest --help* for more details

# L2CAP Throughput

- Use the I2test tool to measure the throughput between two targets
  - Run I2test in server mode in target1
    - ◆ `adb shell I2test -r`
      - ▶ This would create a server on a particular <psm> and listen for incoming rfcomm connections
  - Run I2test in client mode and transfer some data in target2
    - ◆ `adb shell I2test -s -i <BD_ADDR of target 1> -P <psm> -B <file_name>`
      - ▶ This would connect to target1's L2CAP channel <psm> and transfer data packets from the file <file\_name>
  - It will display the TX/RX rate at the console
  - Ref: `adb shell I2test --help` for more details

# Certification

- Example QDL lists and PICS lists are at <https://www.bluetooth.org/tpg/listings.cfm>
- Latest Qualcomm Android BT solution QDID sets (as of 2012/10)
  - Controller subsystem (MSM8960, MSM8930, APQ8064, MSM8227, MSM8677 Bluetooth Component) – B018868
    - ◆ Will be updated for MSM8974 soon
  - Host subsystem (BlueZ host stack) – B019145
  - Profile subsystem (Qualcomm ICS Bluetooth Solution - BT.LA.10.2)- B018762 (can leverage to JB)

# Certification of Android Bluetooth System

Profiles	Licensees may leverage profile listing by Qualcomm (subject to Notes 1 and 2)
BlueZ stack (host subsystem)	Host listing from Qualcomm may be used, subject to Note 2. Linux kernel and BlueZ versions are referenced in the host listing. The listing to be leveraged must be determined based on the corresponding version information in the software release notes.
QCT controller subsystem	Qualification of baseband and link manager from controller subsystem can be leveraged by licensee; see the <a href="#">Bluetooth Controller Subsystem</a> slide for details

**Note:**

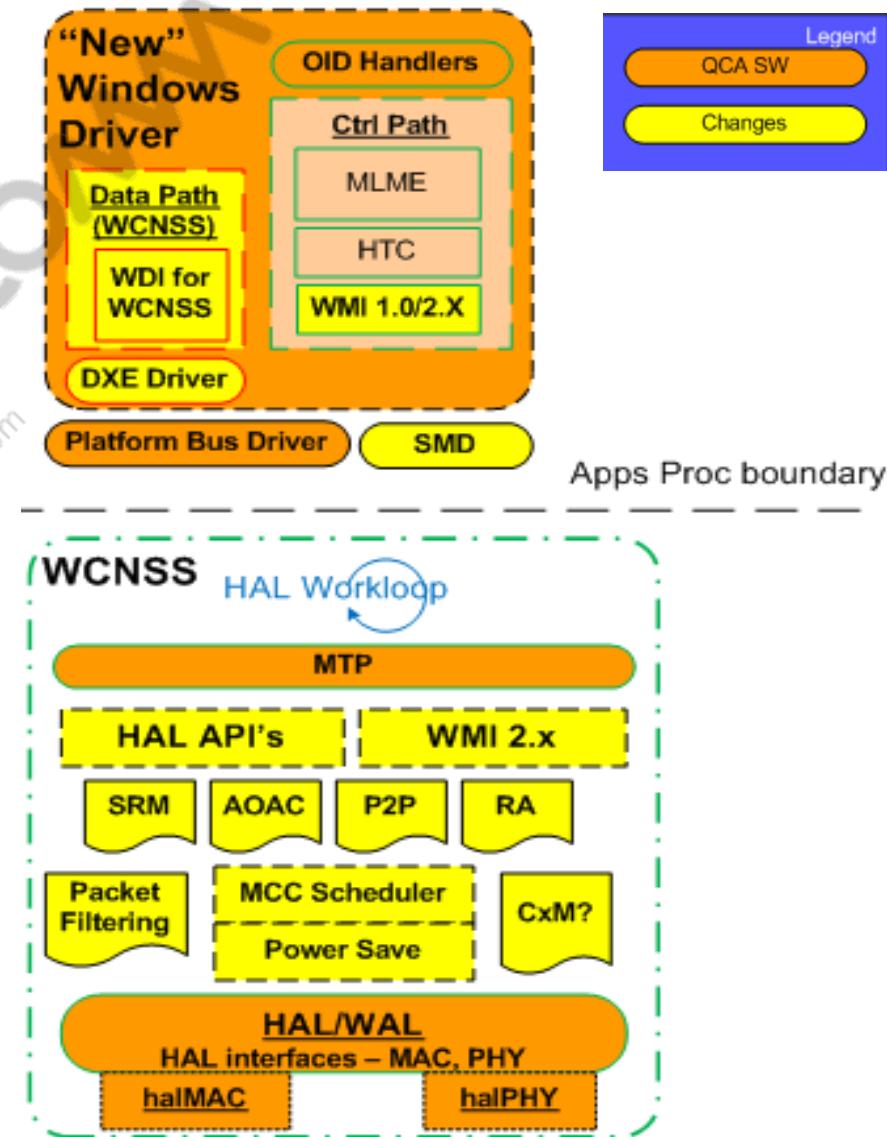
1. Licensees must recertify this component if the profile products are used in source form to compile their own objects/executables. If the licensee uses the object/executable as delivered, no additional qualification testing or listing is required.
2. Licensees may leverage the Qualcomm listings only if the Qualcomm Bluetooth controller is used.

# Windows WLAN Host Software Architecture

REDEFINING MOBILITY

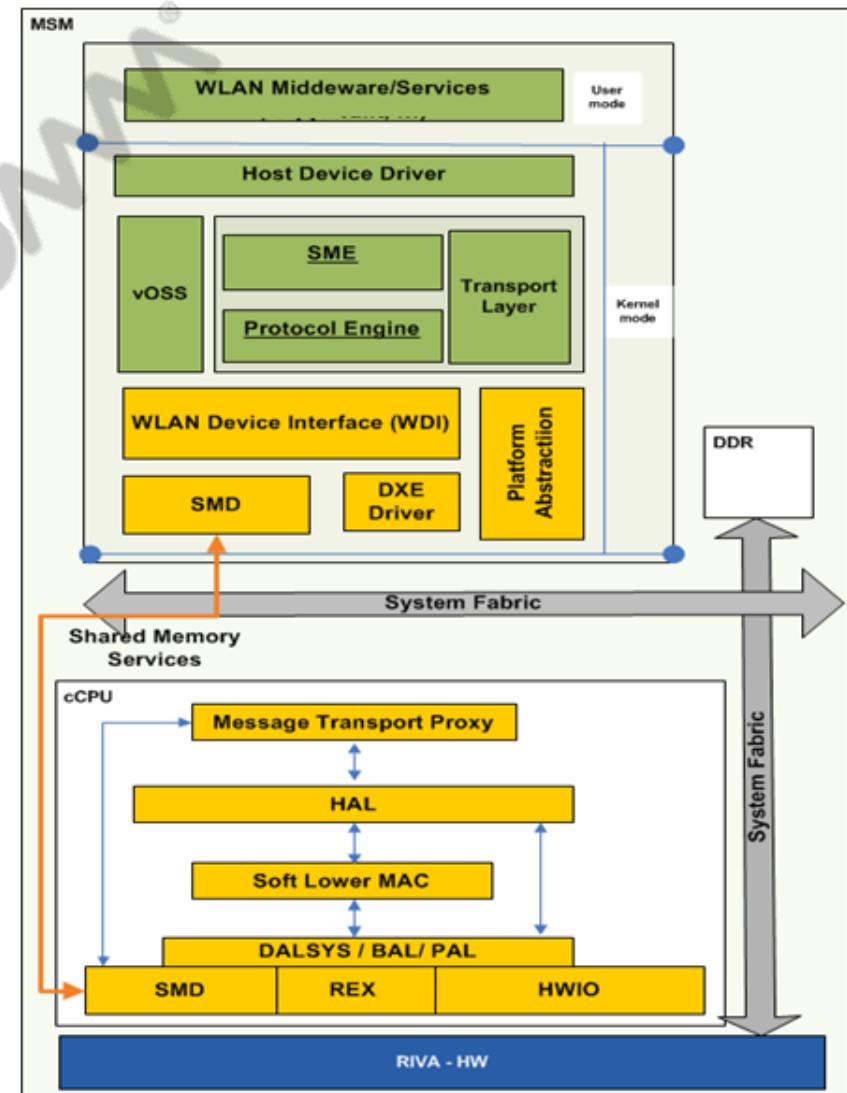
# MSM8974/8x26 Windows WLAN High-Level Architecture

- OID handlers – Shim layer that interfaces host software to the HLOS-specific network driver interface
- MLME (MAC Layer Management Entity) – Software layer responsible for 802.11 MLME functionality, e.g., IE processing, Mgmt frame handling, etc.
- HTC (Host Target Communication layer) – Provides virtual pipe/endpoint abstraction
- WMI (Wireless Messaging Interface) – Used for control path related communication between host and target
- WDI – Wireless Data Interface
- DXE Driver – Software that moves data between system memory and WLAN hardware using DMA
- Share Memory Driver (SMD) – Software that sets up a channel for the control message exchange between WLAN driver and cCPU
- Hardware Abstraction Layer (HAL) – Software layer that does all hardware control functionalities

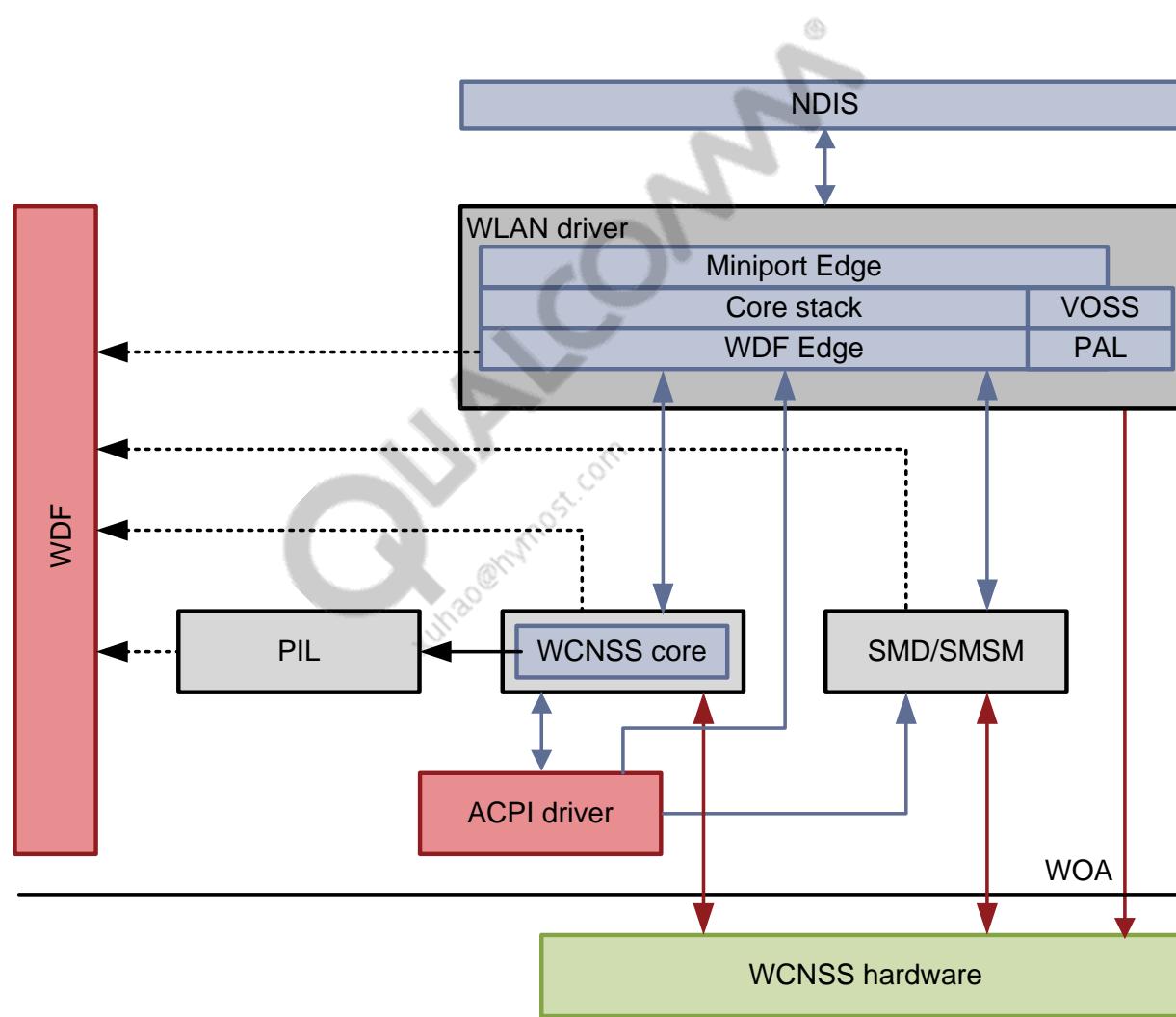


# MSM8960/8x30/AQP8064 Windows WLAN High-Level Architecture

- Host Device Driver (HDD) – Shim layer that interfaces Host software to the HLOS-specific network driver interface
- Station Management Entity (SME) – Software layer responsible for 802.11 SME functionality
- Protocol Engine (PE) – Software layer responsible for 802.11 MLME functionalities
- Transport Layer (TL) – Software that processes data frames that are received and to be transmitted
- DXE Driver – Software that moves data between system memory and WLAN hardware using DMA
- Share Memory Driver (SMD) – Software that sets up a channel for the control message exchange between WLAN driver and cCPU
- Hardware Abstraction Layer (HAL) – Software layer that does all hardware control functionalities
- Platform Abstraction Layer (PAL)
- Soft Lower MAC (SLM) – Software that implements low-level MAC functionalities, e.g., logic for power save, coexistence, RF cal, etc.



# Windows WLAN Windows Architecture Diagram



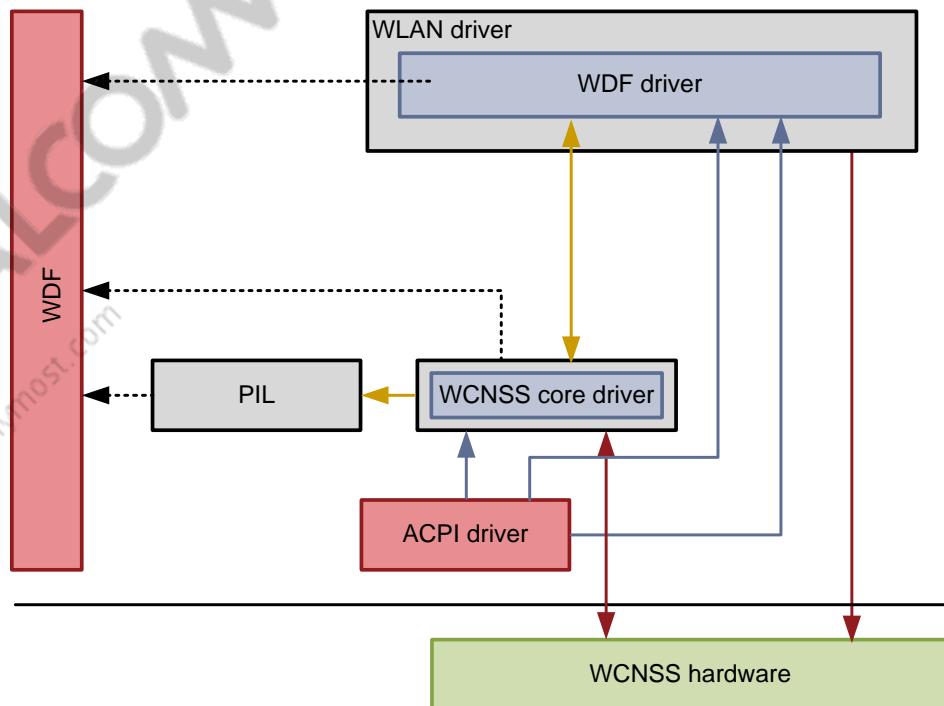
# Windows CoreBSP Boot Flow with UEFI on the MSM core

Components	Storage location	Exec processor	Exec location	Function
PBL	RPM BootRom	ARM7/RPM	RPM ROM	<ul style="list-style-type: none"> <li>▪ External storage device detection</li> <li>▪ Load and authenticate SBL1 (RPM SBL)</li> </ul>
SBL1	eMMC	ARM7/RPM	IMEM	<ul style="list-style-type: none"> <li>▪ Weak battery detection</li> <li>▪ Load and authenticate SBL2 (Krait PBL)</li> <li>▪ Bring Krait out of reset</li> </ul>
SBL2	eMMC	Krait	GMEM	<ul style="list-style-type: none"> <li>▪ Configure external DDR</li> <li>▪ Load and authenticate the TZ image and SBL3</li> <li>▪ Load and authenticate RPM firmware</li> </ul>
TZ Image	eMMC	Krait	IMEM	<ul style="list-style-type: none"> <li>▪ Set up the security environment (xPUs)</li> <li>▪ Provide Secured mode support on ROT</li> </ul>
RPM_FW	eMMC	ARM7/RPM	CodeRAM	<ul style="list-style-type: none"> <li>▪ Resource Power Manager*</li> </ul>
SBL3	eMMC	Krait	DDR	<ul style="list-style-type: none"> <li>▪ Reset detection</li> <li>▪ Shared Memory Initialization</li> <li>▪ Download Mode/Mass Storage mode</li> <li>▪ Load and authenticate UEFI boot image</li> </ul>
UEFI (SEC, DXE, BDS)	eMMC	Krait	DDR	<ul style="list-style-type: none"> <li>▪ Enable UEFI drivers and runtime services to meet the minimum Windows on Snapdragon boot requirements</li> </ul>
Win PE/APPS	eMMC	Krait	DDR	<ul style="list-style-type: none"> <li>▪ Windows on Snapdragon OS loader and kernel</li> </ul>

\*Resource Power Manager (RPM) – A subsystem within the MSM device that manages shared resources to optimize power consumption on the MSM device. This subsystem will facilitate WCNSS power consumption within the MSM device.

# Windows WCNSS Core Driver

- Loaded when the ACPI enumerates the WCNSS device defined in the ACPI table
- Driver of the WCNSS device enumerated by the ACPI driver
  - Consumes and handles the platform resources of the WCNSS device defined in the ACPI table
  - In particular, retrieves the WCN36x0 XO configuration as custom configuration data defined in the WCNSS name space expressed in the ACPI table
  - Interfaces with the PMIC/PEP driver to enable the appropriate voltage regulators required to power up WCN36x0 and WCNSS
  - Interfaces with the GPIO/TLMM driver to configure the WCNSS/IRIS GPIOs
  - Configures registers to enable the 48 MHz XO
  - Interfaces with the Peripheral Image Loader (PIL) to trigger download of the WCNSS image and take the cCPU out of reset



# Windows WCNSS Core Driver (cont.)

- Provides services to the WCNSS technology-specific driver
  - Provides an IOCTL that the technology-specific driver can query to determine if the WCNSS core driver has completed the loading of the WCNSS image and that WCNSS has successfully started
  - Services provided to WLAN driver
    - ◆ Own the DXE TX\_COMPLETE and RX\_AVAIL\_IND Hardware Interrupts and handle them on behalf of the WLAN driver
    - ◆ Notify the WLAN driver when each interrupt occurs by signaling an event to the WLAN Windows Driver Framework (WDF) driver when running the DPC handler associated with each of the interrupts

# Windows Device Enumeration

- Any device that requires a driver software stack needs to be expressed in DSDT
- A minimal device node entry is required in the DSDT to enumerate a WCNSS device.

Device( PRONTO )

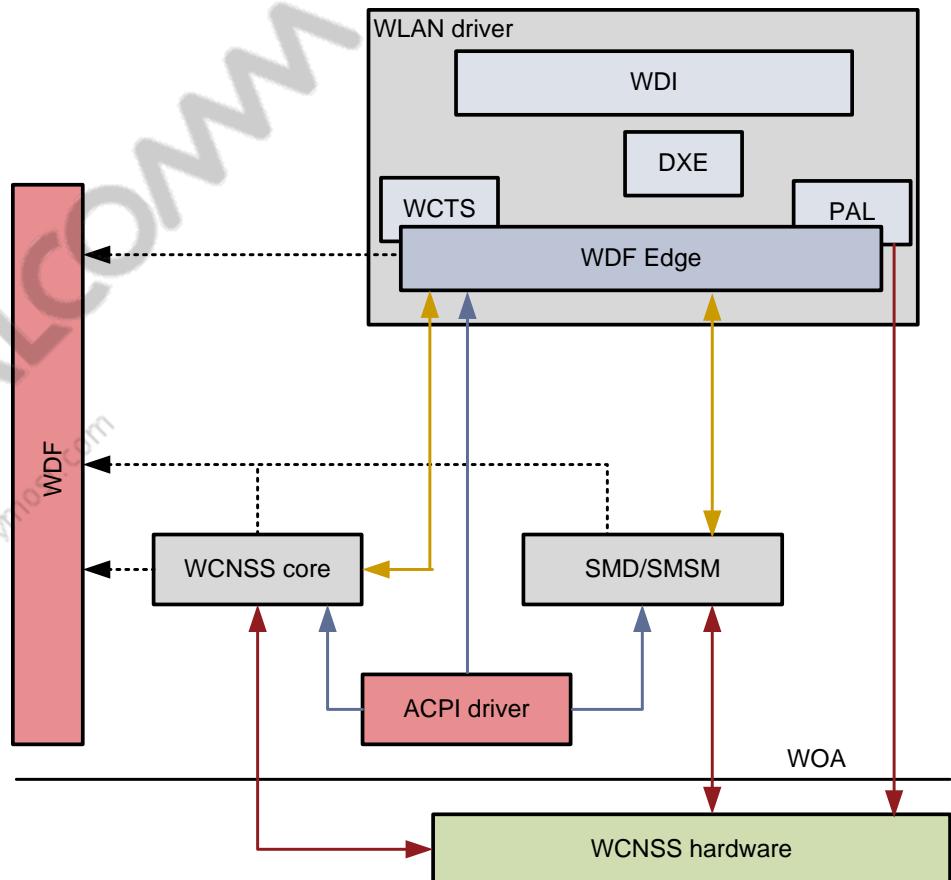
```
{  
    Name(_HID, "QCOM0E20") //Loads QCOM proprietary PRONTO driver  
}
```

- The driver .inf file would need to specify that the driver is ACPI-enumerated and the corresponding HID Value in the Models Section

“Qualcomm CORE WCN Driver”=WCN\_Device, ACPI\QCOM0E20

# Windows WLAN WDF Edge

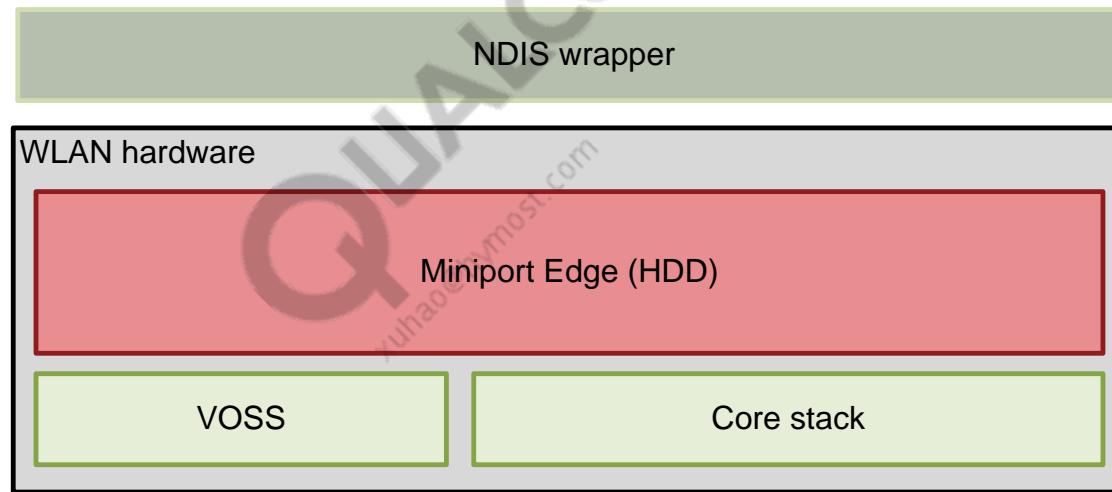
- Lower edge of the WLAN driver
- Advertises the WLAN driver as a WDF driver
  - Required to interface with all underlying drivers that are WDF drivers including the WCNSS core, SMD/ Shared Memory State Machine (SMSM)
  - Implements a worker thread that blocks on events signalization from WCNSS core and SMD drivers
    - ◆ Processes SMD Open and Read events from the SMD driver and serializes those events for processing in the WLAN Main Controller thread



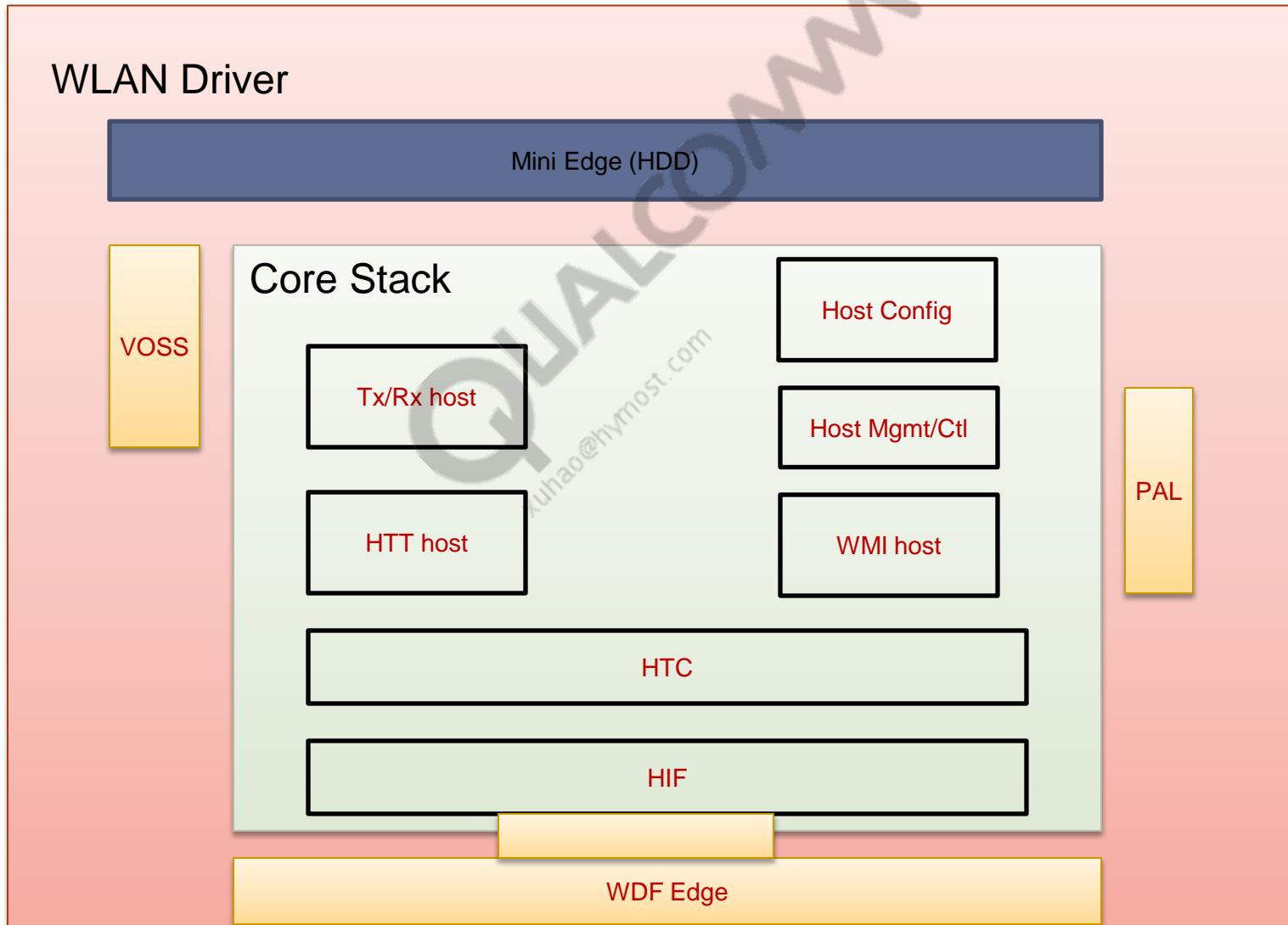
**Note:** WLAN Control Transport Service (WCTS).

# Miniport Edge

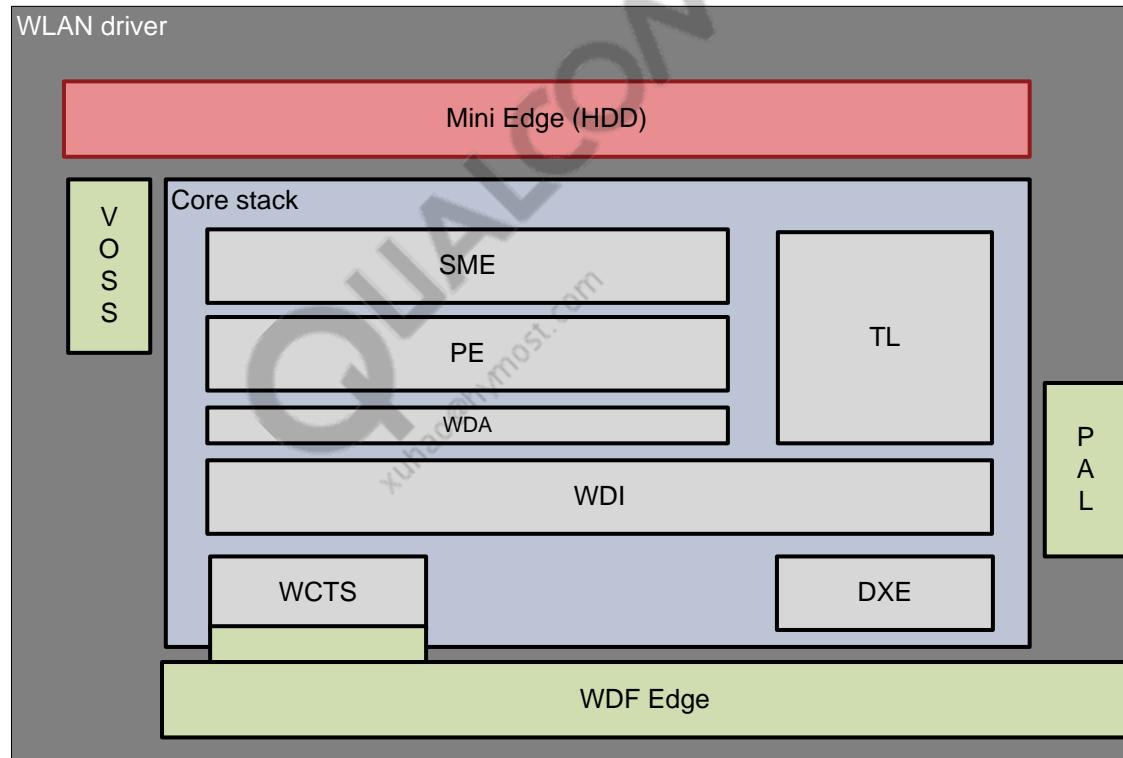
- HDD provides an interface to link Qualcomm WLAN core stack with Windows NDIS/Miniport interface core stack
  - This helps to reuse WLAN across different OSs



# MSM8974/8x26 WLAN Host Core Stack



# MSM8960/8x30/APQ8064 WLAN Core Stack



# Windows WLAN Driver Init Sequence

- Initialization is separated into two parts
  - The first part does not communicate with WCN36x0
  - The second part waits for ready notification from the WCNSS core driver before configuring WCN36x0

# Test Mode Support

- Factory Test Mode (FTM)
  - See [Q3] if customers want to develop their own application
- QXDM Professional – Diag interface support
- Microsoft WinDbg for collecting logs, stack trace, source code debugging
- Event Tracing for Windows (ETW)

# Certification

- Wi-Fi Alliance Certifications (Wi-Fi, WMM, WMM-PS, etc.)
  - Qualcomm will perform in-house precertification
- FCC
  - Qualcomm will perform in-house testing for our reference system to meet FCC requirements

# Windows WLAN Device Driver

- Provided as binary in prebuilt\wlan
  - qcwlans974cfg.dat
    - ◆ Default configurations and cannot be changed
  - qcwlans974.inf
    - ◆ WLAN driver installation
    - ◆ Configuration settings in registry to override default settings
  - qcwlans974.sys
    - ◆ WLAN driver
    - ◆ The device must be provisioned; otherwise, the WLAN driver may be unloaded by the OS; refer to [Q2] for the Provisioning

## Windows WLAN Power Save Feature

REDEFINING MOBILITY

# Windows WLAN Power Save Features

- System software and devices will manage power to provide the right balance of connection responsiveness and battery life
- Keep system power to a minimum while maintaining network presence and connectivity in low power system states
- A subset of Always-On/Always-Connected (AOAC) is supported
  - ARP and NS (Neighbor Solicitation) Offload (Protocol offloads)
    - ◆ Enable/Disable via Registries
    - ◆ Extract IP address value and MAC address through OIDs
    - ◆ Make a call to `sme_SetHostOffload` in case of D2 power state to send this info to Pronto
    - ◆ Send WMI command to get Pronto out of ARP offload
- Keep Alive and Data Inactivity – STA sends uplink keep-alive frames periodically to keep AP happy; the keep-alive frames are sent out only when there is no uplink traffic
- Beacon early Termination – Power down as soon as the TIM bit is seen as being clear without waiting for the entire beacon to be received
  - Enable/Disable and configure via Registries
- Nth Beacon Filtering (wake up to decode full beacon on every Nth beacon)
  - Enable/Disable and configure via registries
- Modulated DTIM
  - STA's Listen Interval (LI) will be a multiple of AP's DTIM period. ( $LI = \text{ModulateDTIM} * \text{AP DTIM}$ ) when DUT is in D2 power state. LI won't be back to 1.

# Windows System Power States

- D0 – Full Power state (See [S1])
  - BMPS can be enabled
- D2 – WoWLAN is enabled
  - Connected
    - ◆ Enter Connected Standby mode
      - ▶ Enter BMPS (WMI\_STA\_PS\_SET\_MODE)
      - ▶ If protocol offload enabled
        - Plumb protocol offload parameters
      - ▶ Entering WoW
        - Plumb WOWL pattern parameters (WMI\_STA\_PS\_SET\_PARAM)
        - Send WMI\_WOW\_ENABLE to Pronto
      - ▶ Exiting WoW
        - Send WMI\_WOW\_DISABLE to Pronto
    - Not connected
      - ◆ Enter Standby mode
        - ▶ Enter IMPS (WMI\_STA\_PS\_SET\_MODE)
  - D3 – WoWLAN is disabled (Pronto power collapsed)
    - ◆ Enter Standby mode
      - ▶ Enter IMPS (WMI\_STA\_PS\_SET\_MODE)

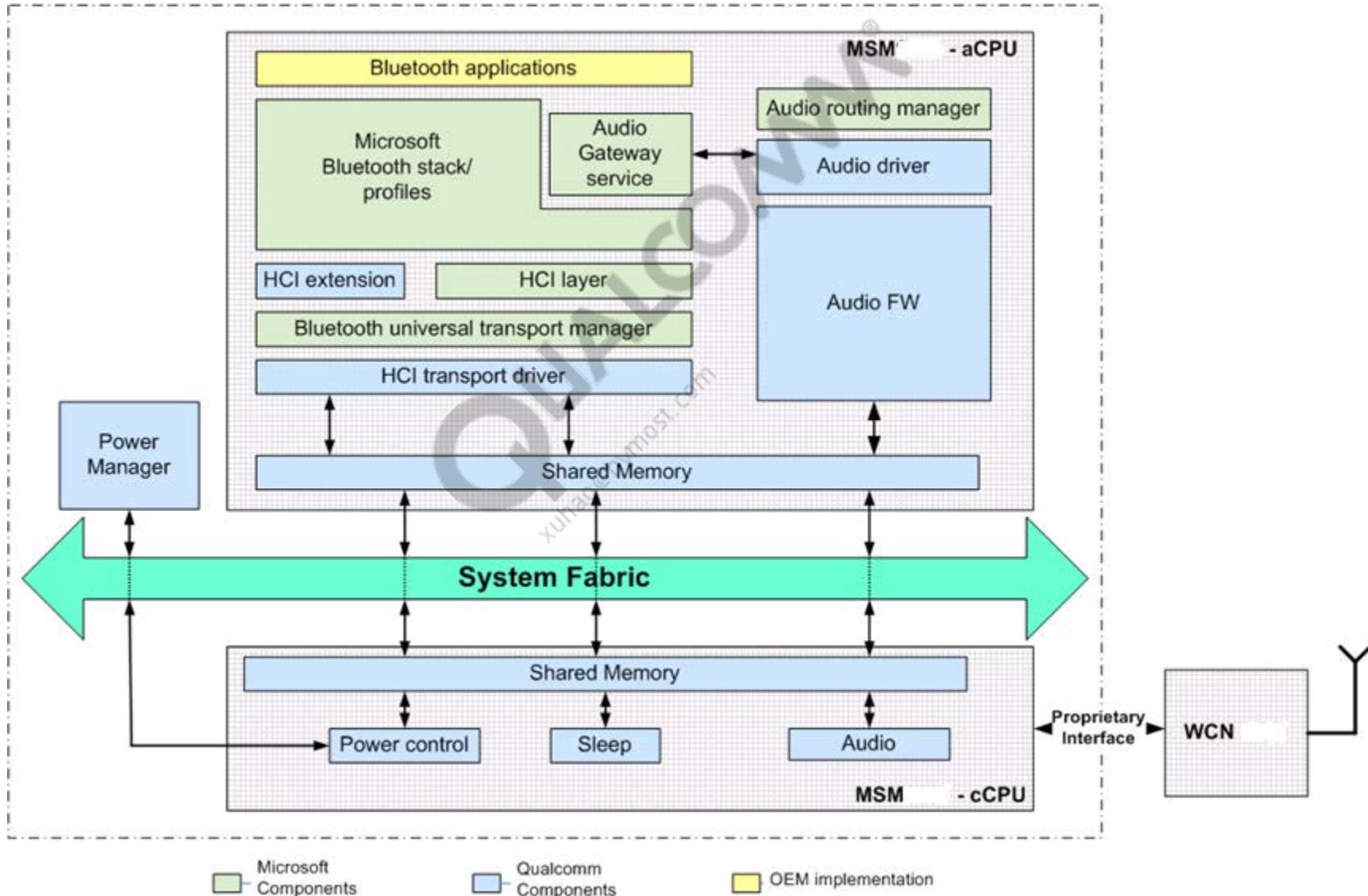
# Windows WLAN Power Save Modes

- Beacon Mode Power Save (BMPS)
  - Low Power state when AP is connected
  - Pronto power collapsed
  - Pronto wakes up at DTIM interval
  - Routes Rx data path to Pronto and performs offloads and packet filtering for WoWLAN
- Idle Mode Power Save (IMPS)
  - Low power state when AP is *not* connected
  - Pronto power collapsed
  - Pronto wakes up when host driver sends WMI\_WOW\_DISABLE

# Windows Bluetooth Host

REDEFINING MOBILITY

# Microsoft Bluetooth Software Architecture



# Microsoft Bluetooth Software Architecture (cont.)

- Shared memory – SMD driver
  - Memory space accessible from both the apps and connectivity processors
  - All communication between connectivity processor and external subsystems is through shared memory driver
- Power Manager
  - Power control and low Power Management
- HCI extensible transport driver, to port Qualcomm Bluetooth SoC into Bluetooth, has the following public APIs:
  - BTHX\_QUERY\_VERSION
  - BTHX\_QUERY\_CAPABILITIES
  - BTHX\_WRITE\_HCI\_COMMAND
  - BTHX\_WRITE\_HCI\_DATA
  - BTHX\_READ\_HCI\_EVENT
  - BTHX\_READ\_HCI\_DATA
  - BTHX\_SUSPEND\_NOTIFICATION

# Power Management

- SoC-based designs aimed for Always On, Always Connected
- Power Management – Granular power control
  - Bluetooth port supports the following Logical Power states:

Power state (Device state)	Condition
Active mode (D0)	Bluetooth controller is fully operational
Low Duty Cycle mode (D2)	Bluetooth controller is discoverable but has no associations; all active links are in Sniff mode
Off (D3)	Bluetooth controller is not discoverable and has no associations Bluetooth controller is not wakeable

- Device states (Dx) are one of the triggers for the bus driver to perform power control decisions
- Bus driver uses standard Wait-Wake mechanism to bring stack back into Active mode

# Profile and Logo Requirement

- Microsoft Bluetooth inbox core-stack
  - Bluetooth 3.0 (eL2CAP)
  - Bluetooth 4.0 LE
- Microsoft Bluetooth inbox profiles
  - HID 1.0, PAN 1.0, OPP 1.2, HCRP (Print)
  - HFP 1.5, A2DP 1.2, AVRCP 1.3
  - Additional profiles depends on Microsoft
- Microsoft logo requirements
  - Must use Microsoft's inbox Bluetooth

# HCI Driver Source Codes

- Bluetooth HCI SMD driver
  - btsmd.c
  - btsmd.h
- Bluetooth HCI port driver
  - driver.c
  - fdo.c
  - io.c

QUALCOMM®  
xuhao@hymost.com

# Debug Tools

- Bluetooth HCI-SMD transport driver
  - WinDbg – Microsoft Windows kernel debugger
    - ◆ Operating system
    - ◆ Drivers
    - ◆ Executables
  - TraceView
  - Logman/unified tracing
- WCN-SS Bluetooth firmware
  - Collect the firmware logs through QXDM Professional® F3 messages (known messages – Bluetooth)
    - ◆ LMP messages
    - ◆ Power status (sleep/wakeup)
    - ◆ HCI sniff packets
  - Some are the readable string format, but most of them are HEX format for the internal debug information; QXDM script will be provided to parse the logs

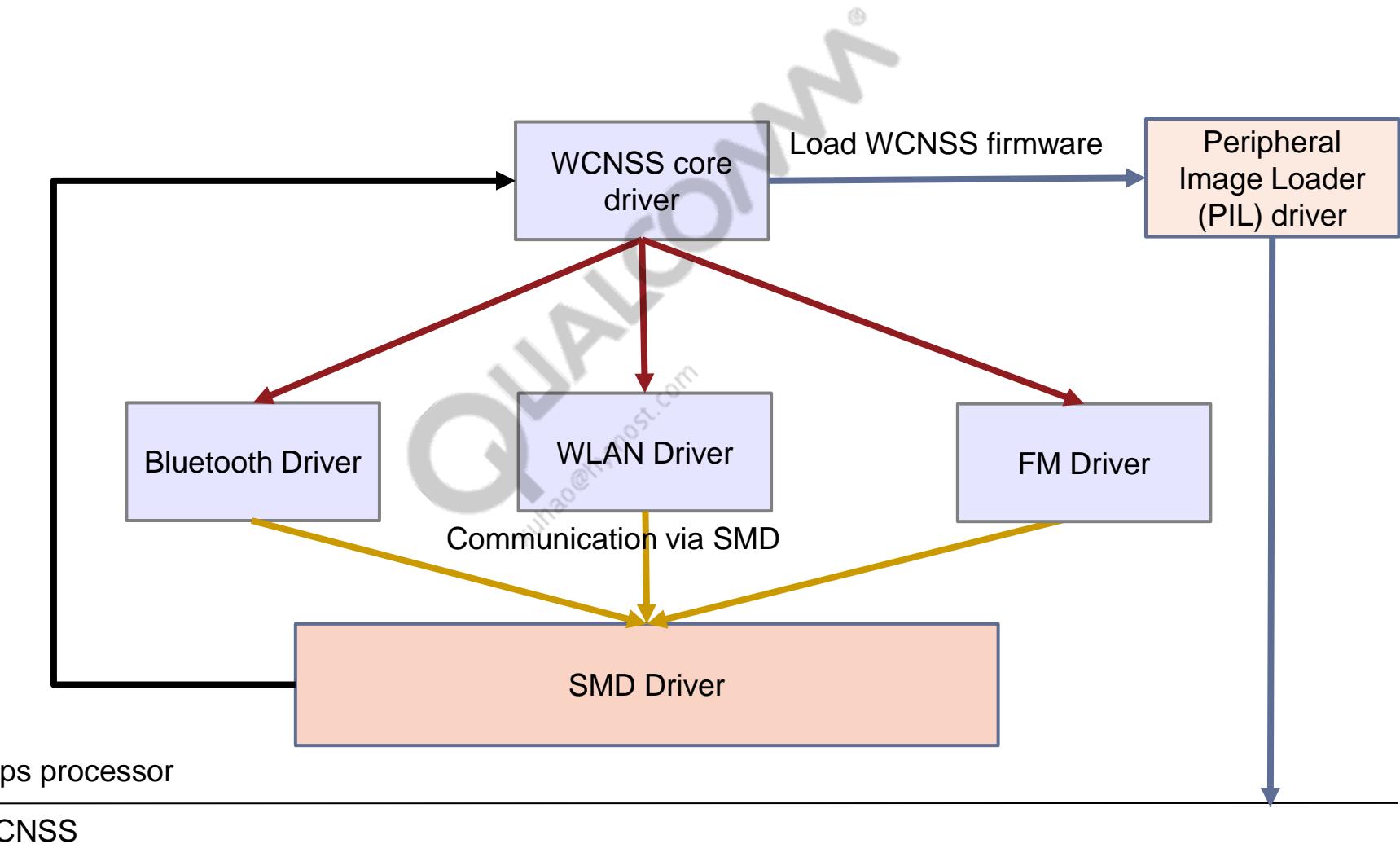
# Host Logs

- Host logging can be divided into two parts
  - Log messages for the Qualcomm Bluetooth transport driver
  - Log message for the Microsoft Bluetooth stack
- TraceView for HCI transport driver logs
  - TraceView can be run on the device to capture real-time transport driver logs
  - Requires the Bluetooth HCI driver PDB (debug symbol) file
- Unified tracing for Microsoft Bluetooth stack logs
  - Logman tool available to capture ETL traces; tool has the option to capture Bluetooth stack logs
  - Log messages can be post-processed by using a parsing tool
  - Using the correct filters, Bluetooth messages can be extracted

# Driver Dependencies

- PEP and PMIC drivers – These two drivers configure the voltage regulators and configure GPIOs needed for WCNSS
- SMD and SMEM driver – These two drivers are needed for communication between the host and controller
- PIL driver – Helps download the binary image needed for WCNSS
- Core Riva driver – Performs various hardware configurations needed in WCNSS
  - Configures the 48 MHz crystal
  - Configures the PLLs
  - Uses the above PIL driver to load WCNSS binary image on RAM
  - Brings WCNSS out of reset

# WCNSS Software Initialization



# WCNSS Power-Up Sequence

Step	Description	Owner
1	Power up WCN3660™ (PMIC and GPIO)	UEFI or PEP (PMIC)
2.0	Enable 48 MHz X0 if populated on board	Bringup – WCNSS Platform driver
2.1	Enable XO_OUT_A2 buffer if using MSM™ 19.2 MHz XO	Bringup – WCNSS Platform driver
3	Enable WCNSS PLL	Bringup – WCNSS Platform driver
4	Configure cCPU CLK to 240 MHz	Bringup – WCNSS Platform driver
5	Execute PIL cold boot sequence of cCPU	Bringup – WCNSS Platform driver

# Install Driver with Scripts and DISM – WinRT

- What is DISM?
  - Deployment Image Servicing and Management
  - Currently the only way to install EA drivers
- Requirement for installation
  - SD card
  - Dism.exe
  - Image install.wim
  - Scripts folder which contains installwoa.cmd and unattend-arm.xml
  - Prebuilt folder contains all the drivers
  - fre/winpe folder which shall contain boot, efi, and source directories

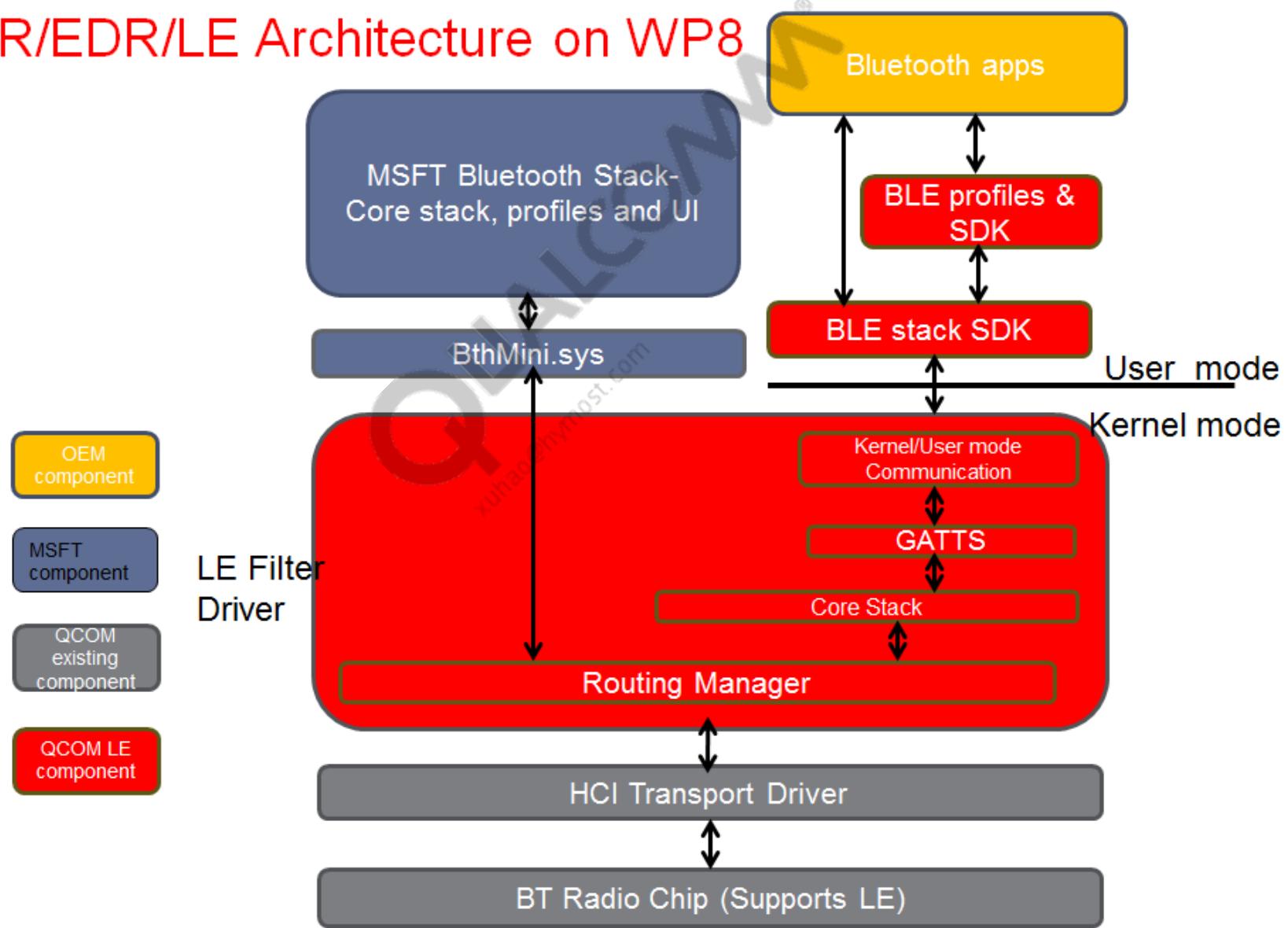
# Install Driver with Scripts and DISM – WinRT (cont.)

- To run scripts:
  - Copy all prebuilt drivers from CRM build to SD card, named prebuilt/
  - Copy Scripts folder to SD card
  - Copy dism.exe to Scripts folder
  - Copy install.wim from fre/client folder to SD card
  - Copy fre/winpe folder to SD card
  - Rename SD card as “BOOTME”
  - Plugin SD card in target and cd to Scripts folder.
  - Run the following command:
    - ◆ `installwoa.cmd -IMG:<path to install.wim> -DRV:<path to prebuilt/> -MN:WOATARGET`
  - The driver installation will be initiated
  - **Note:** Refer to [Q4] for exact location of any of above files and directories

# Install Drivers – WP8

- How to install drivers on Apollo
  - Currently, the only way to install drivers on Apollo is to rebuild the image with your new driver code. Then flash the image into your target.
- How to flash image
  - Requirement for flashing
    - ◆ ffu – Stands for full flash update which is the image going to be flashed
    - ◆ ffutool.exe
  - Get ffutool.exe by running  
  \Windows Kits\8.0\WP8\Docs\simpleio\installsimpleio.bat
  - Flash by running:  
  ffutool -flash <location of WP8.ffu in the build>
  - **Note:**  
  If flashing gets complained about target platform mismatched, run:  
  ffutool -flash <location of WP8.ffu in the build> **-force**

## BR/EDR/LE Architecture on WP8



# WP8 BLE Modules

## ■ BLE core stack.

- BLE core stack is implemented in a driver *QcbluetoothLE8960.sys*.
- By default, this driver will be installed automatically when MSFT classical Bluetooth driver is loaded.
- You can disable this driver, when the below registry key is set to 0  
`\controlSet001\services\QcbluetoothLE\Parameters\FilterEnable`
- This stack includes implementation of BLE modules such as GAP, ATT, GATT and SMP, as required.

## ■ BLE core stack SDK

- Qualcomm will provide BLE stack SDK to OEMS . They can develop their own profiles and applications by calling APIs which are provided in SDK.
- BLE core stack SDK is implemented in *qblestack8960.dll*

## ■ BLE profile SDK

- In order to reduce OEMS' development effort, Qualcomm also will release BLE profile SDK for specific profiles. So far, only FMP is supported.
- BLE profile SDK stands on BLE core stack. It is implemented in *qbleprofile8960.dll*

## ■ BLE core stack test tool

- Qualcomm developed a tool to test the functionality of BLE core stack. The tool will load *qcblestack8960.dll* and call those APIs which are provided in SDK.
- The tool is implemented in *qcblestacktest8960.exe*

## ■ BLE profile test tool

- Qualcomm developed a tool to test the functionality of BLE profile . The tool will load *qcbleprofile8960.dll* and call those APIs which are provided in SDK.
- The tool is implemented in *qcbleprofiletest8960.exe*

# How to test BLE

- BLE core stack test tool setup.
  - Make the device enter the mass storage mode.
  - Copy the LE test tool binary (`qcblestacktest8960.exe` and `qcblestack8960.dll`) to *TEST folder* which located in phone partition with *Windows* folder.
  - Telnet to the DUT and run `qcblestacktest8960.exe`.
  - Execute any commands which test tool supported, and check Events.
  - Check the more detailed commands by typing “/help” or “/?”

## BT/WLAN Coexistence

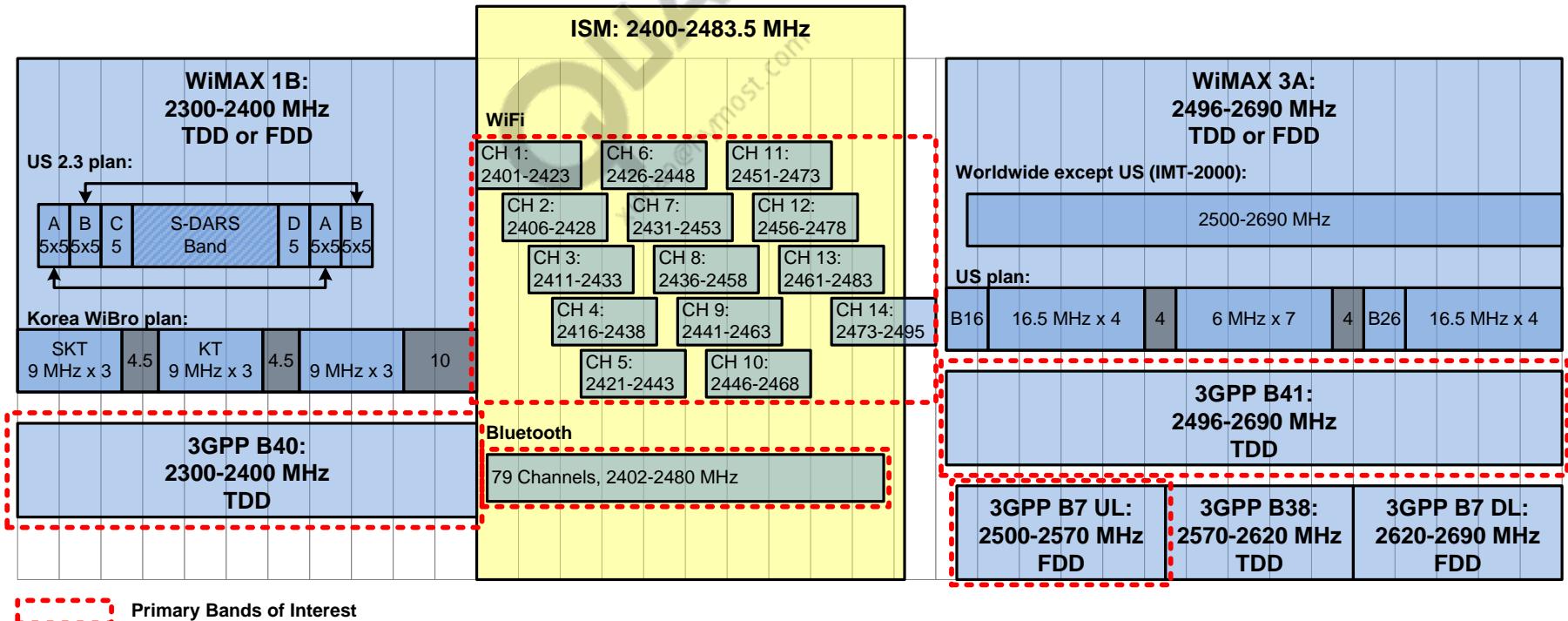
REDEFINING MOBILITY

# Introduction

## ■ Why coexistence?

- Wireless Local Area Network (WLAN) and Bluetooth share the same ISM band
- LTE shares adjacent bands (band 40 for TDD, band 7 for FDD uplink) and can have severe interference due to adjacent bands and limited isolation

## ■ Spectrum environment from 2.3 to 2.7 GHz



## Introduction (cont.)

- LTE-ISM concurrent use cases
  - LTE + Bluetooth

User scenario	Description
Voice	<ul style="list-style-type: none"><li>▪ Case 1a – VoIP over LTE terminated on a Bluetooth headset or carkit</li><li>▪ Case 1b – Voice control</li></ul>
Streaming music	<ul style="list-style-type: none"><li>▪ Case 2a – Audio source over the Internet, e.g., watching YouTube or listening to Internet radio</li><li>▪ Case 2b – Listening to music (local on phone) while doing anything over LTE</li></ul>
Bulk file transfer/synchronization	Case 3a – Transferring or synchronizing files while doing anything over LTE
Internet tethering	Case 4a – Tethering the LTE Internet connection to a laptop via Bluetooth Dial-Up Networking (DUN) or PAN

## Introduction (cont.)

- LTE-ISM concurrent use cases (cont.)

- LTE + WLAN

User scenario	Description
Infrastructure mode	<ul style="list-style-type: none"><li>▪ Case 1a – Connection to an access point with access to a local area network as well as externally to the Internet</li><li>▪ Case 1b – Connection to an access point with access to a local area network (but not external Internet access)</li></ul>
Ad-hoc file transfer	<ul style="list-style-type: none"><li>▪ Case 2a – File transfer or synchronization with another WLAN device in Ad-hoc mode with simultaneous LTE operation; WLAN data source or sink is not LTE</li><li>▪ Case 2b – File transfer or synchronization with another WLAN device in Ad-hoc mode with simultaneous LTE operation; data source or sink is LTE</li></ul>
Ad-hoc video streaming	<ul style="list-style-type: none"><li>▪ Case 3a – Video streaming over WLAN in Ad-hoc mode with simultaneous LTE; video source is not over LTE</li><li>▪ Case 3b – Video streaming over WLAN in Ad-hoc mode with simultaneous LTE; video source is LTE</li></ul>
Internet tethering	Case 4 – Tethering of the LTE data connection to another device using WLAN

# Introduction (cont.)

## ■ RF analysis summary

Band	Impact
7	<ul style="list-style-type: none"><li>▪ When LTE is using the closest channel to the ISM band, Bluetooth/WLAN can be desensed across the ISM band</li><li>▪ Bluetooth will not impact LTE</li><li>▪ WLAN may impact LTE on some channels, but this is highly dependent on the actual filters</li></ul>
40	<ul style="list-style-type: none"><li>▪ When LTE is using the top 30 MHz, Bluetooth and WLAN will be desensed across the ISM band</li><li>▪ When LTE is using the lower 70 MHz, the lower 20 MHz of the ISM band will have some desense</li><li>▪ When Bluetooth or WLAN is using the lower 20 MHz of the ISM band, it will desense LTE across the band</li><li>▪ When Bluetooth or WLAN is above 2420 MHz, only the upper 30 MHz of B40 are desensed</li></ul>
38	Can be solved with filtering
41	TBD

# Introduction (cont.)

- Within ISM bands
  - Bluetooth and 802.11 b/g/n radios occupy the same ISM 2.4 GHz frequency band
    - ◆ Simultaneous operation will cause interference
    - ◆ Coexistence techniques are required to minimize mutual interference
  - There are 14 WLAN channels, each 22 MHz wide
    - ◆ Japan uses all 14 channels
    - ◆ Europe uses 13 channels
    - ◆ U.S. uses 11 channels – Channels 1, 6, and 11 are used to minimize overlap
  - There are Bluetooth channels 2402 to 2480 MHz, 1 MHz step
  - WLAN and Bluetooth radios may directly interfere with each other if they both attempt to transmit near the same frequency at the same time
    - ◆ Can support simultaneous transmission where frequencies are far apart
- WCN36x0 Bluetooth and WLAN share the same antenna/LNA
  - Only one technology can physically transmit at a time
- Coexistence Manager (CxM)
  - Newly introduced from MSM8960 chipset
  - To address the problem of multiple technologies collocated on a single device
    - ◆ LTE, WLAN, and Bluetooth
  - Refer to “Android software versions and roadmap” page for future support schedule

# Solutions to LTE/Bluetooth/WLAN Coexistence

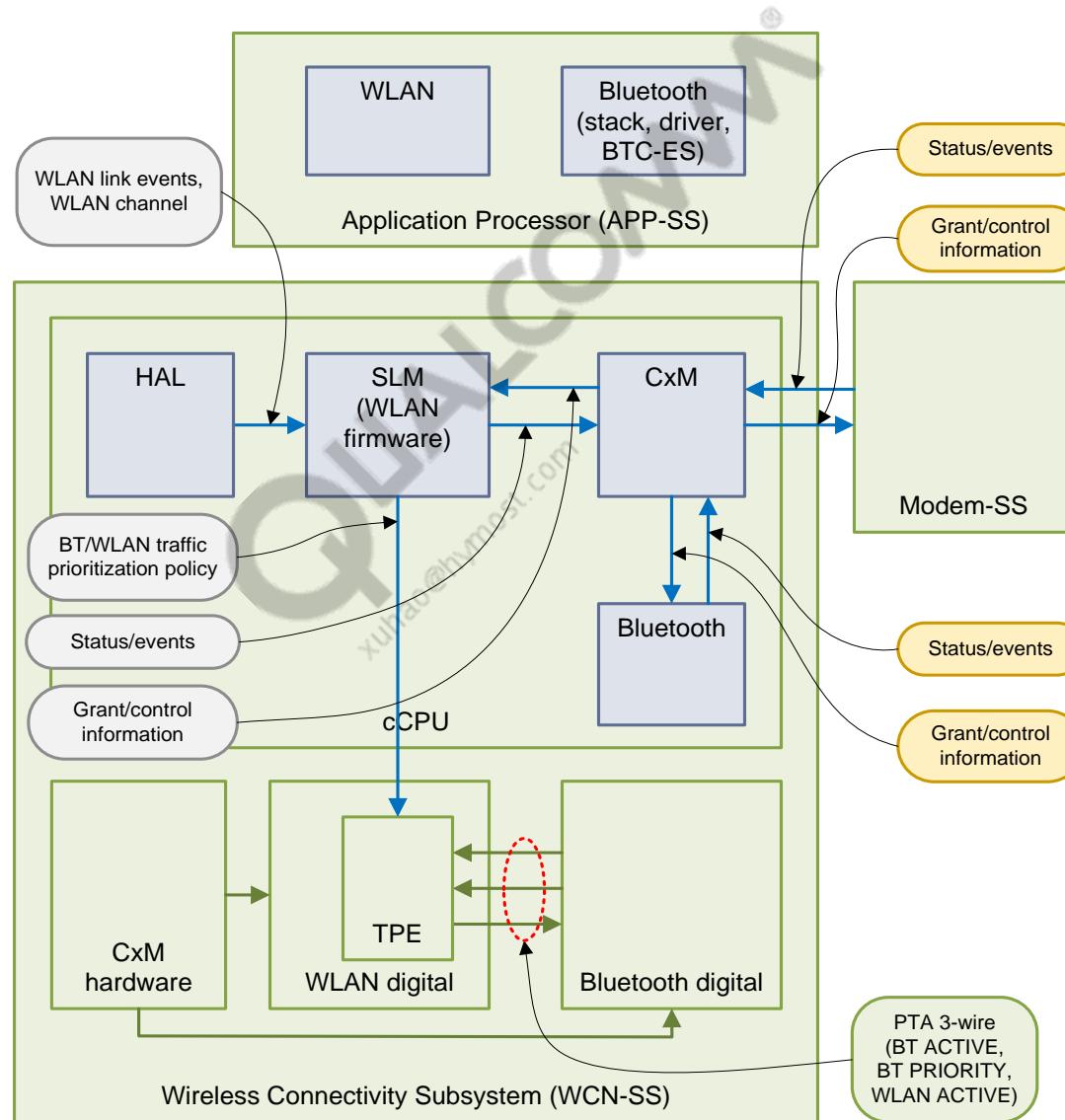
- All coexistence solutions require tradeoffs and our current focus is finding the right balance between LTE performance and Bluetooth/Wi-Fi
- Use either of these two approaches:
  - CxM
  - Filters
- RF filtering should always be used for LTE coexistence

# Solutions to LTE/Bluetooth/WLAN Coexistence (cont.)

## ■ Target coex performance improvements in each use case

Use cases	Filter only		CxM	
	LTE B7	LTE B40	LTE B7	LTE B40
<b>Bluetooth</b>				
▪ LTE VoIP call with Bluetooth headset ▪ LTE data while audio streaming on Bluetooth ▪ LTE video streaming with audio streaming on Bluetooth	Severe Bluetooth audio quality degradation in the worst case scenario (such as ISM filter shift toward B7)	▪ Not supported ▪ Severe Bluetooth audio quality degradation in most cases ▪ LTE can be desensed by Bluetooth	Full concurrency under most conditions	Full concurrency under most conditions
<b>WLAN</b>				
LTE with WLAN portable router	Full concurrency with proper AP channel selection	▪ Full concurrency with proper AP channel selection under most conditions except for highest LTE channels ▪ LTE can be desensed if WLAN Tx level is high	Full concurrency with proper AP channel selection	Full concurrency with proper AP channel selection under most conditions
LTE and WLAN data traffic offload (handover)	▪ Full concurrency when the WLAN AP channel frequency is away from LTE UL frequency ▪ Possible long latency in handover ▪ When LTE data is active, WLAN AP connection may be lost and vice versa	▪ More frequent latency in handover ▪ When LTE data is active, WLAN AP connection may be lost and vice versa	▪ Full concurrency under most conditions ▪ LTE and WLAN AP connection is reliably maintained	▪ Full concurrency under most conditions ▪ LTE and WLAN AP connection is reliably maintained

# CxM Architecture

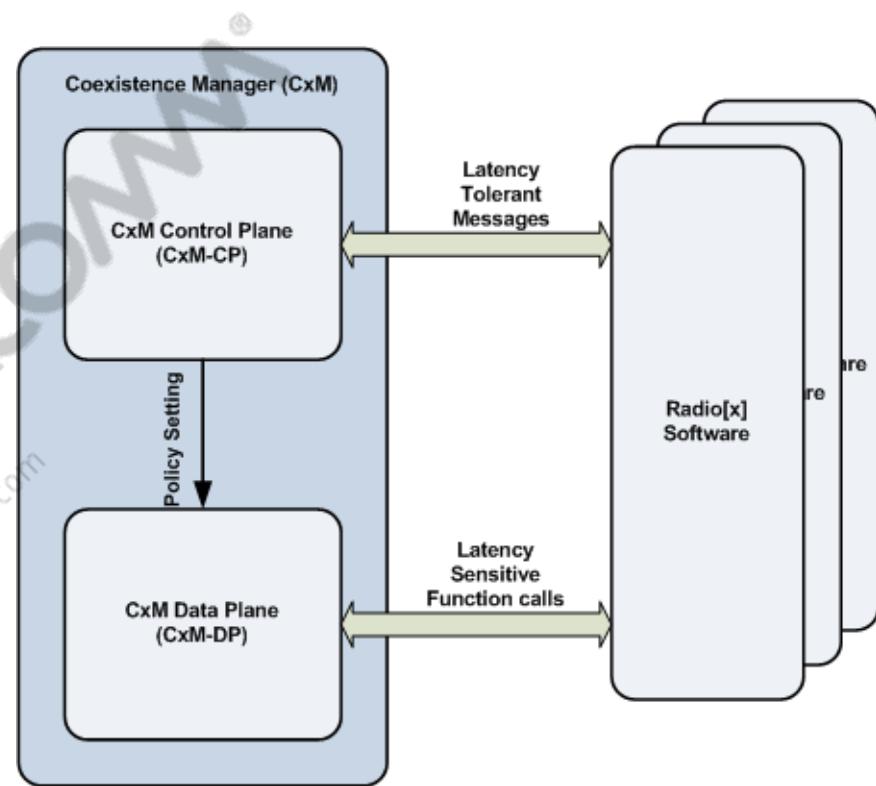


## CxM Architecture (cont.)

- CxM, Bluetooth, and WLAN software are running on the same cCPU
  - In Bluetooth software
    - ◆ Calculates the AFH channel map based on the WLAN channel and writes that channel map to the Bluetooth stack running on cCPU
  - All the BTC relevant information is available within cCPU
- LTE coexistence requires Modem SS feature to support CxM
- CxM hardware is mainly for LTE coexistence
- BTC utilizes PTA 3-wire signals between Bluetooth and WLAN hardware
  - Provides actual accurate timing for available interval to Bluetooth and WLAN

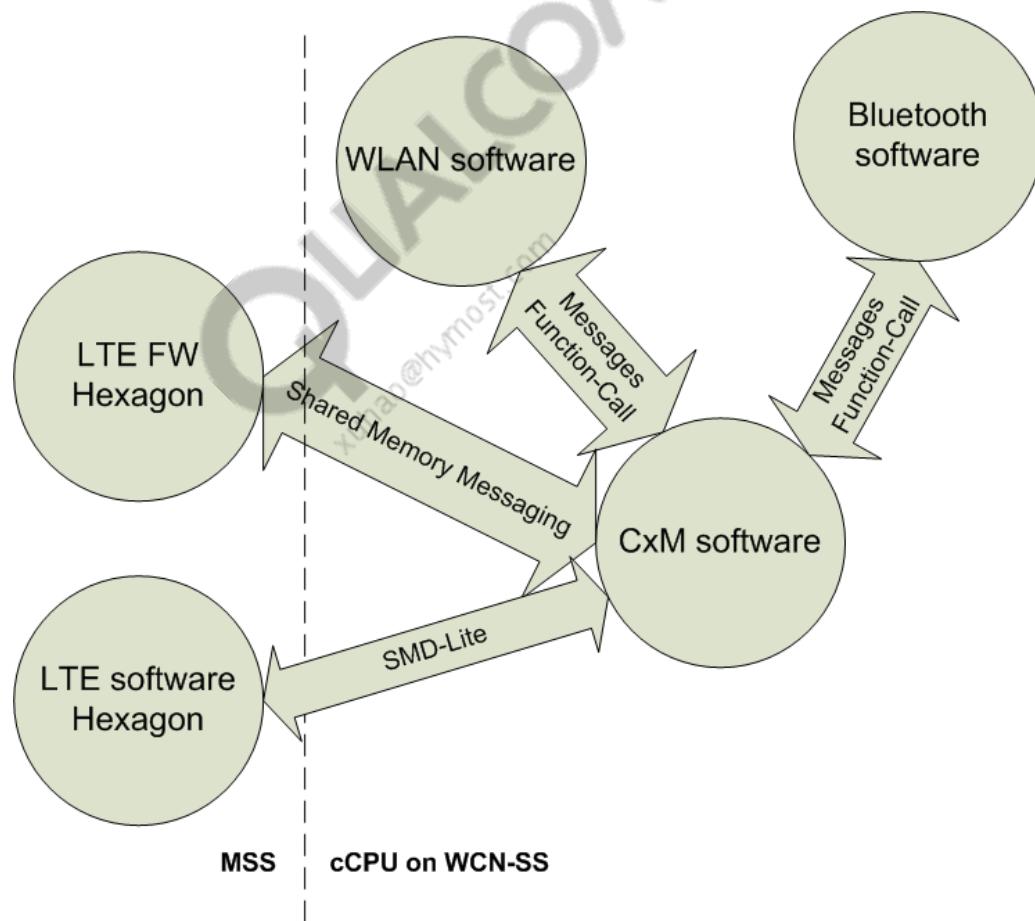
# CxM Architecture (cont.)

- CxM software on WCN-SS composed of two modules
  - CxM Control Plane (CxM-CP)
    - ◆ Lower-priority thread runs as a task on Wireless Connectivity Subsystem (WCN-SS)
    - ◆ Interfaces to Bluetooth, WLAN, and LTE modem to track the state of all radio technologies and set up the policy that allows the data plane to quickly make real-time decisions
  - CxM Data Plane (CxM-DP)
    - ◆ Real-time component
    - ◆ Makes real-time decision about granting LTE/BT/WLAN



## CxM Architecture (cont.)

- CxM software interfaces between software components on WCN-SS, as well as the LTE modem



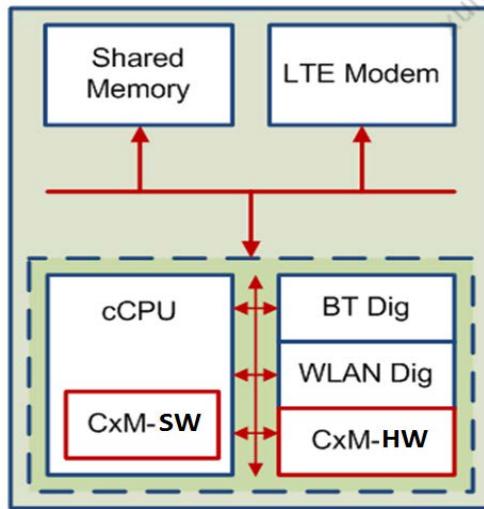
## LTE – CxM Interface

- To support LTE coexistence with ISM, CxM needs:
  - LTE timing transfer
  - Message passing between LTE and CxM
- Messages between LTE and CxM
  - Data plane messages (every millisecond)
    - ◆ LTE → CxM – Used to inform CxM of the details of the upcoming event in the next subframe
      - ▶ Event start and end times, event priority, frequency resource allocation, Tx power, etc.
    - ◆ CxM → LTE – To abort or grant LTE transmission based upon Bluetooth/WLAN events
  - Control plane messages (only sent when property changes)
    - ◆ LTE → CxM – Used to inform the CxM of state-based changes
      - ▶ For example, LTE center frequency, LTE modes of operation, etc.
    - ◆ CxM → LTE – Used to command LTE to perform a certain interference mitigation procedure
      - ▶ For example, CxM asks LTE to do power backoff

# Architecture Comparisons

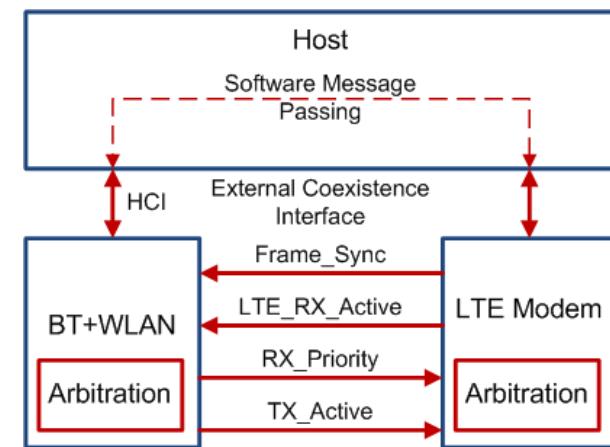
## ■ CxM architecture

- Centralized arbitration with next subframe look ahead and complete event context (event type, RSSI, freq, Tx power, priority, start time, duration, etc.)
- No external interface
- OS and Bluetooth host stack independent
- Licensee not responsible for coexistence integration



## ■ Competitor architecture

- Decentralized, uncoordinated arbitration without look ahead knowledge
- Slow (tens of milliseconds) context exchange via host (HCI)
- External multi-pin coexistence interface without distinction between Bluetooth and WLAN events
- OS and Bluetooth host stack dependent
- Licensee responsible for integration and ultimate performance (over multiple Bluetooth/WLAN and LTE suppliers)



# LTE – ISM Coexistence Manager System Algorithms

- LTE – ISM Coexistence Techniques

- Frequency domain techniques
- Power domain techniques
- Time domain techniques



## Frequency Domain Techniques

- LTE Inter-frequency or Inter-Radio Access Technology (Inter-RAT) handoff
  - LTE can handover to a different frequency or RAT when other techniques do not meet target performance
- Bluetooth AFH
  - Bluetooth can perform adaptive frequency hopping
    - ◆ Choose set of channels with minimum interference to/from LTE
- WLAN channel selection
  - For WLAN in Qualcomm Mobile Access Point SW (QC MobileAP) mode, WLAN can perform channel selection
    - ◆ Choose a channel as far as possible from LTE, i.e., Ch 1 for LTE in B7 and Ch 11 for LTE in B40
    - ◆ Channel selection can be done by end-user
    - ◆ There is a capability of limiting the range of allowed channels during the auto-channel selection, which can be set by phone manufacturer

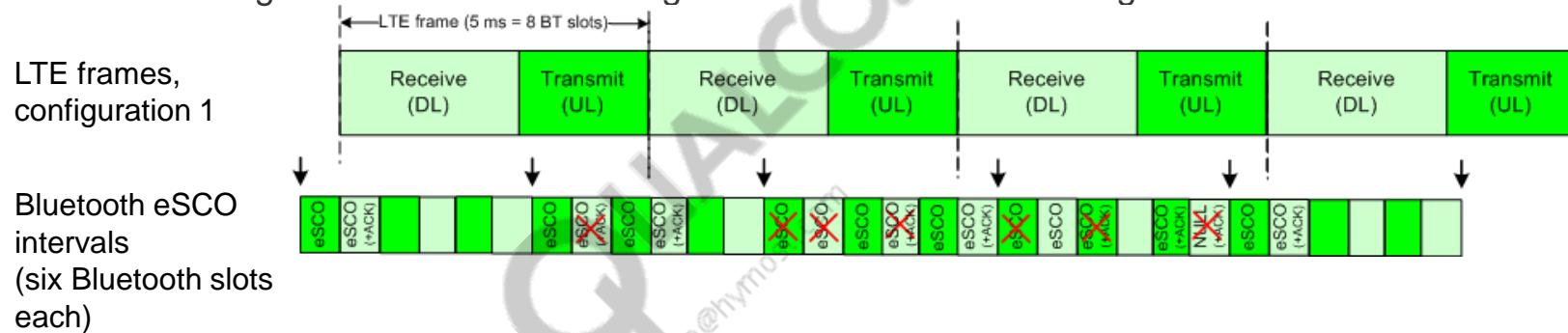
## Power Domain Techniques

- Backoff Tx power of the aggressor to help victim performance
  - For B7, can back off LTE Tx power to help Bluetooth audio quality
  - For B40, can back off BT Tx power or WLAN Tx power to reduce the interference to LTE DL
    - ◆ WLAN Tx power backoff is already implemented for EVM considerations
- Aggressor power Tx backoff is performed in an adaptive manner
  - Tx power backoff is performed only if the aggressor is identified as the source of dominant interference
  - Amount of power backoff is chosen to provide the best user experience
- CxM collects all the relevant information from victim and aggressor to make the intelligent decision on the amount of backoff needed

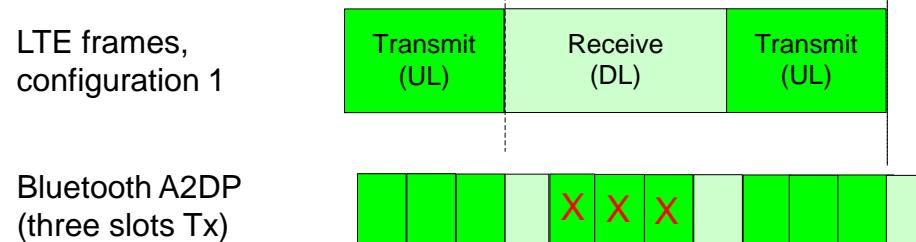
# LTE – ISM Coexistence Manager System Algorithms (cont.)

## Time Domain Techniques – Bluetooth Frame Alignment

- This slide is applicable to LTE band 40 or TDD bands
- Bluetooth being the master allows for alignment with LTE frame timing



- Streaming audio (A2DP) can also be supported with three slot packets

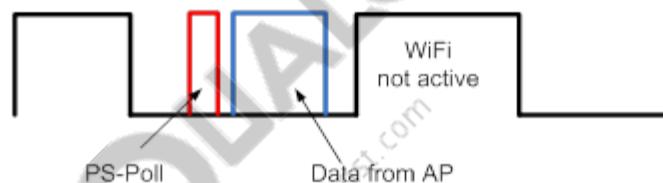


- For Bluetooth slave in B40, frame alignment can still be used in most cases for eSCO (and probably also for A2DP) using a role switch request to become the master

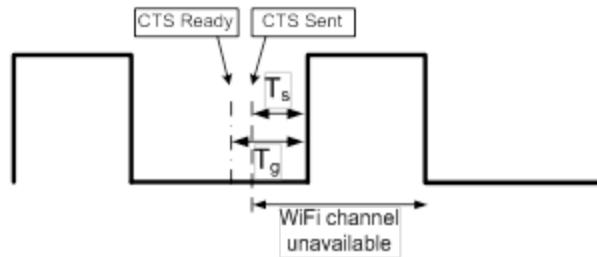
# LTE – ISM Coexistence Manager System Algorithms (cont.)

## Time Domain Techniques – WLAN Flow Control

- PS-POLL mechanism for the STA case
  - Used to receive packets opportunistically during LTE Rx or LTE Tx gaps
  - PS-POLL needs to be sent a guard time ahead of LTE UL to ensure Rx



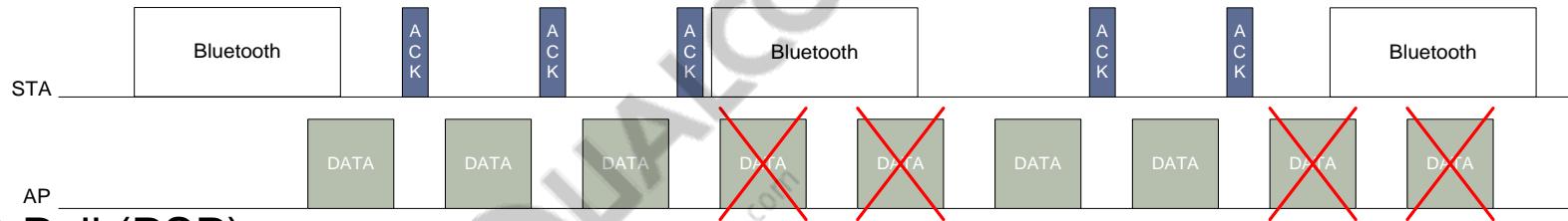
- CTS-to-Self for the QC MobileAP case
  - Used to avoid WLAN Rx during LTE UL, e.g., to use only LTE DL to receive packets
  - CTS-to-Self needs to be ready to send a guard time ahead of LTE UL



# AP Control Techniques for BT/WLAN Coexistence

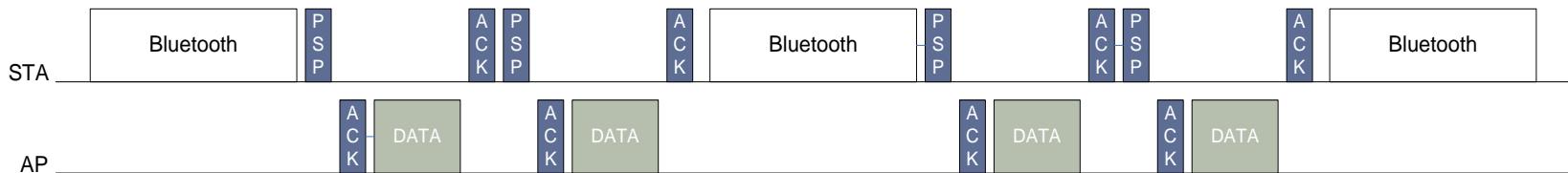
## ■ No AP control

- Allows the AP to send data at-will and rely on packet retransmissions and no rate adaptation



## ■ PS-Poll (PSP)

- STA is in Power Save mode, requests one packet at a time
- Most reliable technique, but has the most overhead



# AP Control Techniques for BT/WLAN Coexistence (cont.)

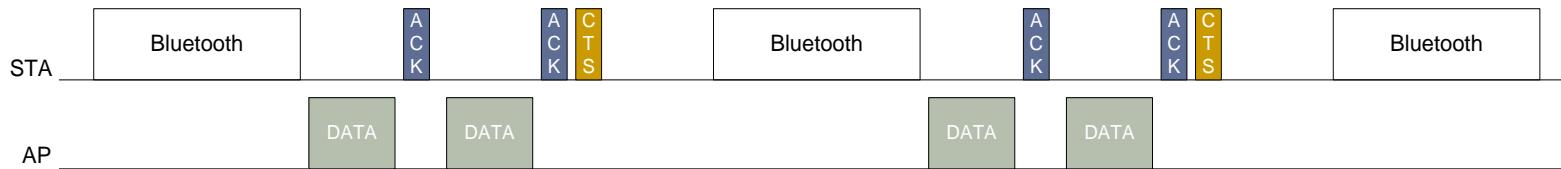
## ■ Power Mode Transition (PMT)

- STA sends frames to enter and exit power save around Bluetooth
- Lower overhead, but not practical for transitions <10 ms apart



## ■ CTS

- STA prohibits all AP/other STA activity during Bluetooth
- Best throughput at high SNR but unfriendly to other users



# Bluetooth/WLAN Coexistence Modes

- Different modes for WLAN and Bluetooth coexistence
  - Defines the way that time intervals are divided between WLAN and Bluetooth
  - Managed by CxM-CP
    - ◆ CxM-CP decides which mode will be best at the specific time based on information provided by WLAN and Bluetooth
    - ◆ CxM-CP sets policy to CxM-DP based on decision made for specific mode to use
    - ◆ CxM-DP controls Bluetooth/WLAN software based on policy made by CxM-CP
- Static modes
  - Bluetooth Takes All
  - WLAN Takes All
  - PTA
  - PTA with No Bluetooth Low (ignoring Low Priority Bluetooth interval)
- Dynamic modes
  - Dynamic Long Interval (generally used for A2DP/ACL)
    - ◆ Dynamic Long Bluetooth Interval
    - ◆ Dynamic Long WLAN Interval
  - Dynamic Short Interval (generally used for xSCO)
    - ◆ Dynamic Short Bluetooth Interval
    - ◆ Dynamic Short WLAN Interval

# Bluetooth/WLAN Coexistence Modes (cont.)

## ■ Comparison of coexistence modes

Mode	Bluetooth high priority	Bluetooth low priority	WLAN STA	WLAN Rx	WLAN Tx
Bluetooth Takes All	Pre-empt WLAN	First come, first served	In PS mode	Restricted	Restricted
WLAN Takes All	Ignored	Ignored	Not in PS mode	Unrestricted	Unrestricted
PTA	Preempt WLAN	First come, first served	Not in PS mode	Unrestricted	Unrestricted
PTA with No Bluetooth Low	Preempt WLAN	Ignored	Not in PS mode	Unrestricted	Unrestricted
Dynamic Long Bluetooth Interval	Preempt WLAN	First come, first served	In PS mode	Restricted	Restricted
Dynamic Long WLAN Interval	May be ignored while PM transitions are being sent; preempts WLAN otherwise	Ignored	May or may not be in PS mode	Unrestricted	Unrestricted
Dynamic Short Bluetooth Interval	Preempt WLAN	First come, first served	In PS mode	Restricted	Restricted
Dynamic Short WLAN Interval	Ignored	Ignored	In PS mode	Unrestricted	Unrestricted

# Bluetooth/WLAN Coexistence Profiles and Parameters

- These parameters can be added/edited in
  - /data/misc/wifi/WCNSS\_qcom\_cfg.ini file to tune BTC execution modes for AP, STA and P2P
- BtcExecutionMode
  - This parameter can be tuned to adjust prefer BT vs WLAN

Profile	Numerical value	Comments
Balanced	0 (Default)	Optimized for best BT and WLAN performance
A2DP	5	Optimized for best BT Audio Quality (A2DP) in heavy interference at the cost of WLAN throughput

# Bluetooth/WLAN Coexistence Profiles and Parameters (cont.)

## ■ BtcConsBtSlotToBlockDuringDhcp:

- This parameter can be tuned to ensure WLAN IP address delivery via DHCP while SCO voice call is active. This will be done at the cost of BT Voice call Audio Quality. Please note that this parameter will have no effect after IP address is obtained via DHCP.

Values: 0 : No protection (Default)  
1 : Best WLAN DHCP protection  
255 : Least protection for DHCP  
default : 8

## ■ BtcA2DPDhcpProtectLevel:

- This parameter can be tuned to ensure WLAN IP address delivery via DHCP while A2DP is active. This will be done at the cost of BT A2DP Audio Quality. Please note that this parameter will have no effect after IP address is obtained via DHCP.

Values: 0 (Protect WLAN DHCP best) to 15 (protect A2DP best)  
default : 7

# Bluetooth/WLAN Coexistence Tunable Parameters

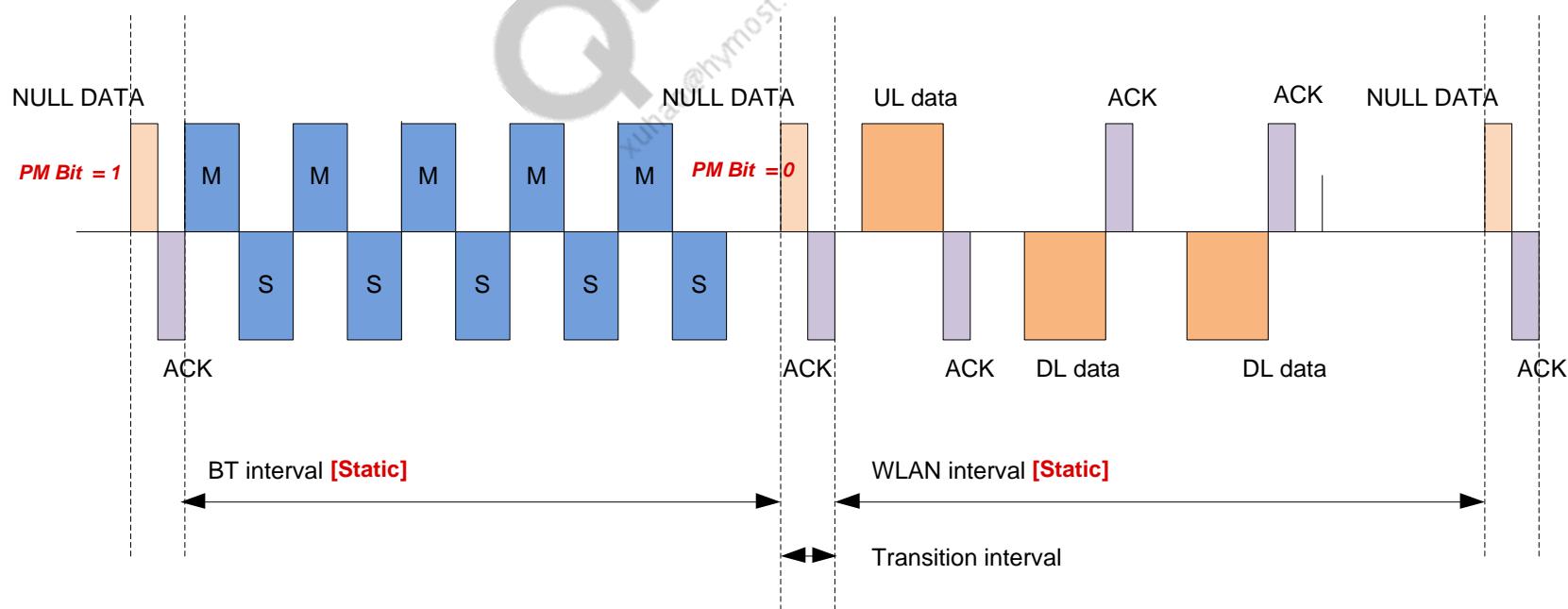
- Coexistence enhancement using tunable parameters
  - Parameters are added to provide BT/WLAN coexistence flexibility and ensure functional operation of each radios simultaneous run

Parameters	Definition
btcStaticLenInqBt	BT interval length during BT Inquiry
btcStaticLenPageBt	BT interval length during BT Paging
btcStaticLenConnBt	BT interval length during BT connection setup
btcStaticLenLeBt	BT interval length during BT LE scan
btcStaticLenInqWlan	WLAN interval length during BT Inquiry
btcStaticLenPageWlan	WLAN interval length during BT Paging
btcStaticLenConnWlan	WLAN interval length during BT connection setup
btcStaticLenLeWlan	WLAN Interval length during BT LE scan
btcDynMaxLenBt	Dynamically adjust BT interval during ACL Active (A2DP/OPP)
btcDynMaxLenWlan	Dynamically adjust WLAN interval during ACL Active (A2DP/OPP)
btcMaxScoBlockPerc	Maximum SCO skipping percentage
btcDhcpProtOnA2dp	Ensure DHCP reliability during A2DP
btcDhcpProtOnSco	Ensure DHCP reliability during SCO

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

## ■ Static Timesharing Timeline

- Static timesharing is used for BT link maintenance operations
  - ◆ Inquiry, Paging, Connection Setup, LE (Low Energy) Scan
- Coex scheduler divides time into BT and WLAN intervals
  - ◆ WLAN get into “flow controlled” state at AP - Send trigger frame with PM bit =1
  - ◆ WLAN get out from “flow controlled” state at AP - Send trigger frame with PM bit=0



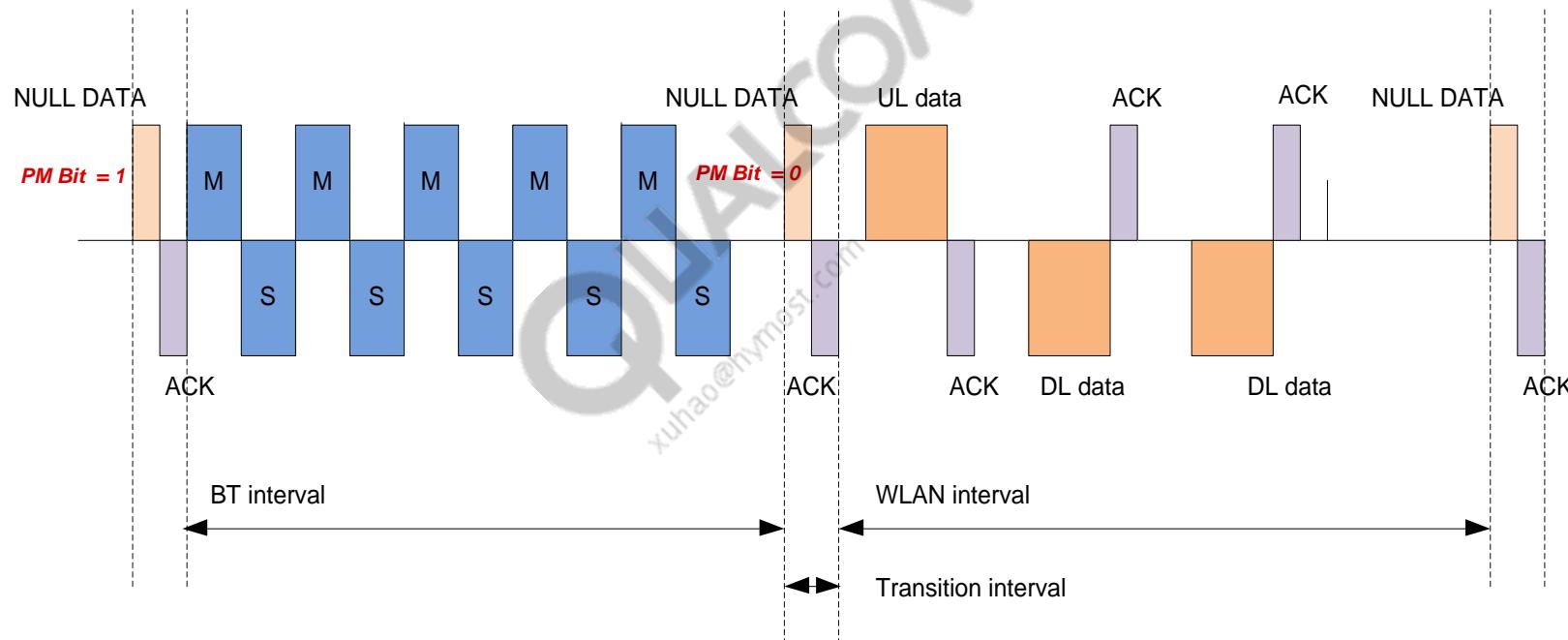
# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

## ■ Static Timesharing Timeline (cont.)

- BT and WLAN intervals can be configured with static(fixed) values
- Each BT operations have individual configurable values – [Mix/Max/Default]
  - ◆ `btcStaticLenInqBt` : Interval length given to BT while BT inquiry - [5ms/500ms/120ms]
  - ◆ `btcStaticLenPageBt` : Interval length given to BT while BT paging - [5ms/500ms/10ms]
  - ◆ `btcStaticLenConnBt` : Interval length given to BT while BT connecting - [5ms/500ms/10ms]
  - ◆ `btcStaticLenLeBt` : Interval length given to BT while BLE scanning - [5ms/500ms/10ms]
  - ◆ `btcStaticLenInqWlan` : Interval length given to WLAN while BT inquiry - [0ms/500ms/30ms]
  - ◆ `btcStaticLenPageWlan` : Interval length given to WLAN while BT paging - [0ms/500ms/0ms]
  - ◆ `btcStaticLenConnWlan` : Interval length given to WLAN while BT connecting - [0ms/500ms/0ms]
  - ◆ `btcStaticLenLeWlan` : Interval length given to WLAN while BLE scanning - [0ms/500ms/0ms]
- Set WLAN interval to "0ms" to give BT 100% of time.

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Dynamic Long (A2DP, OPP) timeline

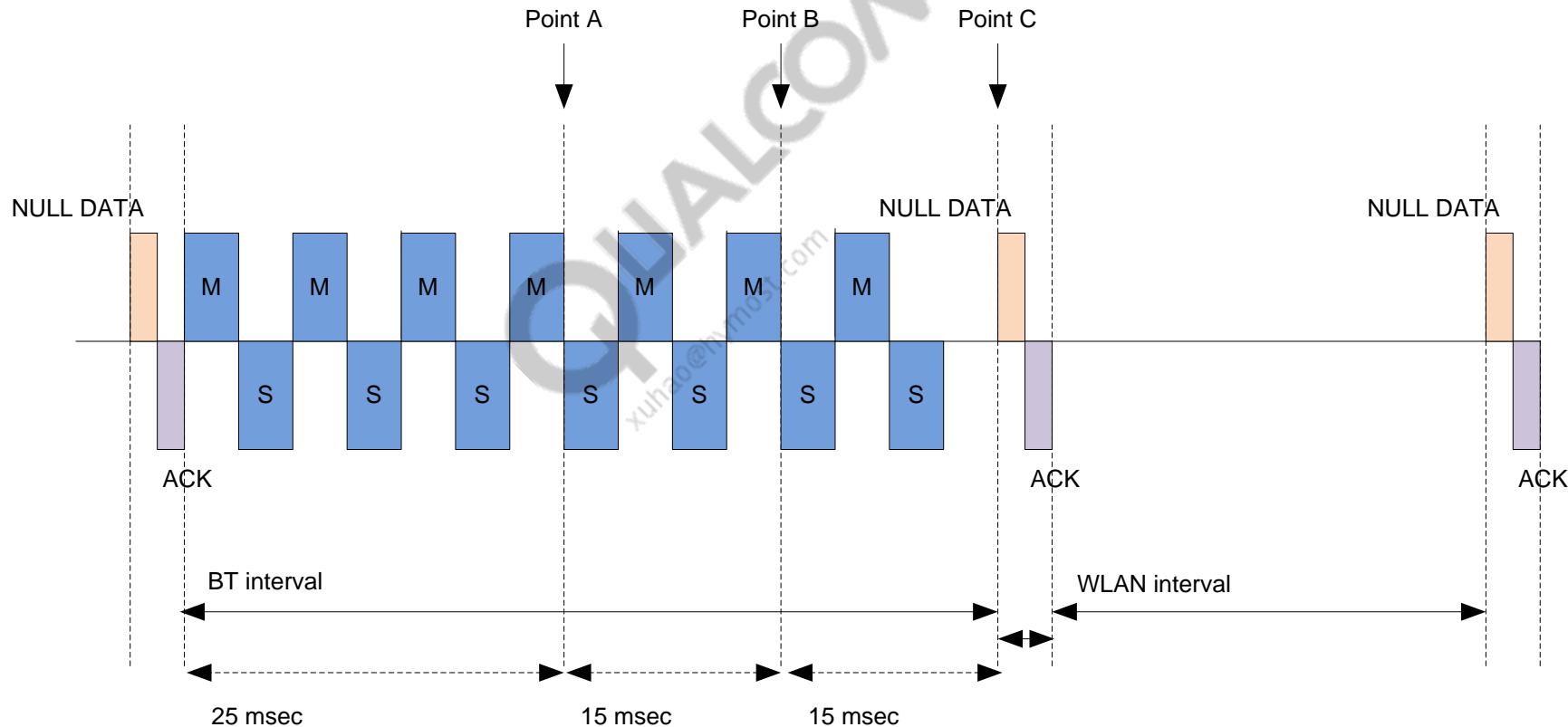


# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Dynamic Long (A2DP, OPP) timeline (cont.)
  - “Dynamic Long” scheduling is used for BT Active ACL links
  - Coex scheduler divides time into BT and WLAN intervals
    - During BT intervals, WLAN enters “flow controlled” state at AP by exchanging a data frame with PM bit set to “1”
    - During WLAN interval, WLAN exits “flow controlled” state at AP by exchanging a data frame with PM bit set to “0”
  - BT intervals are dynamic in length
    - BT transmit/receive metrics are sampled at short intervals
    - If BT metrics indicates more time is needed, BT time is extended
    - A maximum BT grant is enforced, after which WLAN interval will start regardless of BT queue state
    - `btcDynMaxLenBt` : Maximum Interval grant to BT
  - WLAN intervals are dynamic in length
    - WLAN interval length also varies based on BT metrics
    - `btcDynMaxLenWlan` : Maximum Interval grant to WLAN

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Dynamic Long (A2DP, OPP) timeline, **scheduler detail**

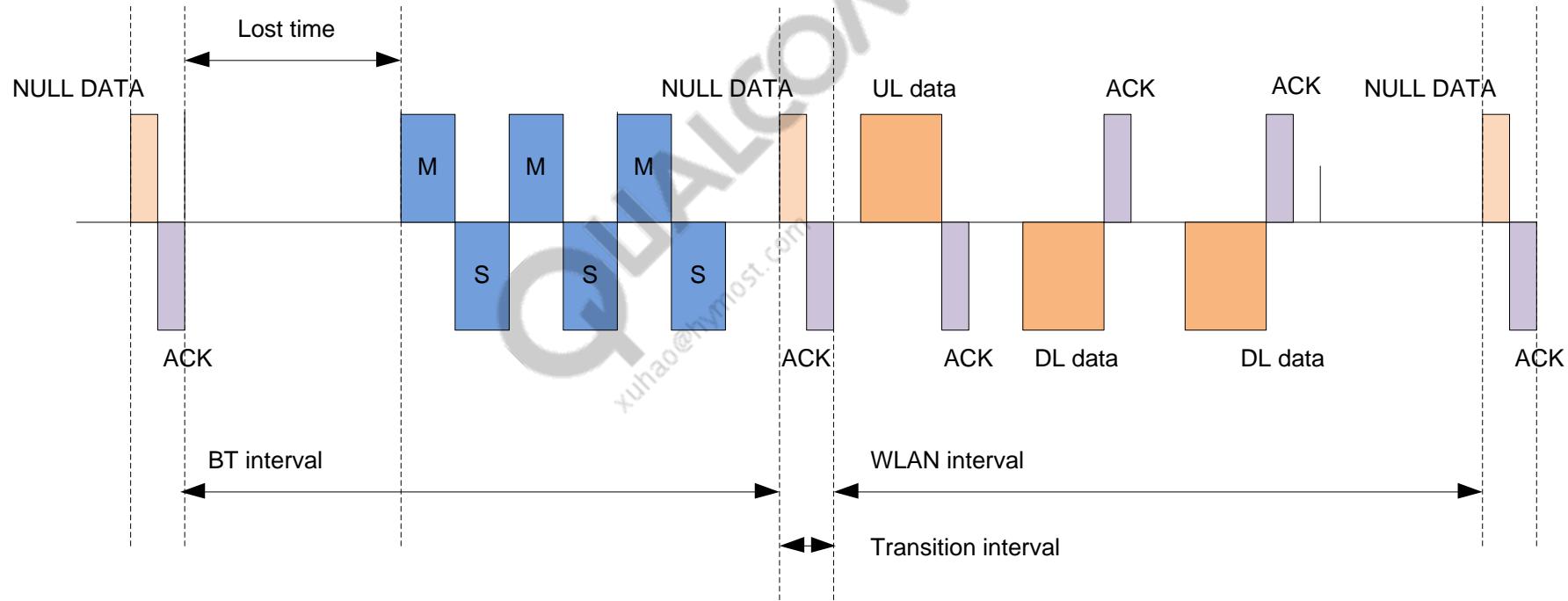


# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Dynamic Long (A2DP, OPP) timeline, **scheduler detail** (cont.)
  - BT is given a “primary” interval followed by some number of “extension” intervals
    - Primary interval is 25msec
    - Extension interval is 15msec
    - Number of extensions allowed is calculated based on the “maximum length” configured
  - Extension intervals are granted based on
    - Last frame type sent
    - Peer flow control state
    - Amount of data in the transmit queue
    - Receive activity in the latest BT interval
    - BT scan activity
    - In figure, this is checked at points A,B & C with A & B revealing that extensions are needed and C revealing that no extension is needed
  - WLAN interval
    - Default length is the “maximum” length configured
    - This length may be shorted based on various criteria
    - Length is calculated at point C in the diagram
  - WLAN interval shortening may occur based on
    - History of BT extension grants
    - History of “lost time” measurements

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

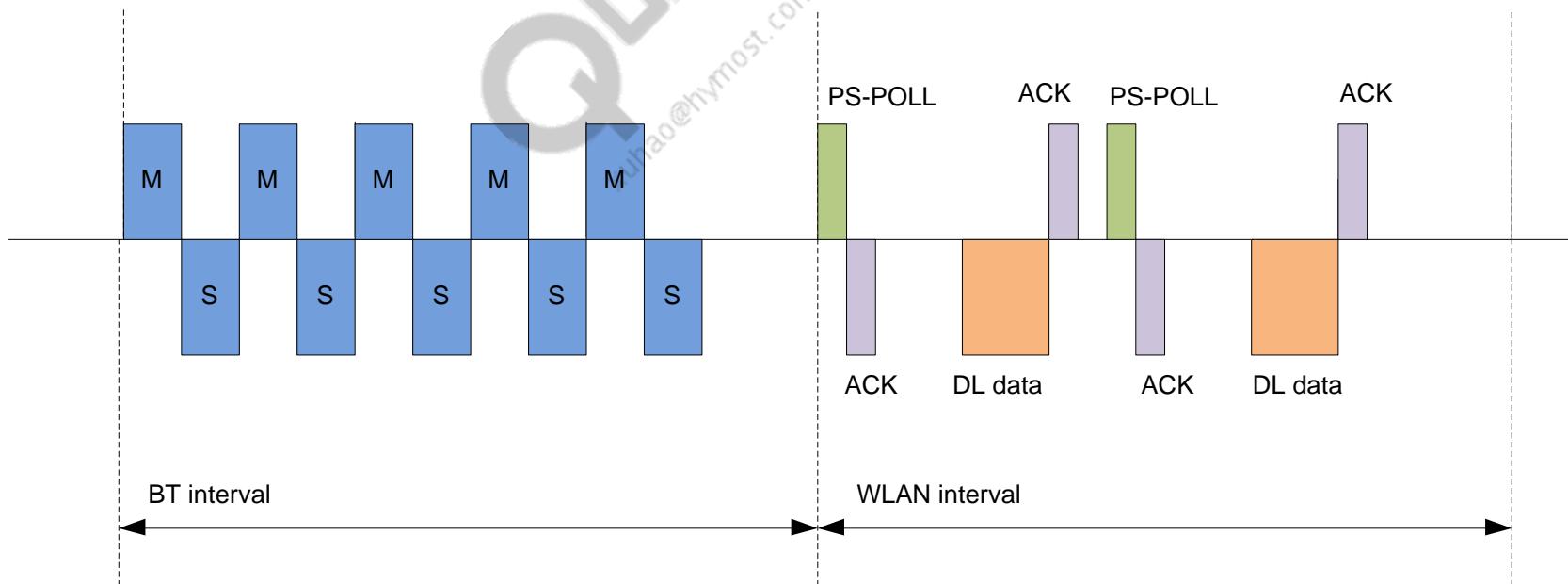
- Dynamic Long (A2DP, OPP) timeline, Lost time



- Dynamic Long (A2DP, OPP) timeline, **Lost time** (cont.)
  - BT usage of the BT interval varies according to Role
    - When the phone is in the BT Master role, it may immediately start sending ACL data when the BT interval starts
    - When the Phone is in the BT Slave role, it must wait for the peer to poll it before it may send
    - The time used looking for Master Polls is lost time
  - The Master polling behavior may vary from headset to headset
    - Headsets may resort to polling back off for different absence periods
    - Headsets may use different polling frequencies once they are backed off
    - Behavior responds to WLAN interval length

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

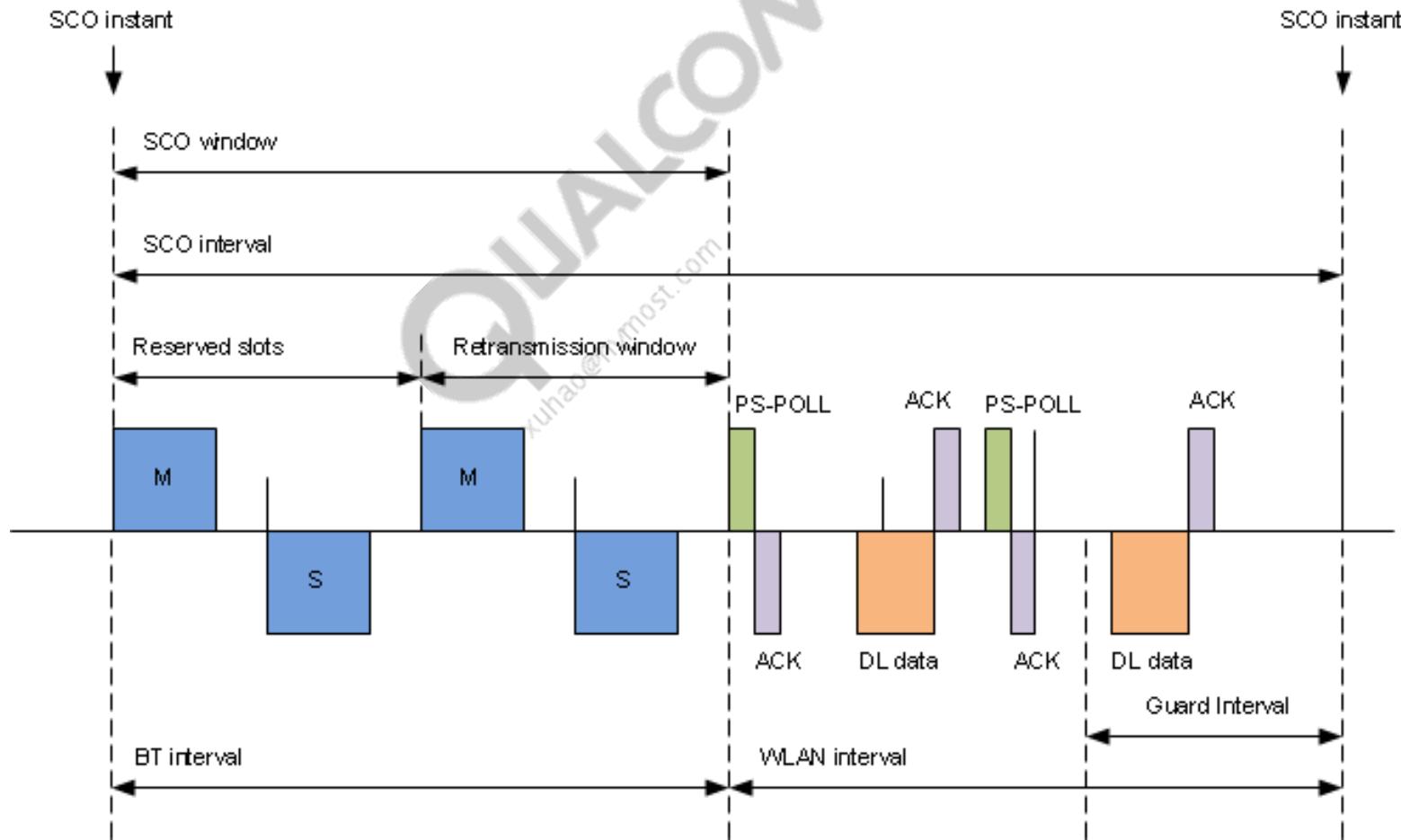
- Dynamic Long (A2DP, OPP) timeline, w/PS-POLLS
  - PS-POLL frames may be used instead of data frames with PM bit alternating
    - This is used when the RF conditions do not support reasonable response time alternating PM bits
    - This is a “fall back” mode in which the goal is to maintain the WLAN link connection



- SCO Timeline
  - Coex scheduler synchronizes to BT timeline, divides time into BT and WLAN intervals based on synchronous BT timeline.
  - BT requires most of every SCO transaction to maintain audio quality, so WLAN is rarely scheduled over BT activity
    - If BT does not request the Retransmission window, this is given to WLAN
  - WLAN enters “flow controlled” state at AP at the beginning of the SCO session, individually polls for downlink packets from AP with “PS-POLL” frames
  - Coex scheduler measures time needed for each “PS-POLL to DL Data” transaction
    - Coex scheduler uses these measurements to calculate and enforce a “Guard Interval” in which no more PS-POLL frames are attempted

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

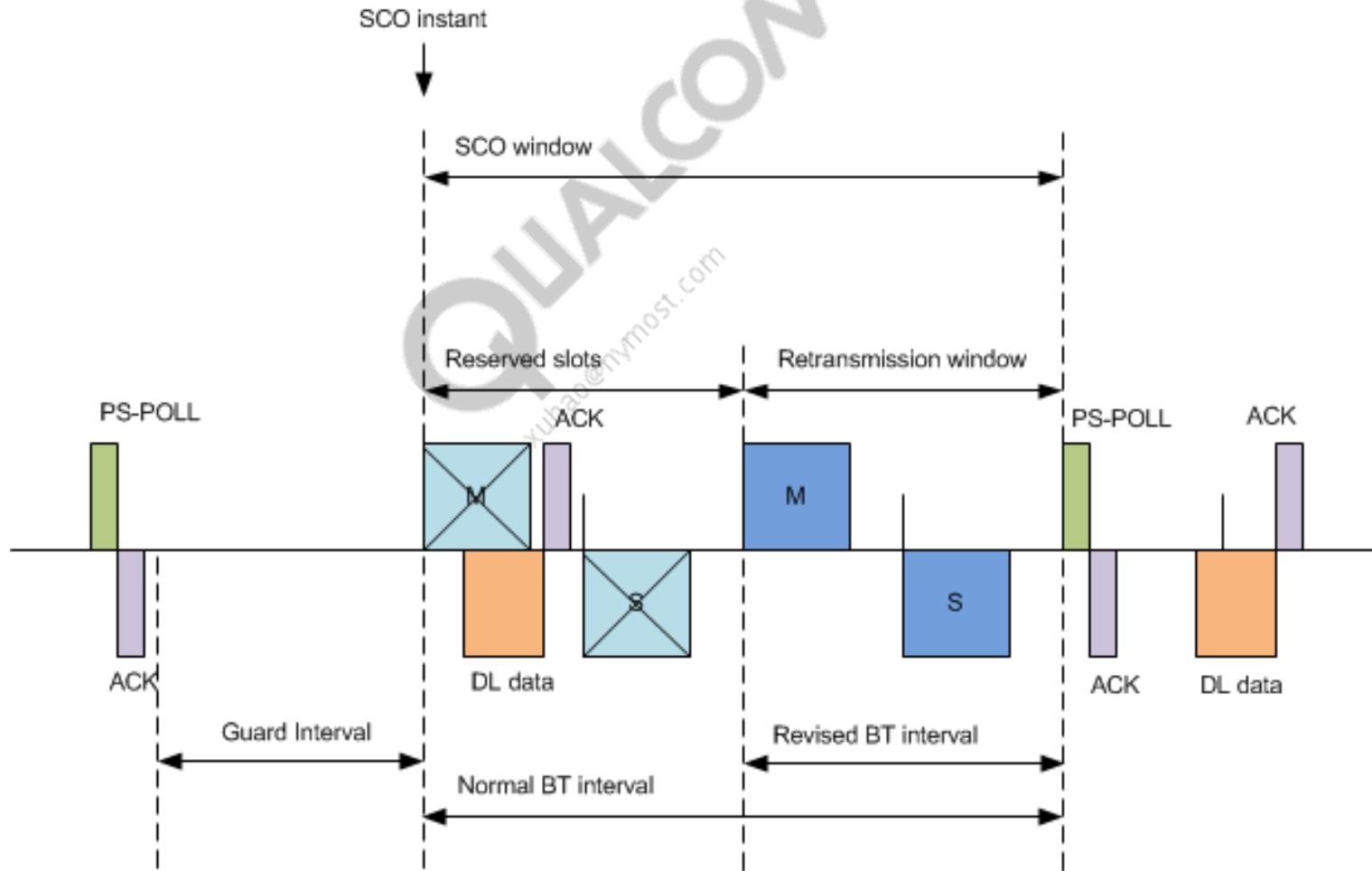
- SCO Timeline (cont.)



- SCO Timeline Special Case – **Late DL data**
  - What happens if the DL data frame comes after the BT interval has started?
    - If BT and WLAN traffic collide, this can lead to WLAN link stability issues.
  - If the DL data frame arrives after the “Reserved Slots” have started and before the “Retransmission Window”, the BT interval may be delayed to start until the “Retransmission Window” to allow WLAN to complete
    - This likely has little impact on BT audio quality as the retransmission window is still available for BT SCO use
  - If the DL data frame arrives after the “Retransmission Window” has started, this BT interval may be skipped to allow WLAN to complete
    - Too much skipping can impact audio quality
    - **btcMaxScoBlockPerc** : BT SCO interval may only be skipped less than a configurable percentage aimed at minimizing BT audio quality impact while enhancing WLAN link stability - [0%/100%/1%]

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- SCO Timeline Special Case – Late DL data (cont.)



# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Special Topics: WLAN connection setup
  - WLAN connection setup is delimited by five link state messages
    - Pre-assoc
    - Post-assoc
    - Set key done
    - Enter BMPS
    - Link terminated
  - This yields 5 WLAN link states we can recognize in CoEX
    - No connection – before “pre-assoc” and after “link terminated”
    - Association exchange – between “pre-assoc” and “post-assoc”
    - Security exchange – between “post-assoc” and “set key done”
    - DHCP exchange – between “set key done” and “enter BMPS”
    - Steady state – between “enter BMPS” and “link terminated”

- Special Topics: WLAN connection setup, [Association](#)

- WLAN has no ability to flow control the AP before association is complete
  - Timesharing may occur, but the downlink will not be explicitly flow controlled
- A2DP
  - “Dynamic long” scheduling is the base behavior
  - No data frames with the PM bit set/cleared or PS-POLLs frames are sent
- SCO
  - “Dynamic short” scheduling is the base behavior
  - No data frames with the PM bit set/cleared or PS-POLLs frames are sent

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

- Special Topics: WLAN connection setup, **Security**
  - WLAN may flow control the AP starting at this phase
  - A2DP
    - “Dynamic long” scheduling is the base behavior
  - SCO
    - “Dynamic long” scheduling is the base behavior
    - We use “long” type scheduling here because the security exchange often occurs at 1Mbps rate, and the longest security frames will not fit between SCO traffic
    - The use of “long” allows the security exchange to take place as well as passing some audio

# Bluetooth/WLAN Coexistence Tunable Parameters (cont.)

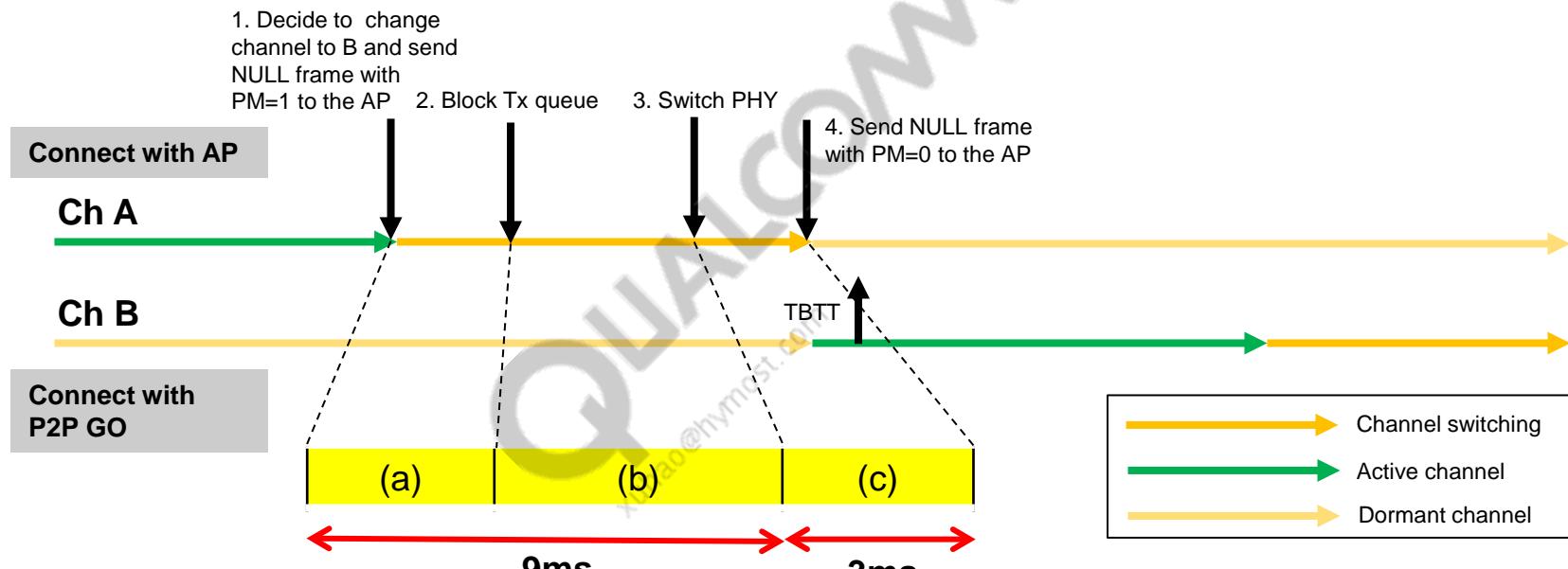
- Special Topics: WLAN connection setup, **DHCP**
  - WLAN DHCP traffic utilize broadcast frames
    - Broadcast frames defy Coex DL flow control methods
    - DL Broadcast frames are sent at the WLAN beacons (in Coex cases)
    - Solution is to synchronize Coex scheduler to WLAN beacons, make “holes” for WLAN broadcast traffic during DHCP exchange
  - A2DP
    - “Dynamic long” scheduling is the base behavior
    - Schedule WLAN intervals in windows around WLAN beacons during DHCP exchange
    - No expected effect on Audio quality
    - **btcDhcpProtOnA2dp** : A2DP DHCP protection toggle [OFF/ON – Default: ON]
  - SCO
    - “Dynamic short” scheduling is the base behavior
    - Block SCO traffic in windows around WLAN beacons during DHCP exchange
    - Minor effect on Audio quality until DHCP is completed
    - **btcDhcpProtOnSCO** : SCO DHCP protection toggle [OFF/ON – Default: OFF]

# BTC Concurrency – Multi-Channel Concurrency (MCC) Overview

- Time slicing is done between two WLAN BSSs
- Supported BSS types
  - Infra-structure client (STA)
  - P2P GC (Client)
  - P2P GO
- Scheduling considerations
  - Beacon protection
  - Fairness
    - ◆ Static fairness – Pre-Configured duty cycle
    - ◆ Dynamic fairness – Link usage sending, weighting toward larger consumer

# BTC Concurrency – Multi-Channel Concurrency (MCC) Scheduler

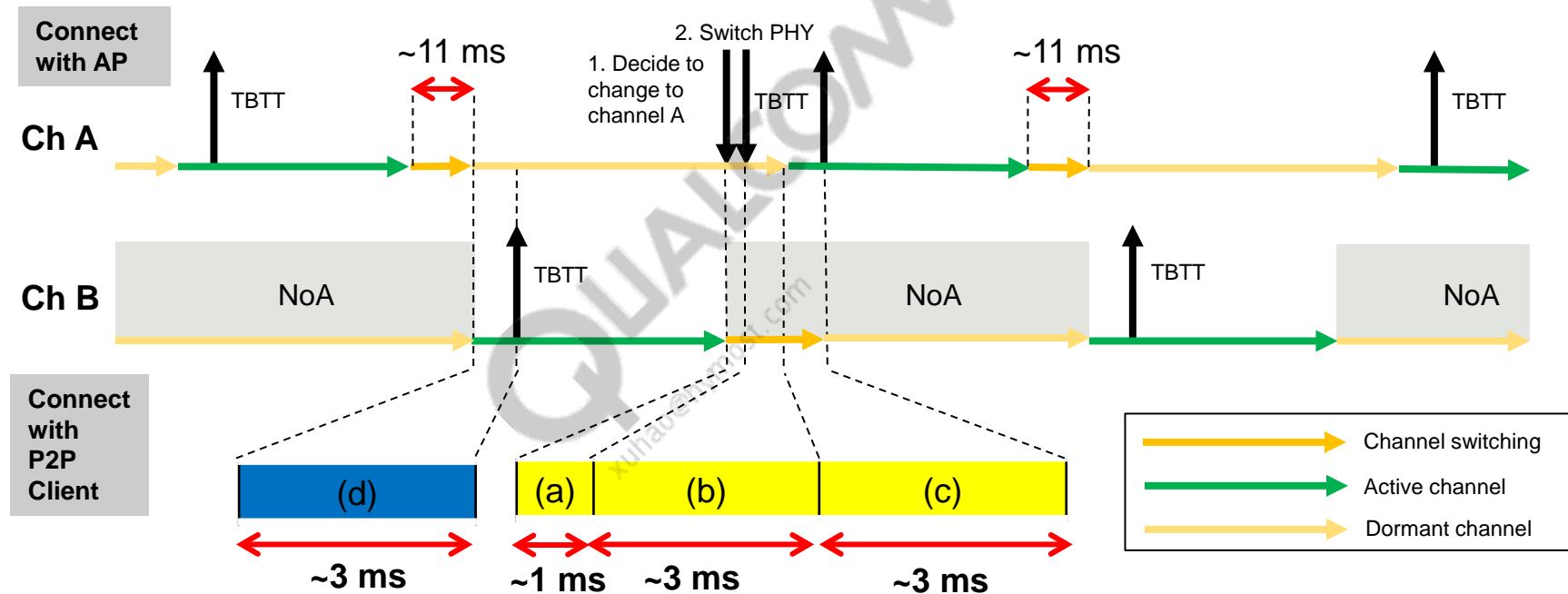
## ■ Channel changing for STA + P2P GC (Client)



- Approx. 12 ms (a)+(b)+(c) channel switch overhead, including
  - (a) = Timeout for transmitting null data; worst case will be 9 ms
  - (b) = Rx drain timeout as there are several APs around which take a while to honor the PM transition request coming from STA; if (a) has 9 ms, it can be ignored
  - (a) +(b) = Maximum 9 ms
  - (c) = PHY switch timeout; mostly calibration, no Rx and Tx

# BTC Concurrency – Multi-Channel Concurrency (MCC) Scheduler (cont.)

- Channel changing for STA + P2P GO



- Approx. 4 ms (a)+(b)+(c)-(d) channel switch overhead, including
  - (a) = Tx early stop
  - (b) = PHY switch timeout
  - (c), (d) = Early parking to other channel

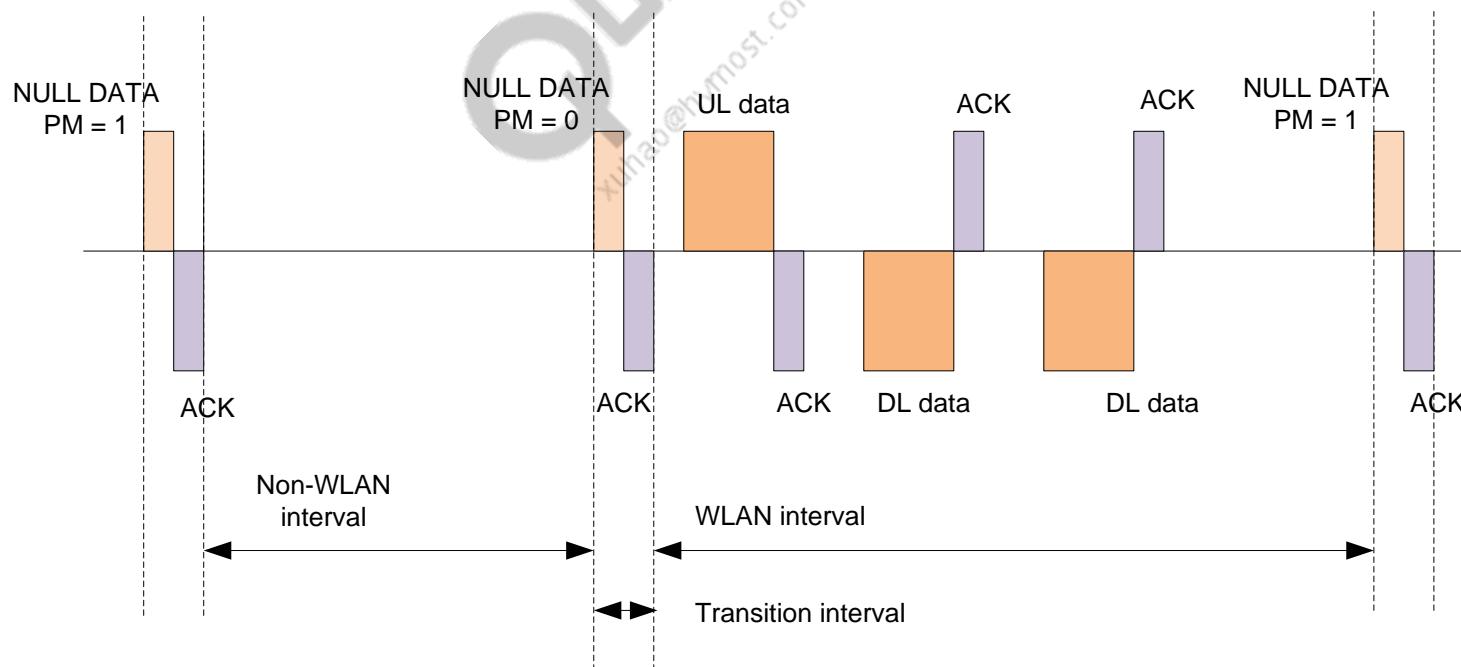
# BTC Concurrency – Time Slicing

- Both MCC and BTC require time slicing between mutually exclusive consumers of the antenna
- On the WLAN side, this requires flow control of one's self and one's peers to shape traffic into the WLAN/BSS slices
- Methods of flow control
  - Several derive from standard provided mechanisms for power savings
    - ◆ PM state, PS-POLLs, Notice of Absence (NOA)
  - Others depend on direct NAV manipulation and usage
    - ◆ CTS2SELF

# BTC Concurrency – Time Slicing (cont.)

## PM State with PM State Transitions

- APs and P2P GOs allow clients to enter power-managed state in which traffic must be buffered for that client
- By entering and exiting this state at the AP/P2P GO, a client can shape traffic into larger slices



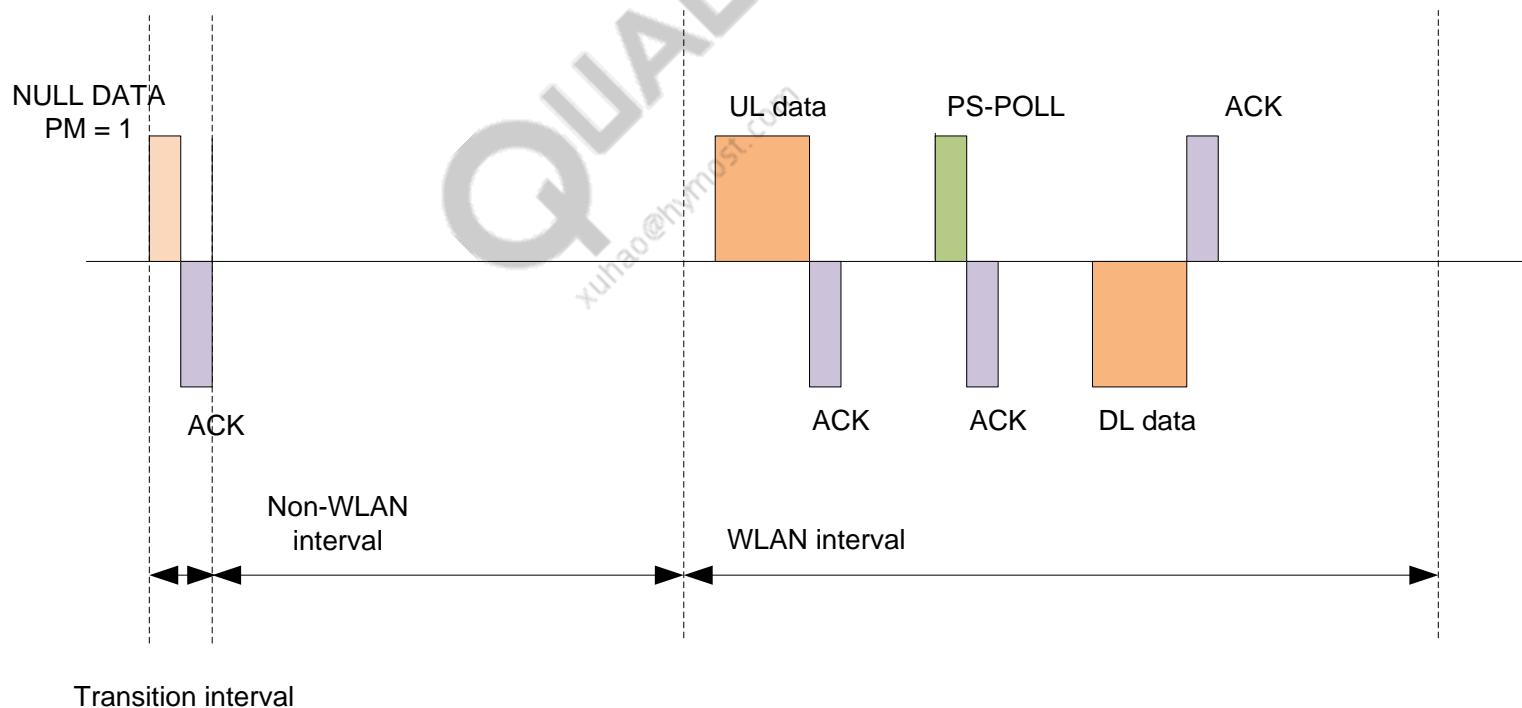
## PM State with PM State Transitions

- Advantages
  - Once a PM state is entered, it is “sticky”
    - ◆ Does not require any additional transmissions to maintain
  - No per-packet overhead
- Disadvantages
  - State transition overhead
    - ◆ Successful transmission and ACK of data frame is required
    - ◆ In poor RF conditions, this can take tens of milliseconds
  - AP/P2P GO behavior may vary from peer-to-peer (IOT)
    - ◆ Response time to PM mode entry, i.e., time from ACK'ing the control frame to stopping transmission of data
    - ◆ Response time to PM mode exit, i.e., time from ACK'ing the control frame to starting transmission of data

# BTC Concurrency – Time Slicing (cont.)

## PM State with PS-POLLS

- Enter PM state at AP/P2P GO and never exit
- Individually poll for downlink data packets using PS-POLL frames



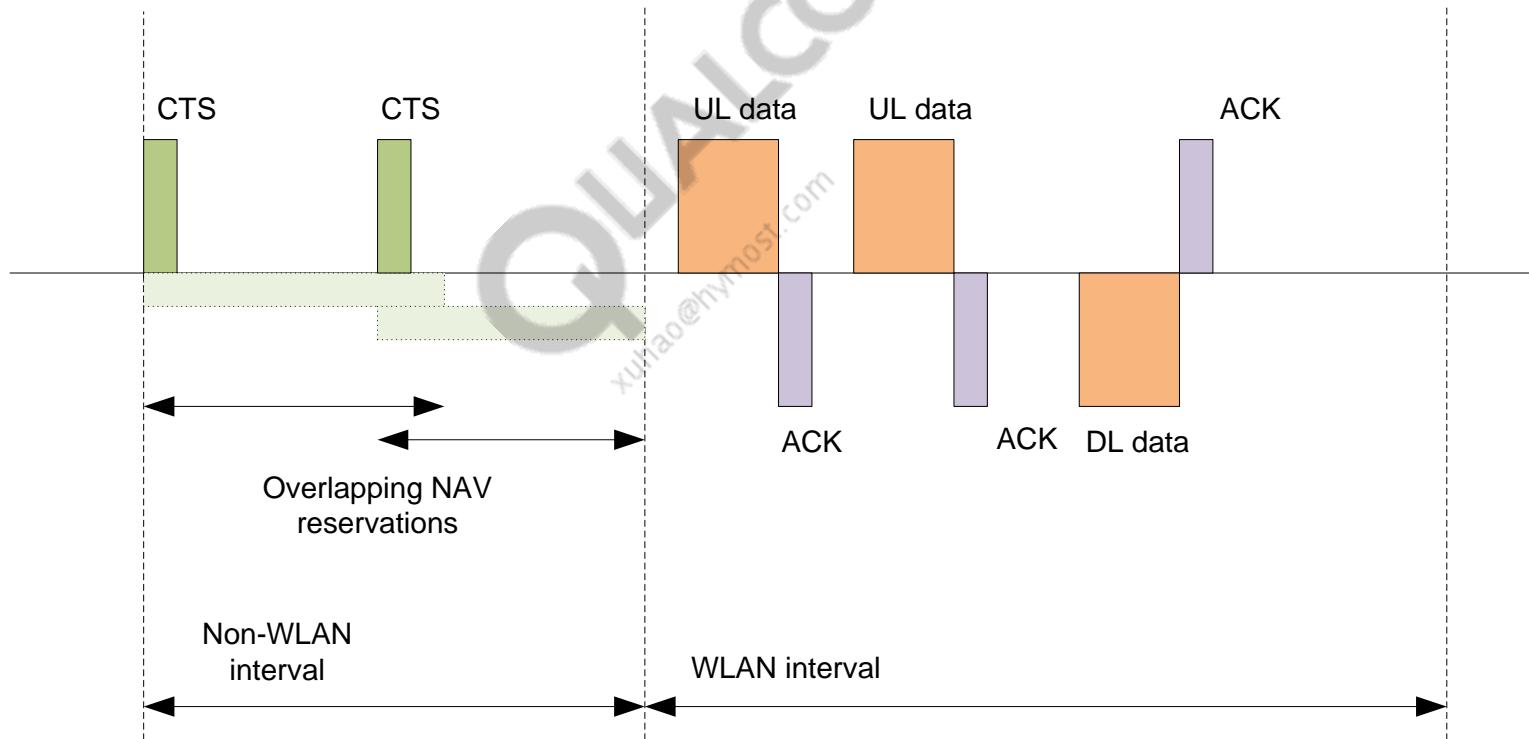
## PM State with PS-POLLS

- Advantages
  - Once a PM state is entered, it is “sticky”
    - ◆ Does not require any additional transmissions to maintain
  - No mode switch overhead
- Disadvantages
  - Per downlink packet overhead
    - ◆ Every downlink packet requires a PS-POLL/ACK pair preceding it
  - Response time to PS-POLLS is highly IOT and environment-dependent
    - ◆ PS-POLL/ACK and subsequent DL DATA/ACK are not the same TxOP; other traffic can get between them
    - ◆ AP/P2P GO response time once a PS-POLL is ACK’ed is not specified by the standard; varies from peer to peer

# BTC Concurrency – Time Slicing (cont.)

## CTS2SELF

- Send CTS frames with our own address to reserve the NAV



## CTS2SELF

### ■ Advantages

- Broadcast flow control
  - ◆ All receivers are controlled

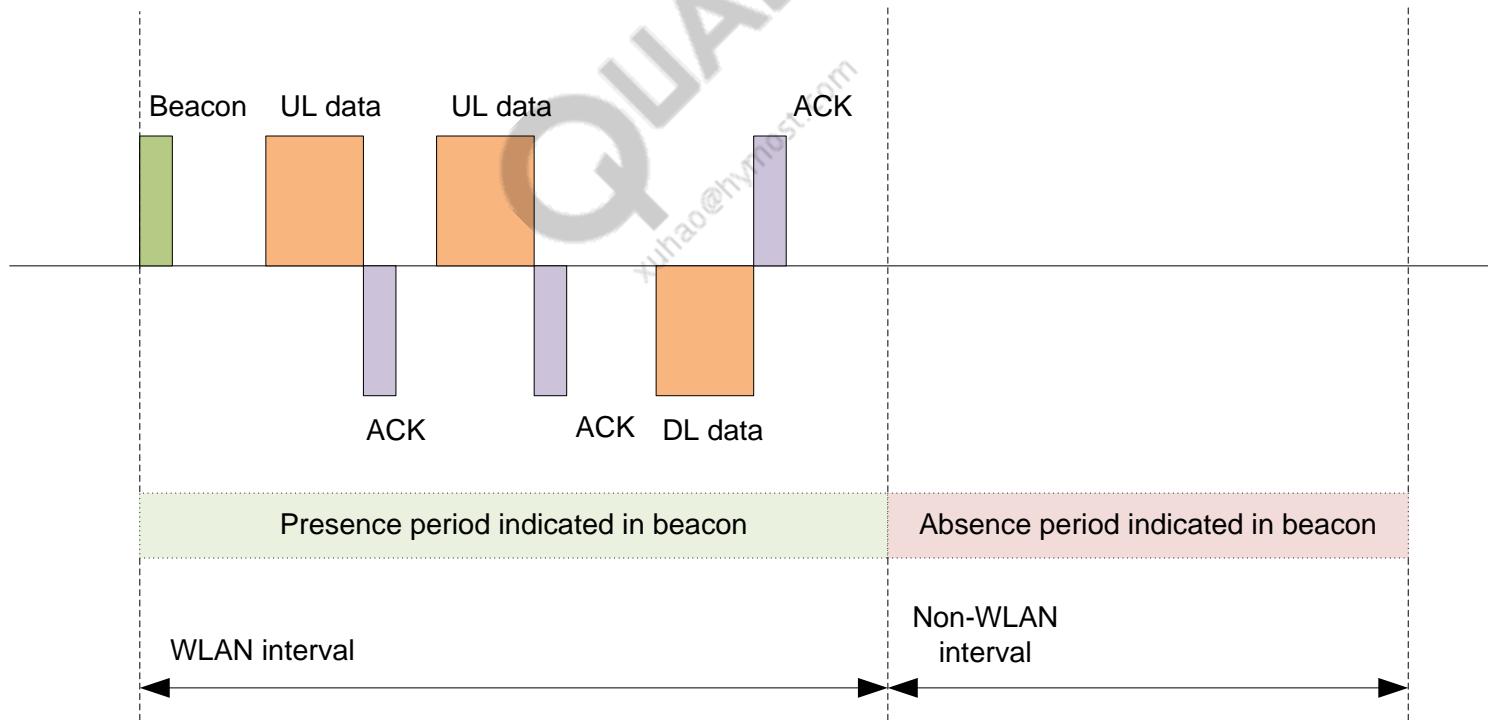
### ■ Disadvantages

- Not all intended stations may see it
  - ◆ Not an ACK'ed frame, so no reliability
  - ◆ Intended peer may be power-collapsed or hidden
- Unintended stations may see it
  - ◆ Any receivers on channel may see it, even ones not in your BSS
  - ◆ These peers are also controlled
- Short maximum duration
  - ◆ Can only cover up to ~30 msec before another must be sent
  - ◆ Can require transmissions during the “non-WLAN” interval

# BTC Concurrency – Time Slicing (cont.)

## Notice Of Absence (NOA)

- P2P GO only
- Advertise times when P2P GO is present (and absent), typically via beacons



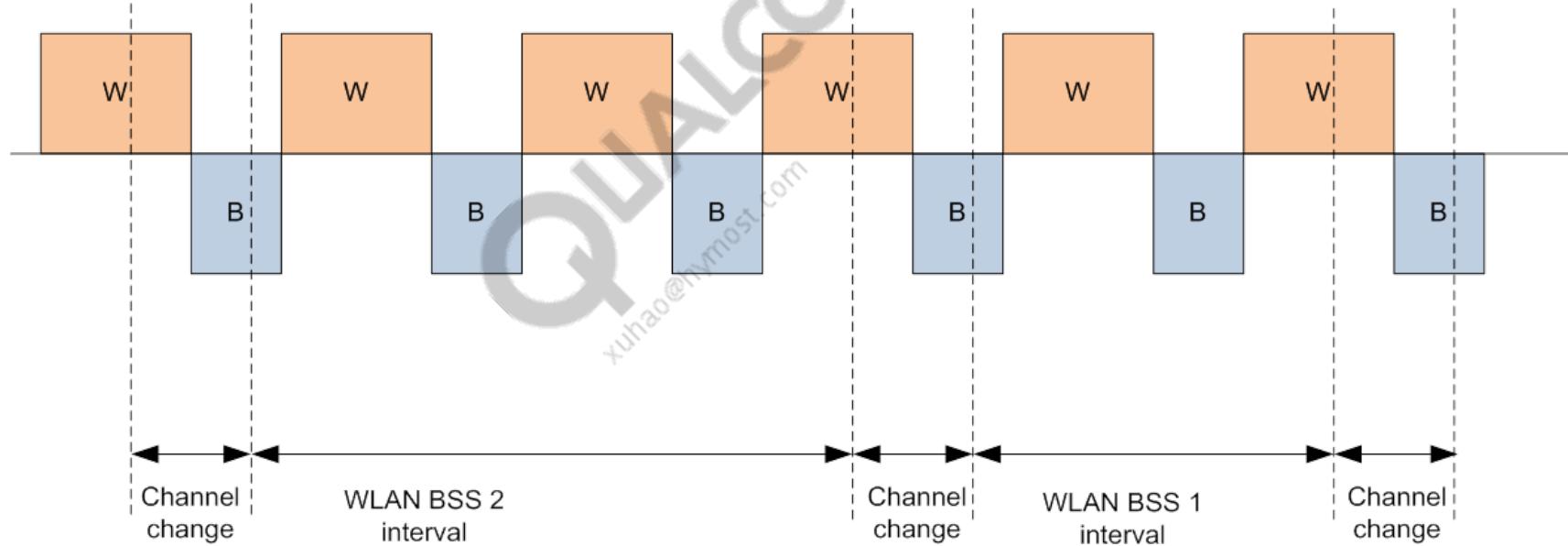
## Notice Of Absence (NOA)

- Advantages
  - No overhead
  - Flow controls only and all intended peers
- Disadvantages
  - Publication time is slow
  - Cannot react quickly to dynamically changing scheduling needs

# BTC Concurrency – Contextual Behavior Differences in BTC Modes

- Within modes, the moment-to-moment behavior may be changed to accommodate transient link needs
  - BT ACL sniff window
  - BT inquiry and paging scans
  - WLAN DHCP resolution
  - WLAN scans
- MCC/CoEX overlay modes are driven via this model
  - MCC state provides Coexistence context to drive appropriate behavior

## BTC Concurrency – Mode Adaptation, MCC and CoEX Schedule Overlay



# BTC Concurrency – Mode Adaptation, Dynamic Short {MCC}

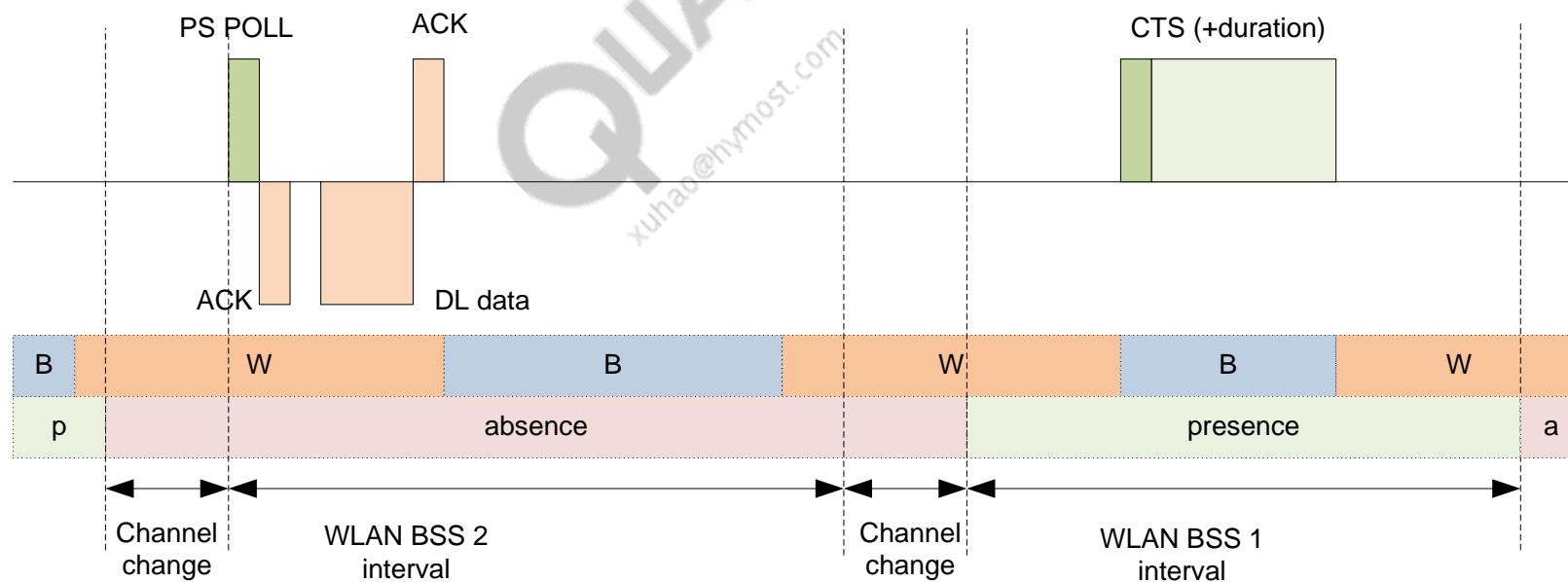
- Dynamic short {MCC} is used for SCO
- In Dynamic short {MCC}, the MCC scheduler and CoEX scheduler run independently, as before, with no recognition of one another
- BSSs are flow-controlled by MCC as usual
  - PM state with PM state transitions is used for {STA + P2P Client} cases
    - ◆ {PM = 0} transmissions on 2.4 GHz BSSs from MCC are suppressed
  - PM state with PM state transitions + NOA is used for {STA + P2P GO} cases
- The behavior on air is driven by the overlay of the two resultant schedules
  - MCC state provides a context to CoEX

- Contextual adaptation

- In {W + 2.4 GHz STA/P2P client BSS}, PM state with PS-POLLs
- In {W + 2.4 GHz P2P GO BSS}, CTS2SELF frames are used
  - ◆ Periodic NOA is already used to control BSS 1 vs. BSS 2 times
  - ◆ Only one periodic NOA is supported
- In {W + 5 GHz BSS}, WLAN is not controlled by CoEX
  - ◆ MCC polices determine WLAN behavior
- In {W + channel change}, WLAN is not controlled by CoEX
  - ◆ MCC policies determine WLAN behavior

# BTC Concurrency – Mode Adaptation, Dynamic Short {MCC} (cont.)

- Timelines are not to scale
- CTS is sent when P2P GO is in 2.4 GHz; otherwise, no action is needed



# BTC Concurrency – Mode Adaptation, Dynamic Long {MCC}

- Dynamic long {MCC} is used
  - In cases where Dynamic long or Static long may have been used
  - When both WLAN BSSs are in 2.4 GHz
  - In STA + P2P GO scenarios
- In Dynamic long {MCC}, the MCC scheduler and CoEX scheduler run independently, as before, with no recognition of one another
- BSSs are flow-controlled by MCC as usual
  - PM state with PM state transitions + NOA is used
  - {PM = 0} transmissions on 2.4 GHz BSS from MCC are suppressed
- The behavior on air is driven by the overlay of the two resultant schedules
  - MCC state provides a context to CoEX

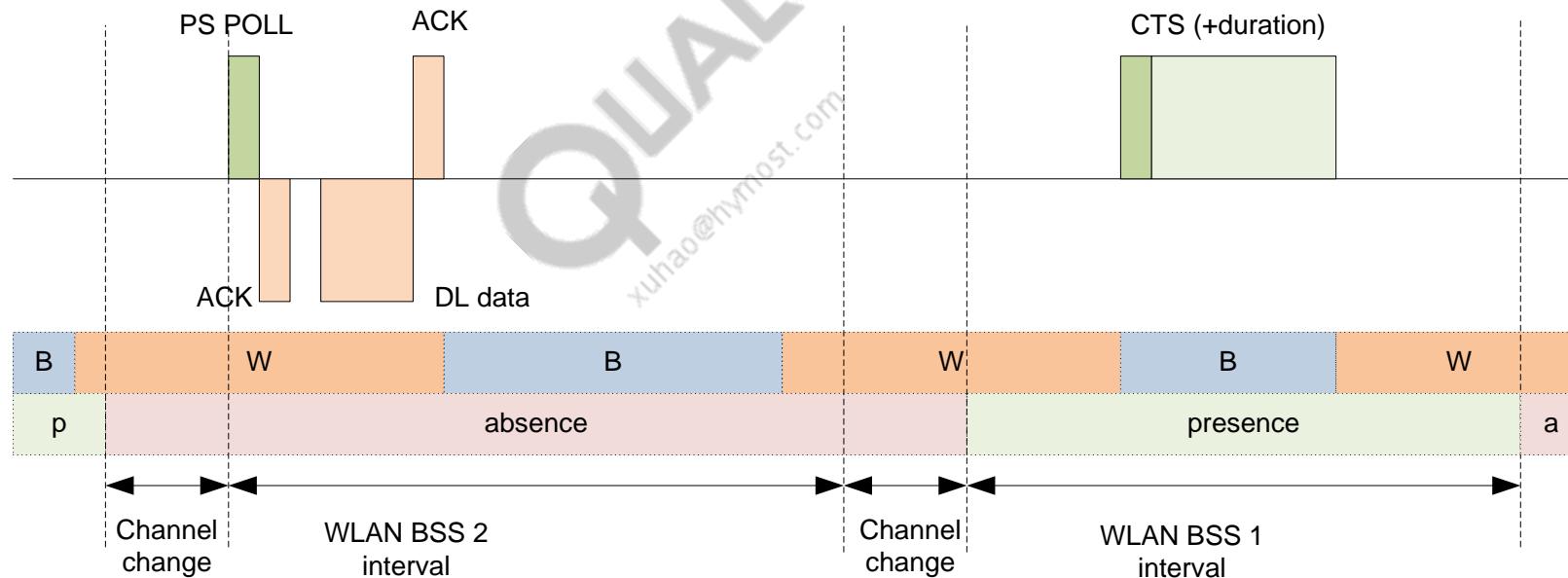
# BTC Concurrency – Mode Adaptation, Dynamic Long {MCC} (cont.)

## ■ Contextual adaptation

- In {W + 2.4 GHz STA BSS}, PM state with PS-POLLs
- In {W + 2.4 GHz P2P GO BSS}, CTS2SELF frames are used
  - ◆ Periodic NOA is already used to control BSS 1 vs. BSS 2 times
  - ◆ Only one periodic NOA is supported
- In {W + channel change}, WLAN is not controlled by CoEX
  - ◆ MCC policies determine WLAN behavior

# BTC Concurrency – Mode Adaptation, Dynamic Long {MCC} (cont.)

- Timelines are not to scale

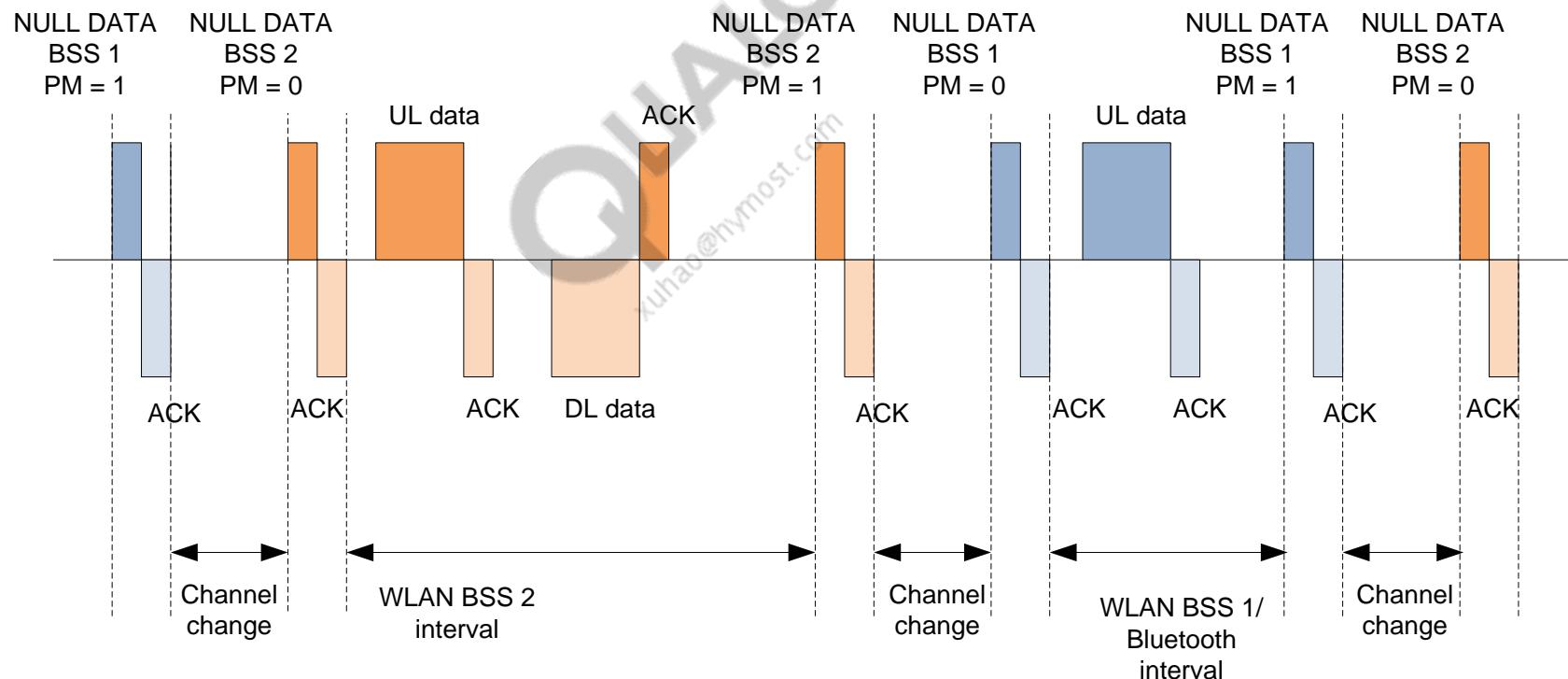


# BTC Concurrency – New Mode, MCC Long {2.4 + 5}

- MCC long {2.4 + 5} is used
  - In cases where Dynamic long or Static long may have been used
  - When one WLAN BSS is in 2.4 GHz and one is in 5.0 GHz
- In MCC long {2.4 + 5}, the MCC scheduler is driving the intervals with guidance from the CoEX scheduler
  - CoEX scheduler gives basic interval length guidance, MCC scheduler controls exact interval length and phase
  - CoEX interval guidance is slower than used in Dynamic long mode to allow for overhead
    - ◆ NOA publication time
    - ◆ Interface overhead with MCC scheduling
  - CoEX guidance is driven by BT QoS needs as before
  - MCC exact interval length and phase are driven by beacon protection

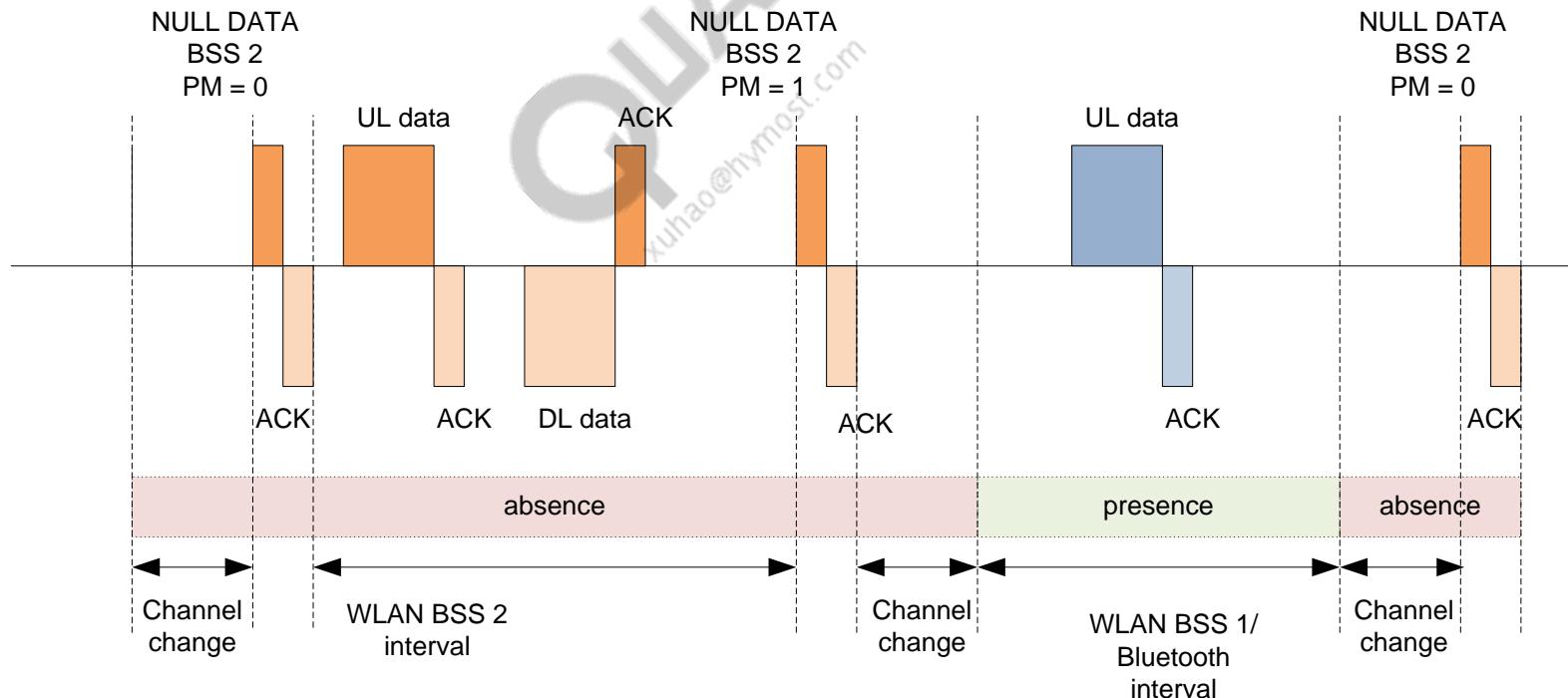
# BTC Concurrency – MCC Long {2.4 + 5}, STA + P2P Client

- When both WLAN BSSs are client-like, we use PM state with PM transitions
- 5 GHz BSS is fully concurrent with Bluetooth



# BTC Concurrency – MCC Long {2.4 + 5}, STA + P2P GO

- When one WLAN BSS is P2P GO, we use NOA and PM state with PM state transitions
- 5 GHz BSS is fully concurrent with Bluetooth

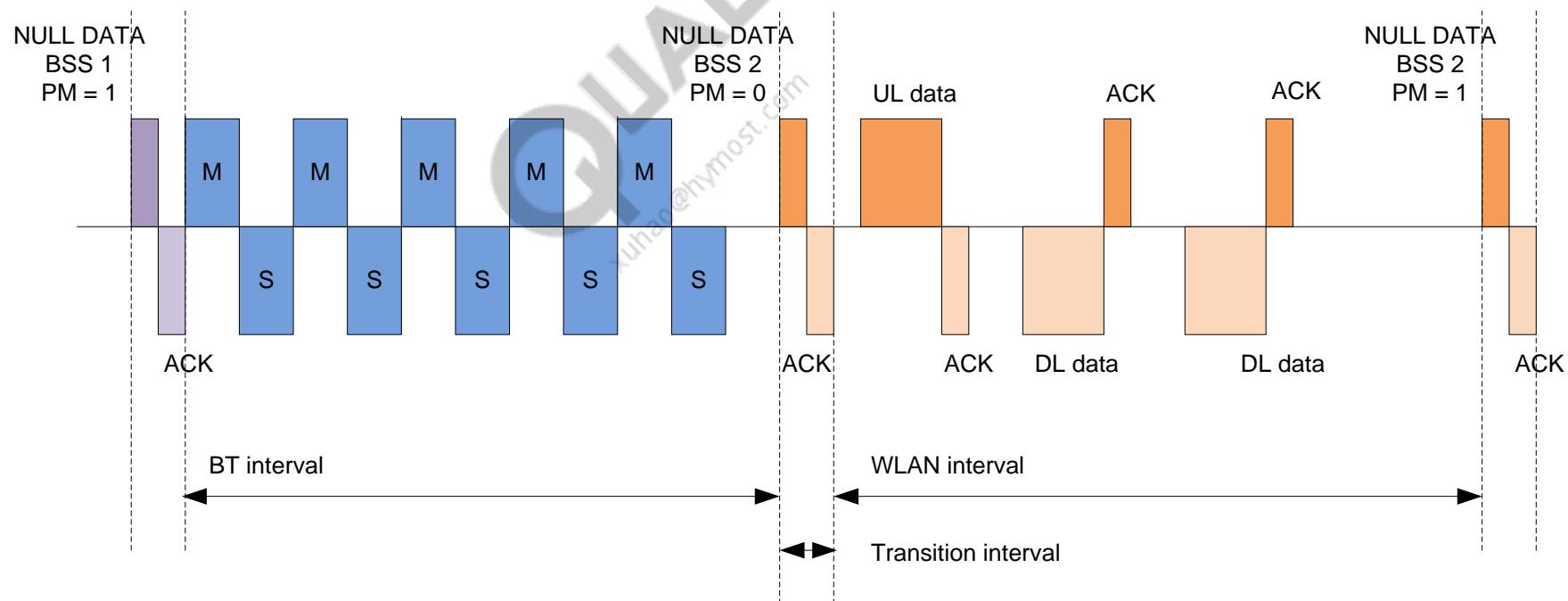


# BTC Concurrency – New Mode, MCC Long {2.4 + 2.4}

- MCC long {2.4 + 2.4} is used
  - In cases where Dynamic long or Static long may have been used
  - When both WLAN BSSs are in 2.4 GHz
  - In STA + P2P Client scenarios
- In MCC long {2.4 + 2.4}, the CoEX scheduler is driving the intervals with no guidance from the MCC scheduler
  - CoEX scheduling is driven by BT QoS needs as before
    - ◆ WLAN intervals are given to alternating BSSs
    - ◆ Throughput may be shifted from one BSS to the other by varying the alternation schedule
    - ◆ Identical responsiveness (to BT QoS needs) to non-concurrency case
  - No explicit beacon protection is given
    - ◆ Experience with non-concurrency case implies this is not needed in most cases

# BTC Concurrency – New Mode, MCC Long {2.4 + 2.4} (cont.)

- Entire WLAN intervals are given to one BSS
- BSS schedule may be varied to yield non-symmetric servicing

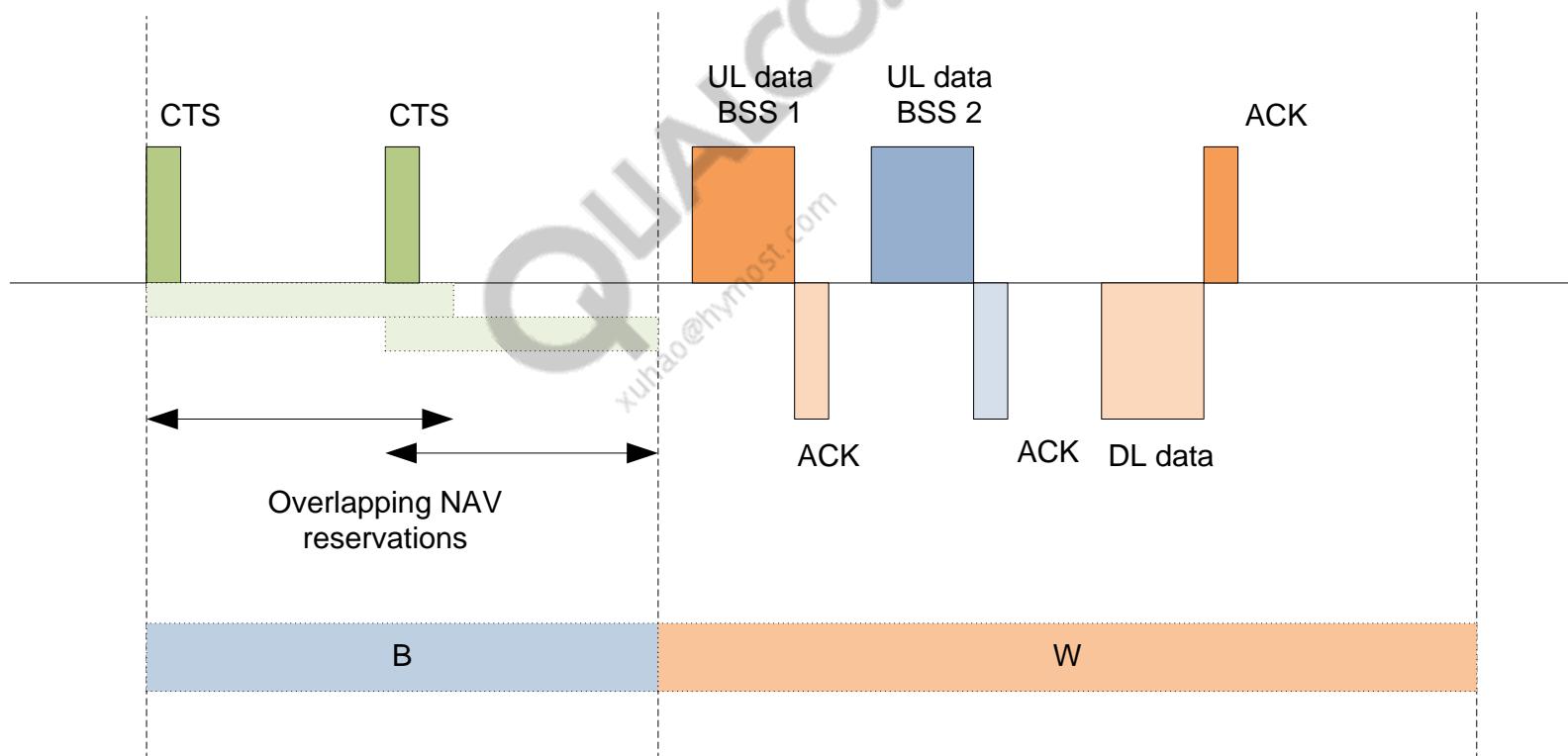


# BTC Concurrency – Special Topic, Single Channel Concurrency

- Single channel concurrency has no concurrency scheduler, as there are no BSSs to switch between
- CoEX in single channel concurrency is simply mapped to processing we perform for the Soft AP (SAP) case
  - SAP CoEX in non-concurrency cases uses the same scheduling as mentioned above (dynamic long, static long, dynamic short), but uses CTS2SELF to flow control the peer rather than PM State methods
  - CTS2SELF will flow control both BSSs with the same frame

# BTC Concurrency – Special Topic, Single Channel Concurrency (cont.)

- BSS 1 and BSS 2 traffic is intermingled



# Bluetooth/WLAN Coexistence Logging

- CxM logging can be done by QXDM
  - Since CxM is tightly working with BT and WLAN, all the BT and WLAN logging should be enabled
  - “WLAN reserved 10” is actual option to turn on to get CxM logs

## WiFi Direct (P2P)

REDEFINING MOBILITY

## Why Wi-Fi Direct?

- Connect devices directly in a new kind of Wi-Fi network without joining a traditional home, office, or hotspot network at existing Wi-Fi speeds and also be able to broadcast their availability
- Enable the same device connectivity as Bluetooth®, but at ranges and speeds equivalent to what users experience with existing Wi-Fi connections.
- Support for WPA2 (Wi-Fi Protected Access 2) and AES encryption
- Support in any Wi-Fi device, from mobile phones, cameras, printers, and notebook computers to human interface devices such as keyboards and headphones.
- A Wi-Fi Direct-enabled mobile phone could establish a connection with a non-Wi-Fi Direct notebook computer to transfer files between the two

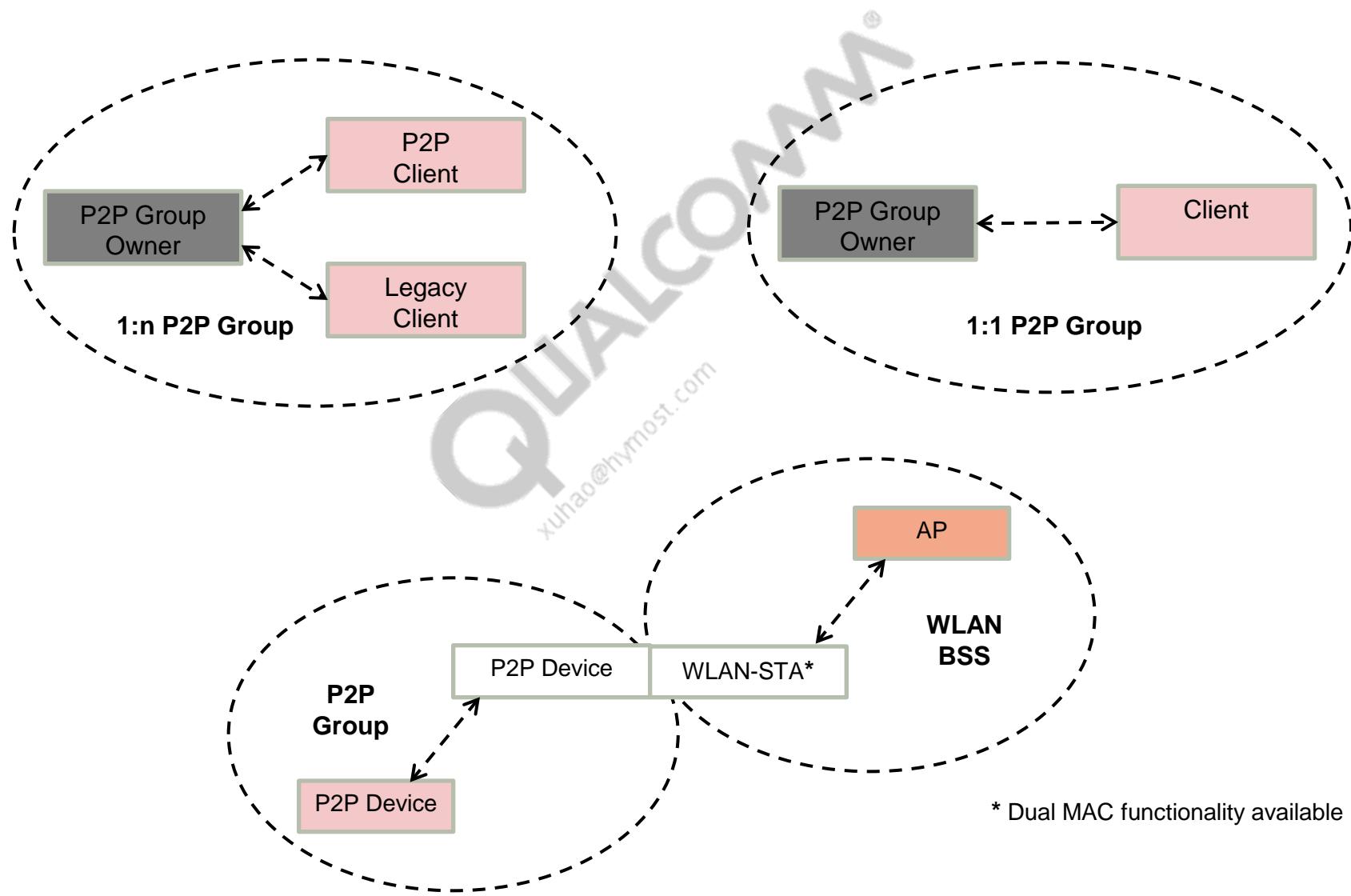
# Introduction

- Wi-Fi Direct defines an architecture and set of protocols that facilitate:
  - Wi-Fi device-to-device connectivity
  - Backward compatible with existing Wi-Fi CERTIFIED devices
- Scope
  - Discovery (Device and Service)
  - Pairing (Grouping and Invitation)
  - Connectivity
  - Power Management
  - Group Management
  - Coexistence
  - Legacy

# P2P Components

- P2P device
  - Supports both P2P Group Owner and P2P Client Role
- P2P Group Owner Role
  - AP-like functionality
  - Wi-Fi Protected Setup (WPS) Internal Registrar Role
  - Communication between clients
  - Concurrency
- P2P Client Role
  - Non-AP STA functionality
  - WPS enrollee functionality

# P2P Topology



REDEFINING MOBILITY

## P2P Discovery

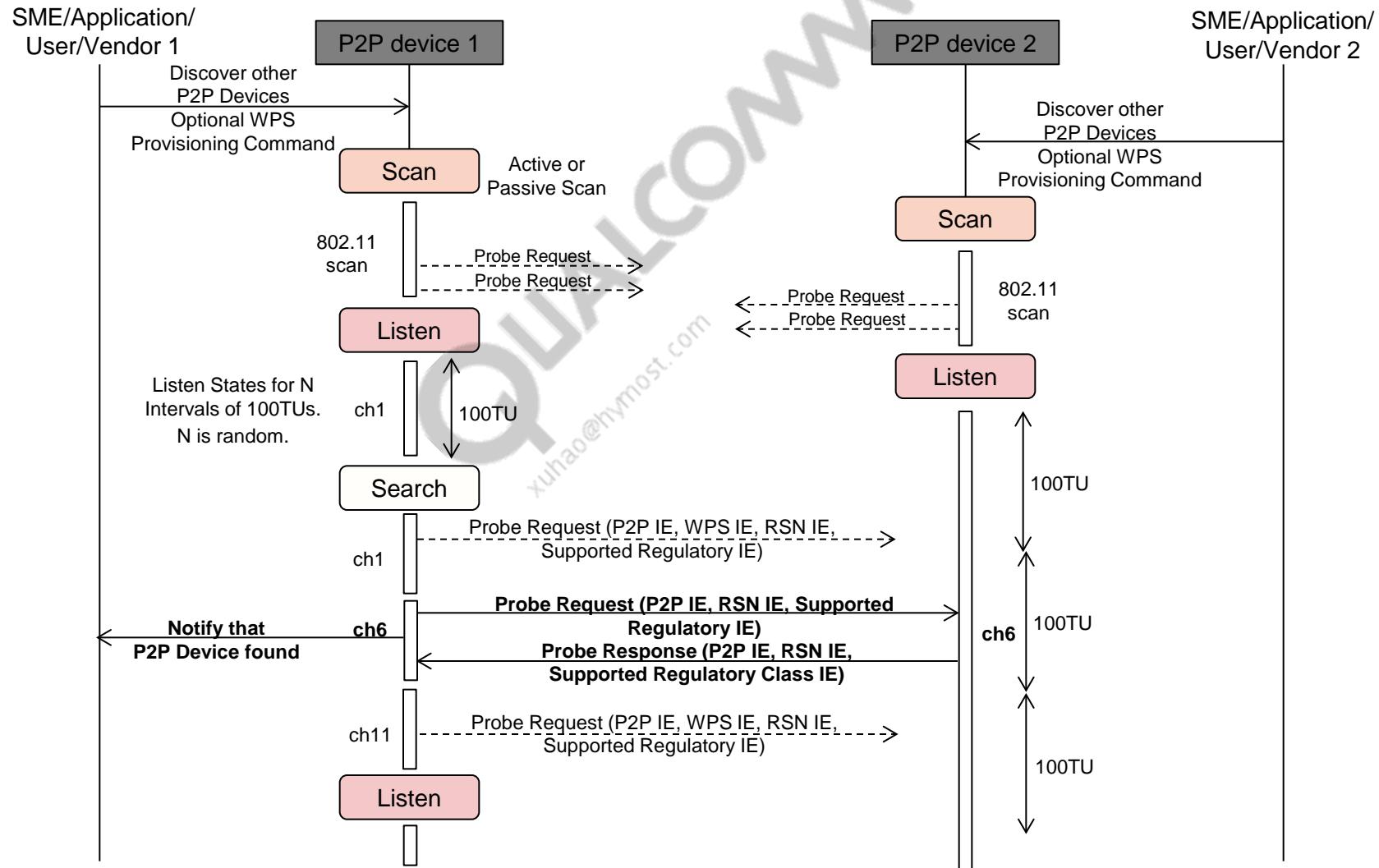
# Device Discovery

- Scan phase
  - Similar to existing Scan Procedure (active or passive scan)
  - Scan all supported channels (802.11 scan)
  - Discover operational P2P Groups and legacy networks
- Find phase
  - Listen state – Responds to Probe Request from other P2P devices
  - Search state – Scans for other P2P devices by sending Probe Request
  - Listen and Search on social channels 1, 6, and 11
- P2P devices on the same channel in Discovery state can discover each other

# Cases for Device Discovery

- Two P2P devices in Discovery
  - Discover each other during the Find phase on social channels
- Discover P2P Client in P2P Group
  - GO discovery first during the Scan phase; exchange information using Probe Response from GO
  - Uses device Discoverability Request/Response frames to get information about the P2P Client connected and then invite the P2P Client to form a group
- Discover a P2P Group Owner during the Scan Phase
- Discover a P2P device associated with an infrastructure AP
  - A P2P device that is associated with an infrastructure AP shall still need to be in the Listen state for device discovery
- Discover a Legacy Client
  - Legacy client can discover only P2P Group Owner
  - P2P Group Owner does not respond to 11b rates only

# Example for Two P2P Devices in Discovery on ch 6



# P2P Invitation Procedure

- The P2P optional invitation procedure is for:
  - A P2P group owner inviting a P2P device to become a P2P client in its P2P group
  - A P2P client inviting another P2P device to join the P2P group of which the P2P client is a member, because it wishes to use a service of the P2P device
  - Requesting to invoke a persistent P2P group for which both P2P devices have previously been provisioned
- P2P invitation request and invitation response frame exchange is carried for this procedure

# Service Discovery

- Optional frame exchange that may be performed at any time to any discovered P2P device
- Enables different higher layer service advertisement protocol types such as Bonjour and UPnP
- Leverages the Generic Advertisement Service (GAS) protocol/frame exchange as defined in IEEE P802.11u
- It can be used to find:
  - A list of all services offered by P2P device
  - Information about a single service offered by P2P device
  - Information about multiple services offered by a P2P device
  - Any change in the services offered by a P2P device

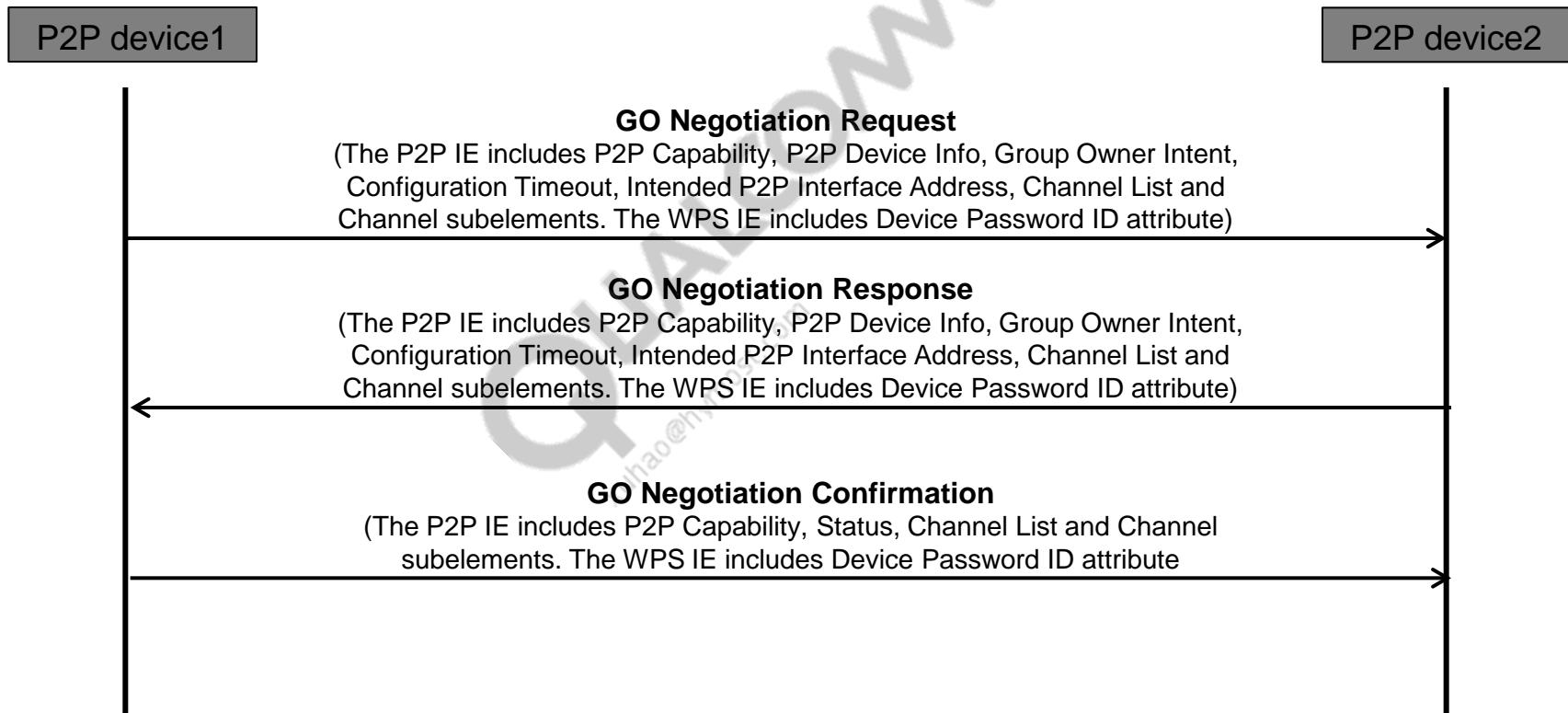
## P2P Group Formation

REDEFINING MOBILITY

# Group Formation

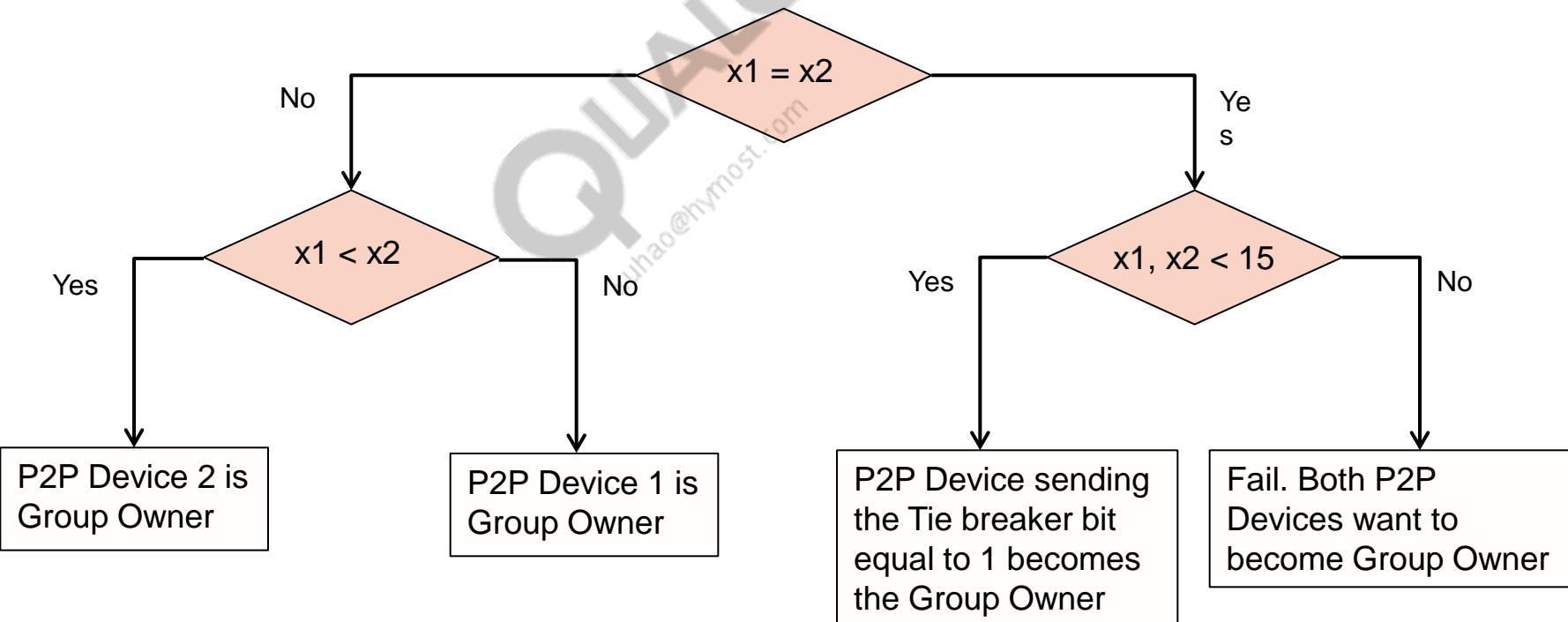
- Form a new P2P Group instead of autonomously starting a P2P group by becoming a P2P Group Owner
- Consists of Group Owner Negotiation and Provisioning
  - Group Owner Negotiation to determine which device shall be P2P Group Owner via Group Owner Intent Algorithm
  - Provisioning to exchange the credentials
    - ◆ Use WPS
    - ◆ May use Provision discovery frames to obtain the required information, e.g., PIN

# Group Owner Negotiation



# Group Owner Intent Algorithm

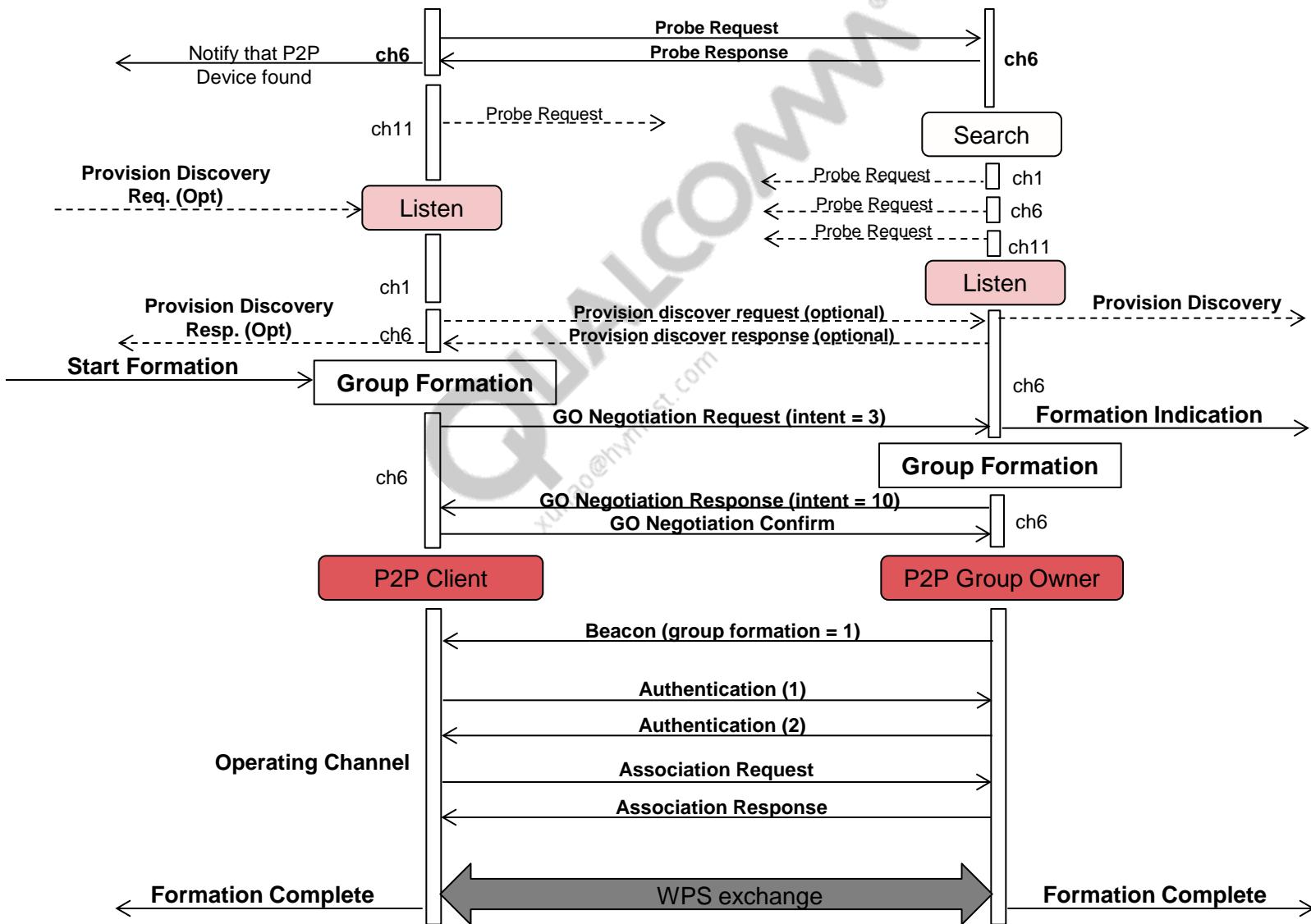
$x_1$  = Group Owner Intent Value of P2P Device 1  
 $x_2$  = Group Owner Intent Value of P2P Device 2



# Provisioning

- Described in Wi-Fi Protected Setup (WPS) with the following modifications
  - P2P Group Owner shall serve the role as the AP with Internal Registrar
    - ◆ Shall only allow association by the P2P device with which it is currently in Group Formation
    - ◆ Registrar shall send M2 in response to M1 and shall not send M2D
  - P2P Client shall serve the role as the STA Enrollee
    - ◆ Shall associate to the P2P device with which it is currently in Group Formation
- If failed, Group Formation ends and group session is terminated by the GO
- Group formation bit in P2P capability sub IE shall be set to 1 until provisioning succeeds

# Scenario for Group Formation



# Example of Group Formation – from Devices Discovery

- User discovers P2P devices in the range
- User selects which of the discovered P2P devices to connect to and enters Provisioning information, e.g., WPS PIN
- Negotiate which device shall become GO (AP-like entity)
  - Most appropriate device, e.g., TV
- Use authentication in WPS to ensure that these are correct devices
  - TV supplies SSID and WPA2 PSK using WPS
- GO starts P2P Group session and Client joins Group



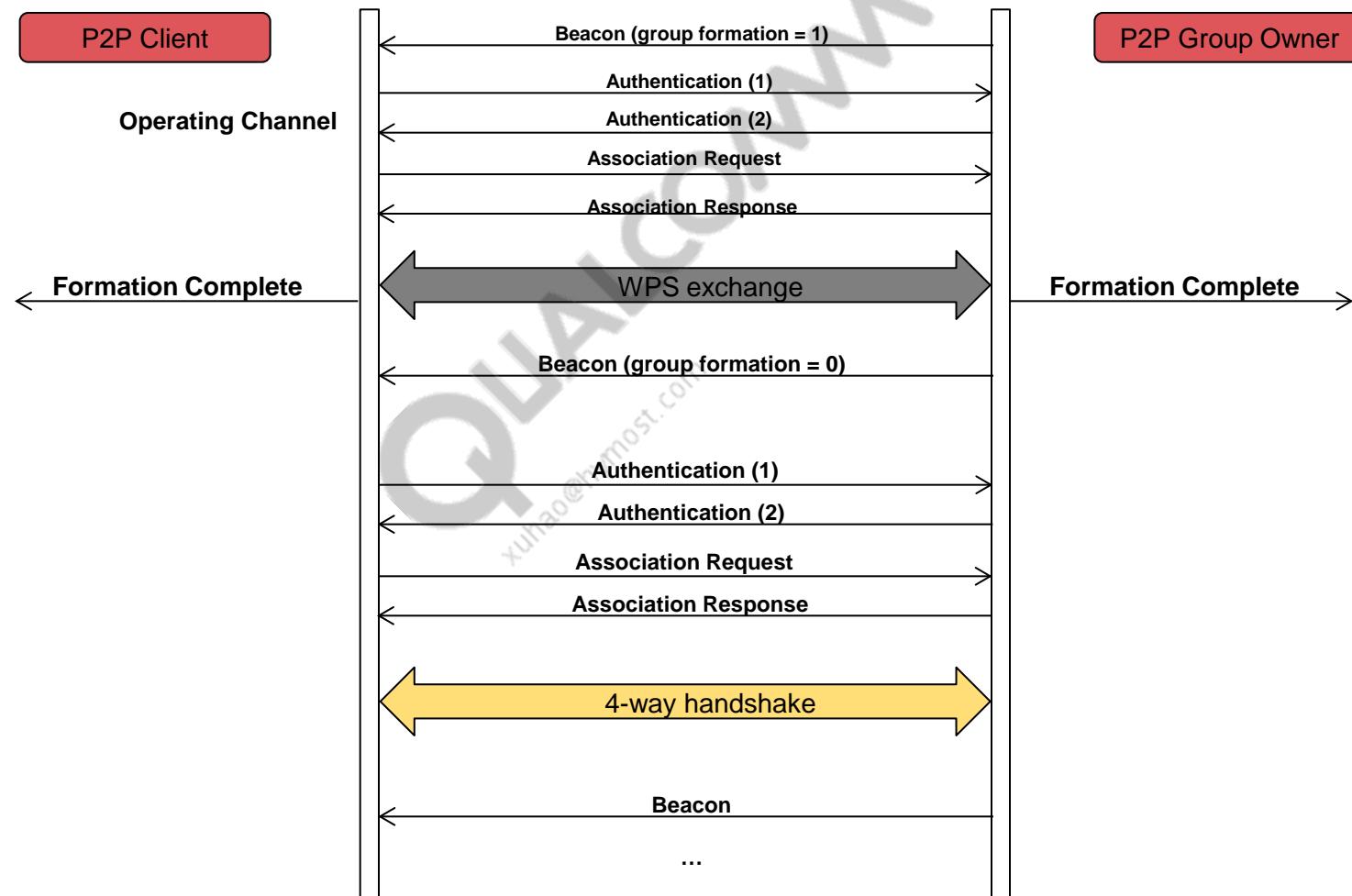
## P2P Group Operation

REDEFINING MOBILITY

# P2P Group Operation

- Starting and maintaining a P2P Group session
  - Similar to infrastructure BSS operation
    - ◆ P2P Group Owner assuming the role of the AP
    - ◆ P2P Client assuming the role of the STA
  - Credentials for a P2P Group issued to a P2P device shall use:
    - ◆ WPA2-PSK as authentication type, Pass phrase maintained by Group Owner
    - ◆ AES as encryption type, A Network Key Type of 64 Hex characters
    - ◆ SSID for each group to assure that all P2P Groups are unique
      - ▶ DIRECT-xy, xy is generated randomly from upper/lowercase letters and number
  - Group Owner shall set SSID and select the operating channel
- Persistent Group operation
  - After successful Invitation exchange, GO of a persistent group shall restart the Persistent P2P Group (BSSID, operating channel may not be the same)
- Connecting to a P2P Group
  - Client acquires the Group Credentials through static configuration or WSC
- Disconnecting from a P2P Group

# Scenario for Connecting to P2P Group



# Communication in a Group

- Use WPA2-PSK security with AES-CCMP as encryption cipher
- Higher layers use IP
  - P2P GO (acting as a DHCP server) shall provide IP address to connected P2P Clients (acting as DHCP client) that use IP
- P2P GO provides data distribution service to all clients within the group

# Disconnection

- P2P Client can disconnect from a P2P group anytime using:
  - Deauthentication frame to P2P GO
  - Disassociation frame to P2P GO
  - P2P GO does not depend on any indication from the P2P Client
- P2P GO may disconnect any P2P Client from a P2P group using:
  - Deauthentication frame to P2P client
  - Disassociation frame to P2P client
  - End the group session anytime it desires; by sending a broadcast deauthentication frame to all clients

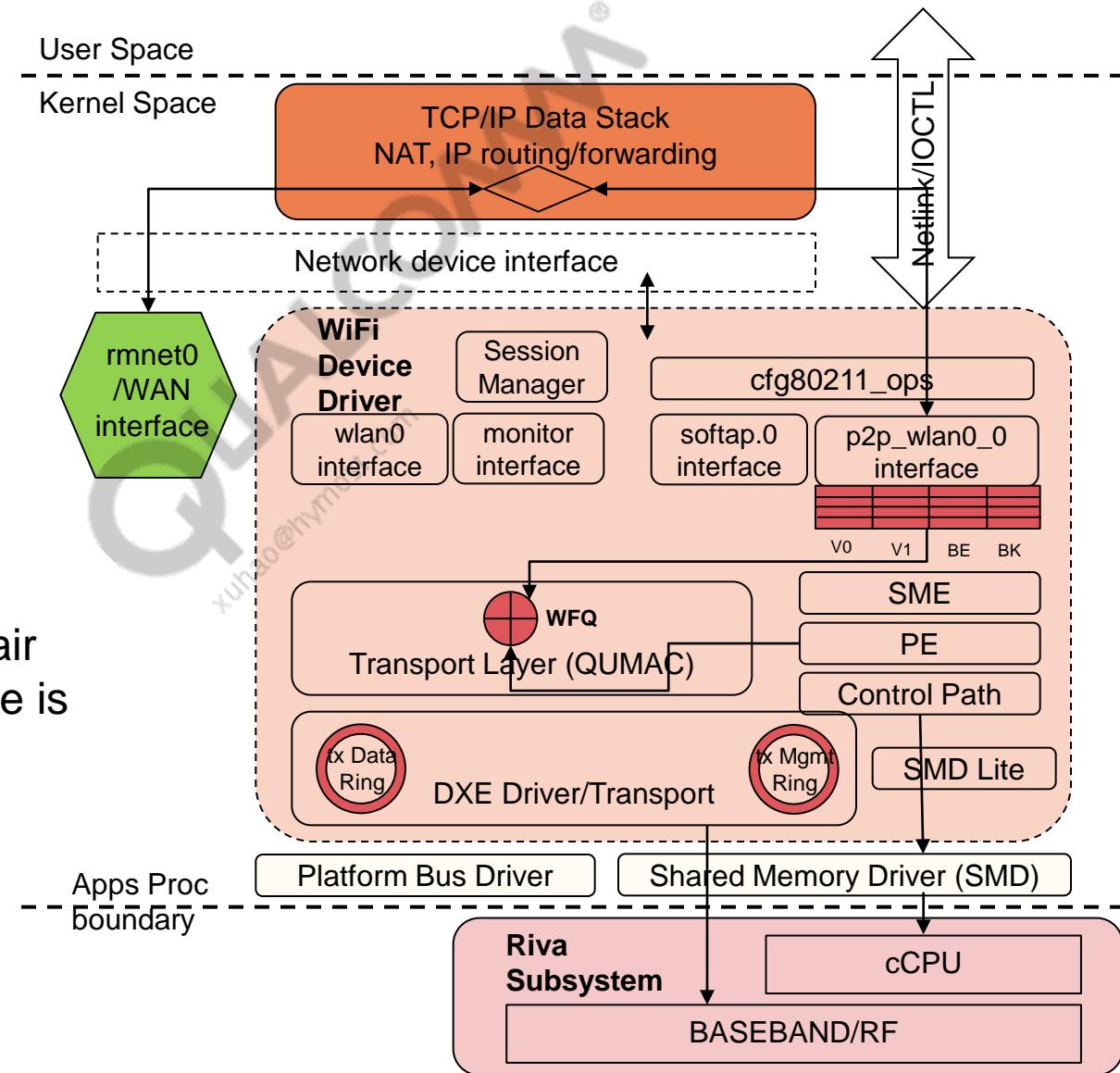
## P2P Software Architecture

REDEFINING MOBILITY

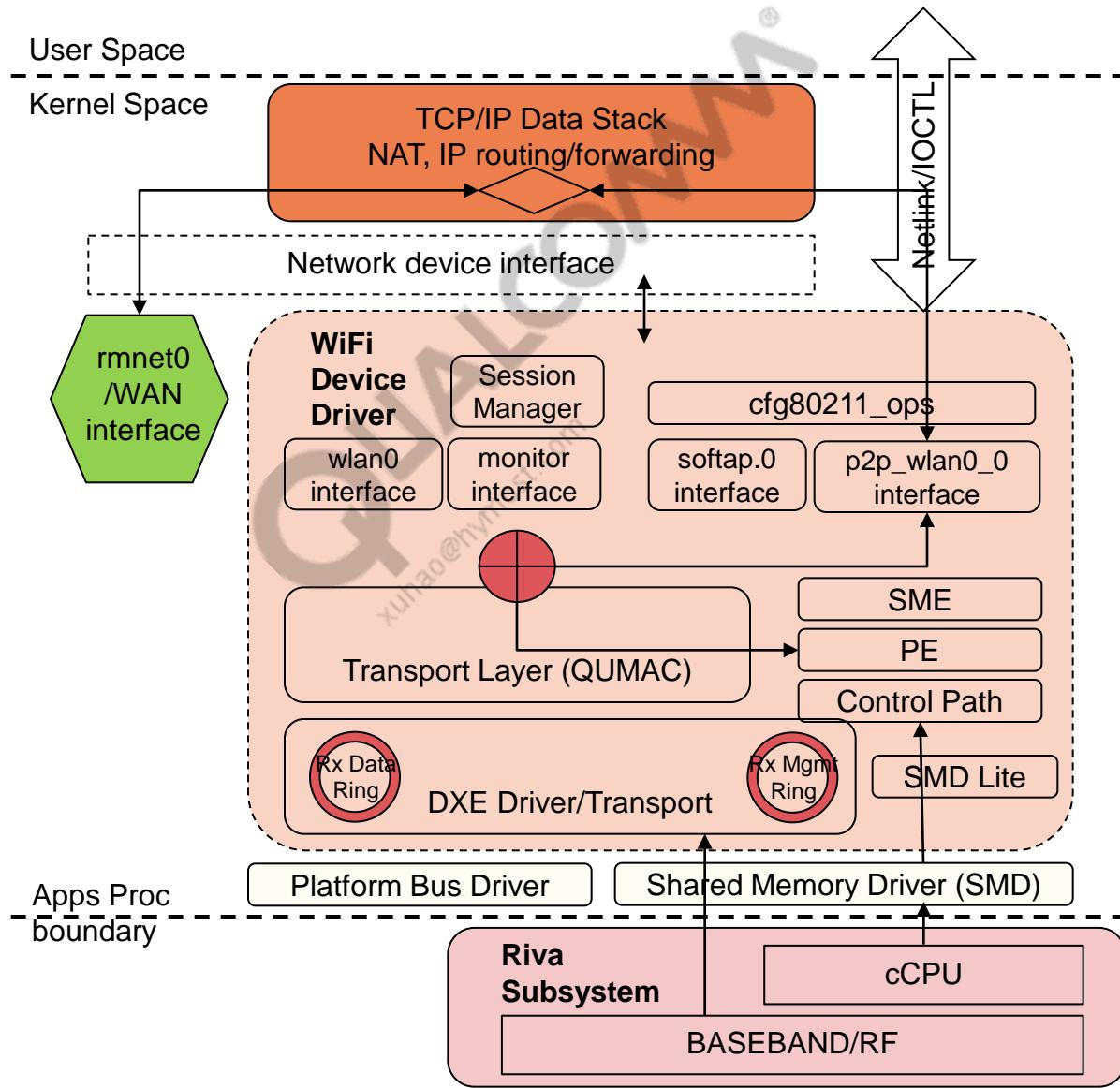
# P2P Software Architecture

- WLAN Core Stack, comprising TL, SME, PE, DXE, etc., are involved in the Data path
- TL is responsible for handing off packets between data vs. management between various “sessions”
- All testing limited to the CLI
  - OSS CLI tools
  - iw, wpa\_cli, iperf, etc., for STA Client

# P2P Tx Flow



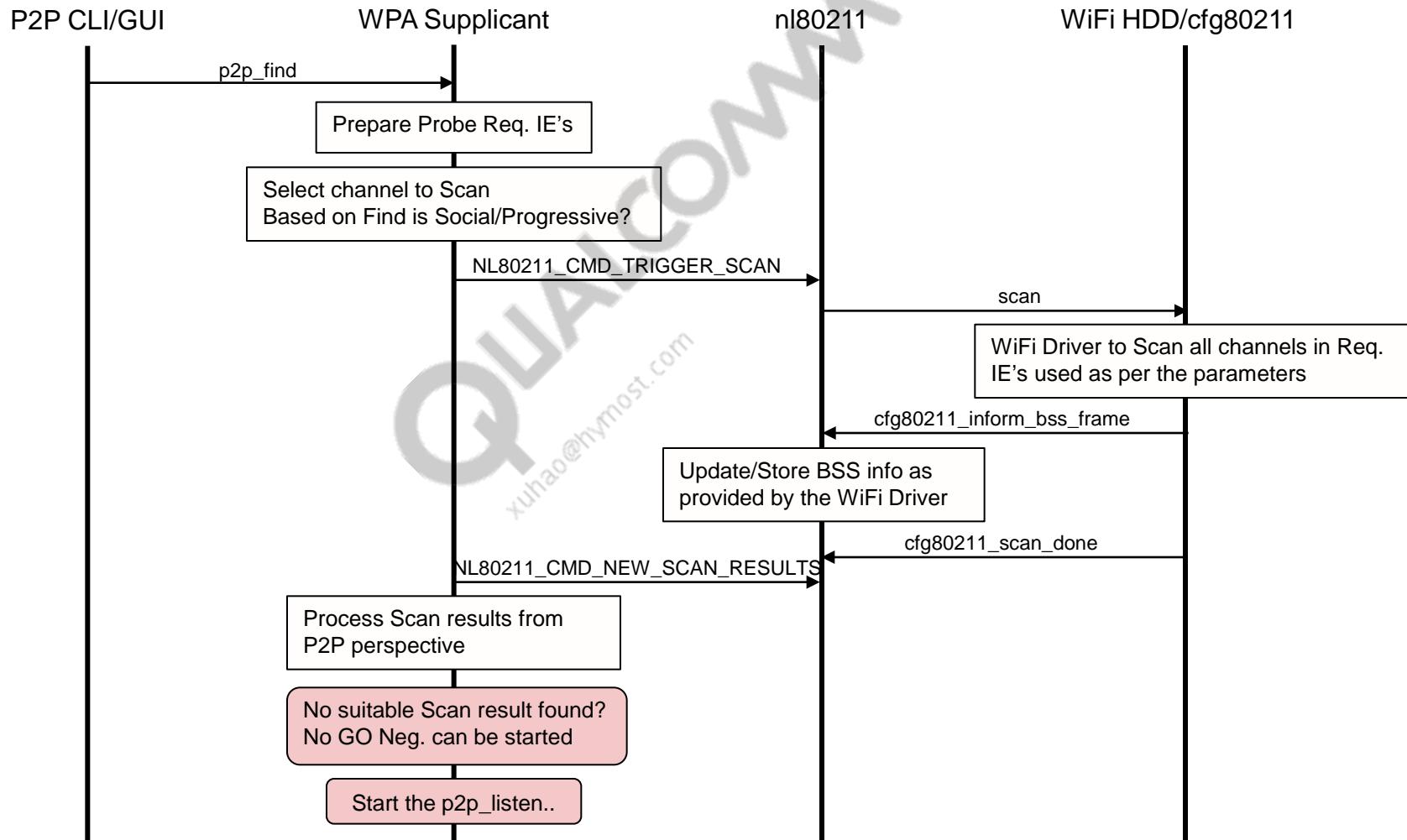
# P2P Rx Flow



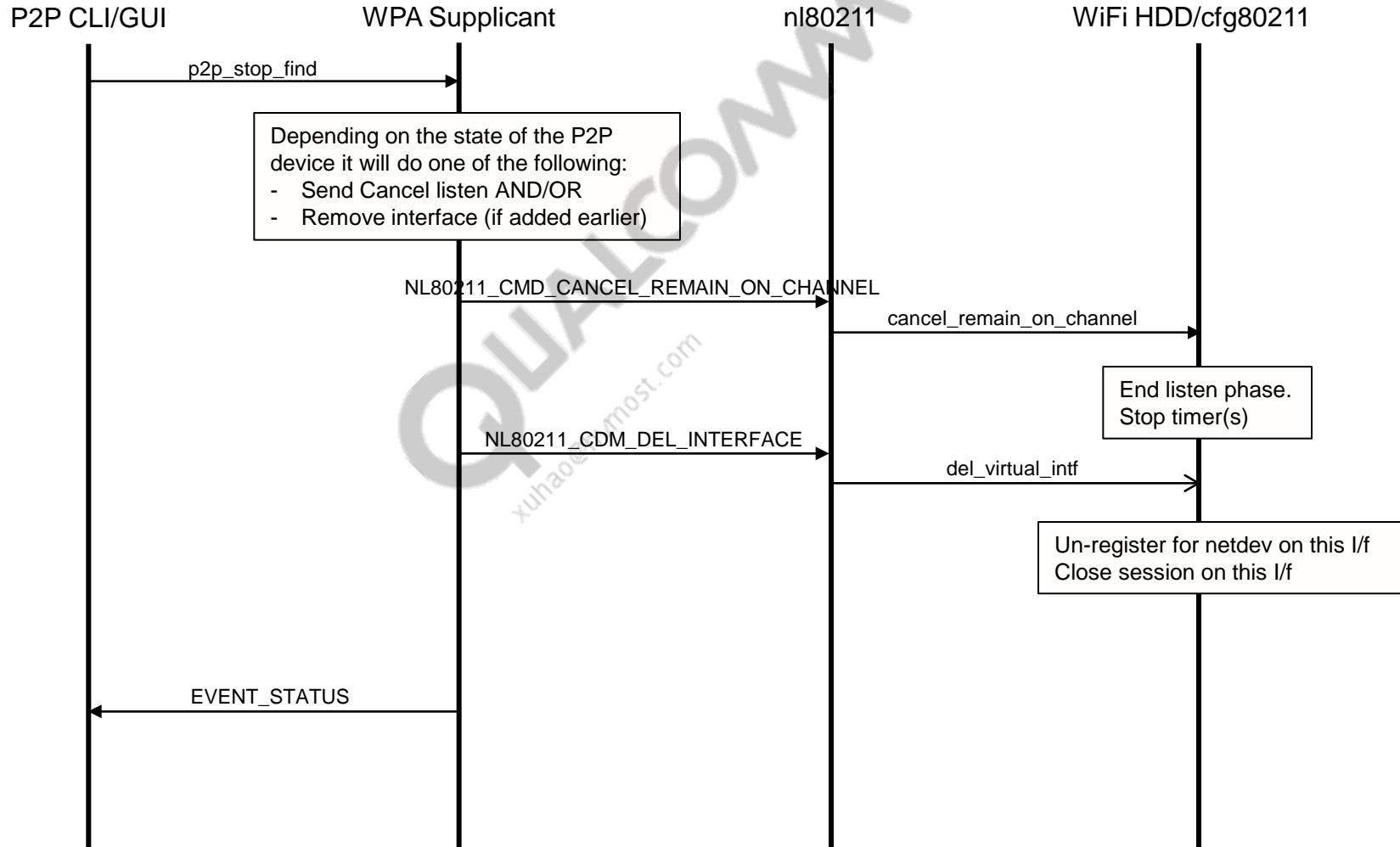
## P2P Event/Message Sequence

REDEFINING MOBILITY

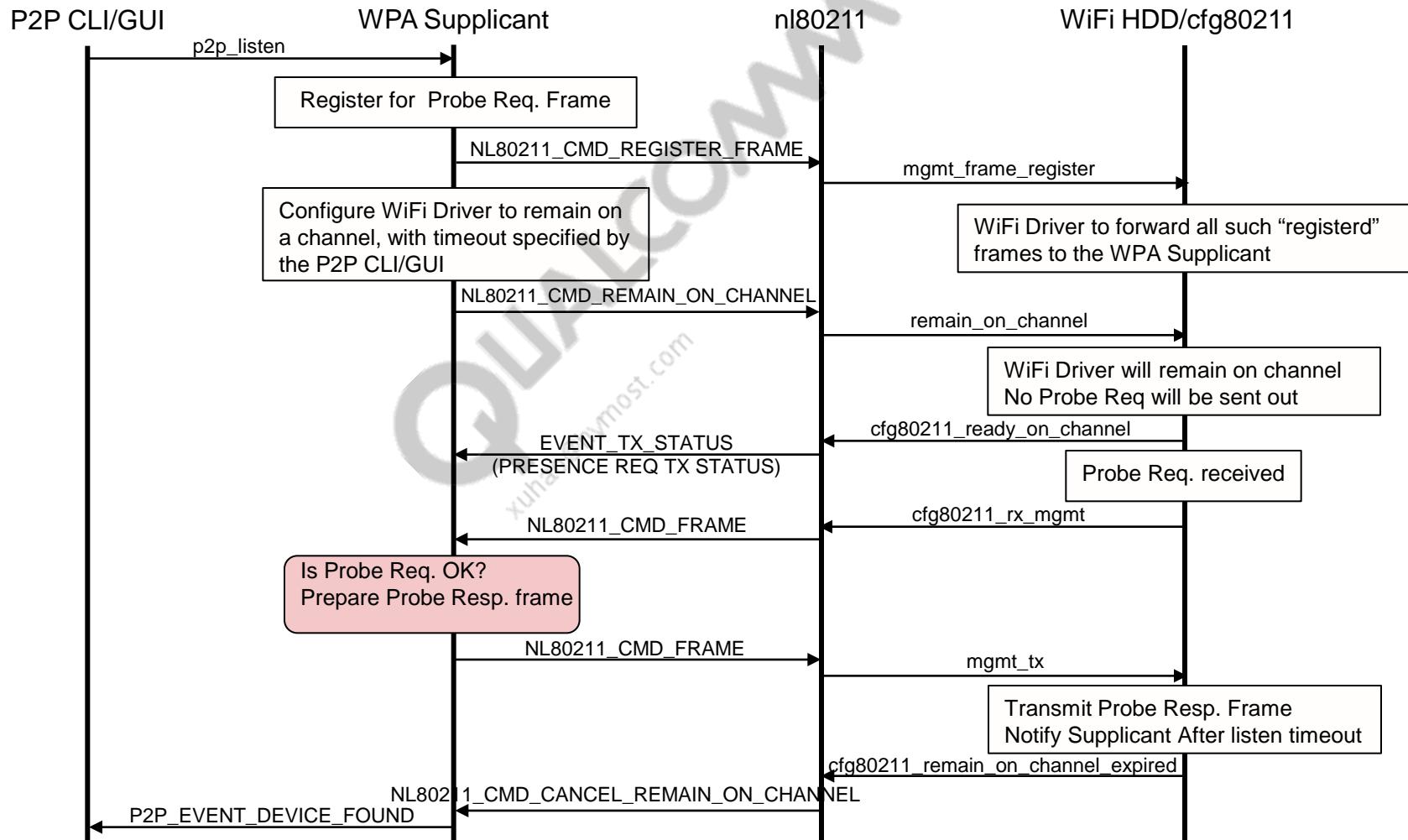
# P2P Find



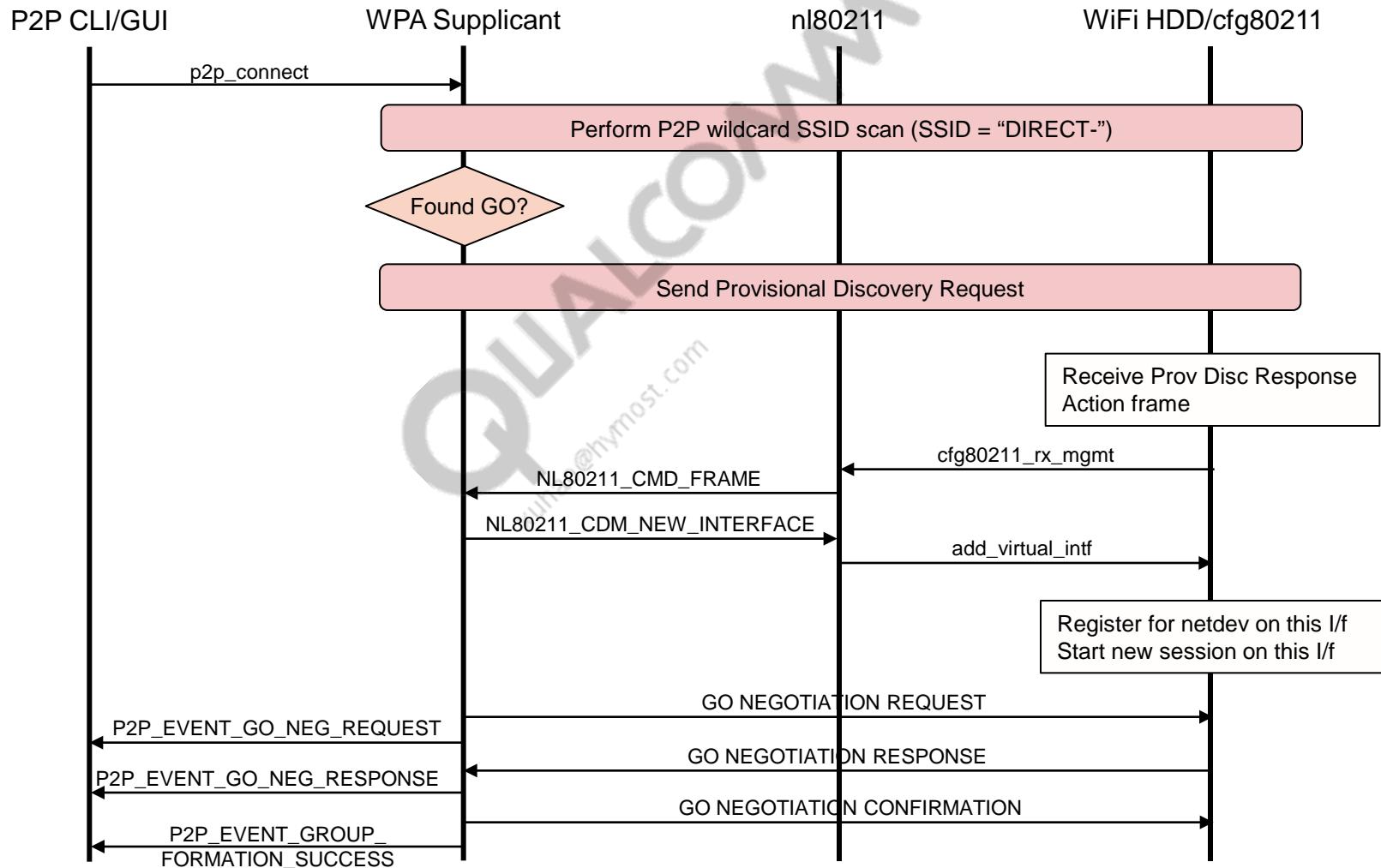
# P2P Stop Find



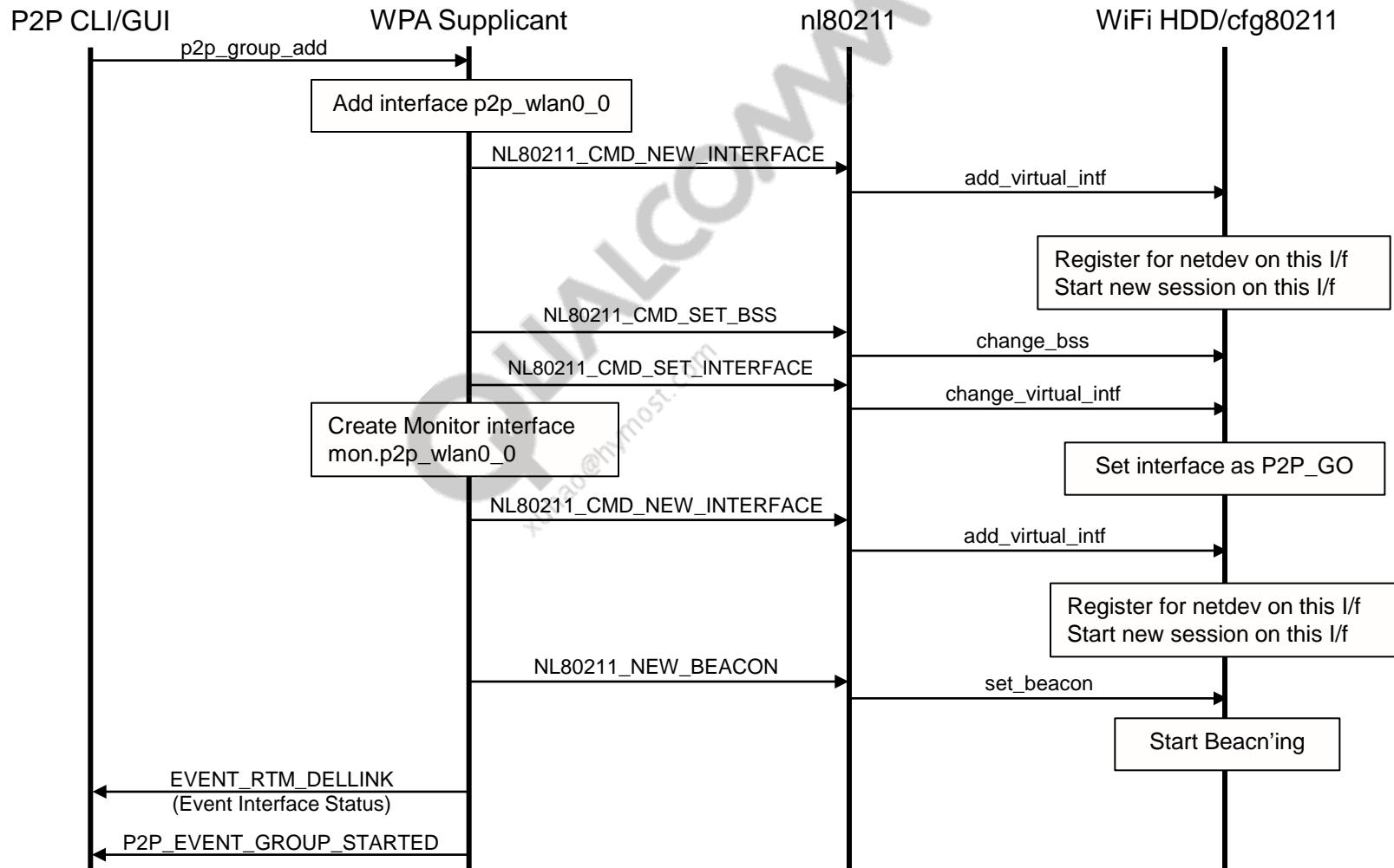
# P2P Listen



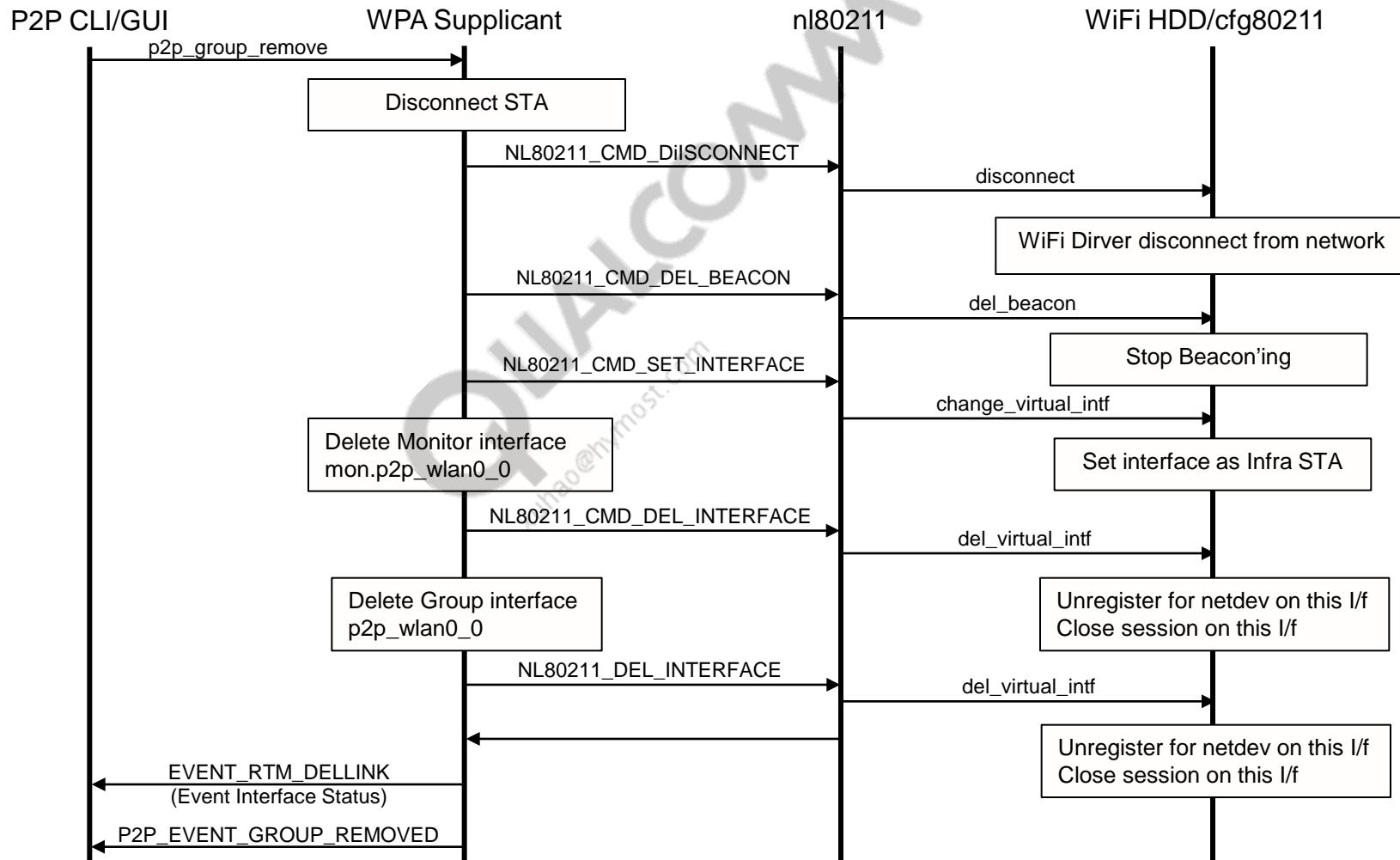
# P2P Connect



# P2P Group Add



# P2P Group Remove



# How to Set Up 5 GHz, HT 40 and parameters in p2p\_supplicants

1. Disable WIFI in WIFI GUI
2. adb pull /data/misc/wifi/ p2p\_supplicant.conf from a device
3. Put operating channel setup in p2p\_supplicant.conf  
e.g., p2p\_oper\_reg\_class=115  
      p2p\_oper\_channel=48
4. adb push p2p\_supplicant.conf /data/misc/wifi/ to the device

# How to Set Up p2p HT-40 at 5 GHz (1 of 2)

## P2P Devices A and B

1. Disable WIFI in WIFI GUI.
2. adb push p2p\_supplicant.conf to /data/misc/wifi/ in devices A and B
3. Add in /data/misc/wifi/WCNSS\_qcom\_cfg.ini (note: NOT /etc/firmware/wlan/prima/inis/WCNSS\_qcom\_cfg.ini) in devices A and B

gChannelBondingMode5GHz=1

BandCapability=0

\*) depending on test scenario, select operating channel setting or not

## How to Set Up p2p HT-40 at 5 GHz (2 of 2)

4. Follow commands to set up P2P connection at 5GHz band:

Devices A & B

[A & B] insmod /system/lib/modules/wlan.ko

[A] ifconfig wlan0 192.168.1.119

[B] ifconfig wlan0 192.168.1.120

[A & B] wpa\_supplicant -Dnl80211 -iwlan0 -c/data/misc/wifi/p2p\_supplicant.conf -B

[A & B] wpa\_cli

[B] p2p\_group\_add (B becomes GO and will beacon)

[A] p2p\_find

[A] p2p\_connect <B's mac\_addr> pbc

[B] p2p\_connect <A's mac\_addr> pbc

# Parameters in p2p\_supplicant.conf (1 of 2)

	Description	Option	default
p2p_listen_reg_class	P2P listen regulartory class	81 = supporting ch1,2,3,4,5,6,7,8,9,10,11,12,13	-
p2p_listen_channel	P2P listen channel	Ch1, 6, 11	-
p2p_oper_reg_class	P2P operation regulartory class  Operating reg class can be anything out of the list you mentioned. User has the choice to select which channels he would want to operate and respective reg class can be considered.	81 = supporting ch1,2,3,4,5,6,7,8,9,10,11,12,13  115=supporting ch36,40,44,48  124=supporting ch149, 153, 157, 161  116=supporting ch36,44  117=supporting ch40, 48  126=supporting ch149, 157  127=supporting ch153, 161	-
p2p_oper_channel	P2P operation channel	All 2.4GHz/5GHz channels	-
p2p_go_intent	The higher number indicates preference to become the GO.	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,1  5	7
p2p_ssid_postfix	Postfix data to add to the SSID	This data will be added to the end of the SSID after the functions p2p_get_oper_freq, p2p_add_device	NULL

## Parameters in p2p\_supplicant.conf (2 of 2)

	Description	Option	default
persistent_reconnect	Disable/enabled persistent reconnect for reinvocation of persistent groups. If enabled, invitations to reinvoke a persistent group will be accepted without separate authorization.	1=enabled 0=disabled	0
p2p_intra_bss	Intra BSS communication is supported	1=enabled 0=disabled	1
p2p_group_idle	Maximum idle time in seconds for P2P group  This value controls how long a P2P group is maintained after there is no other members in the group.  As a GO, this means no associated stations in the group. As a P2P client, this means no GO seen in scan results.  As a P2P client, the maximum idle time of P2P_MAX_CLIENT_IDLE seconds is enforced, i.e., this parameter is mainly meant for GO use and for P2P client, it can only be used to reduce the default timeout to smaller value.	0=no time limit (P2P group remains in active state indefinitely until explicitly removed.)	10

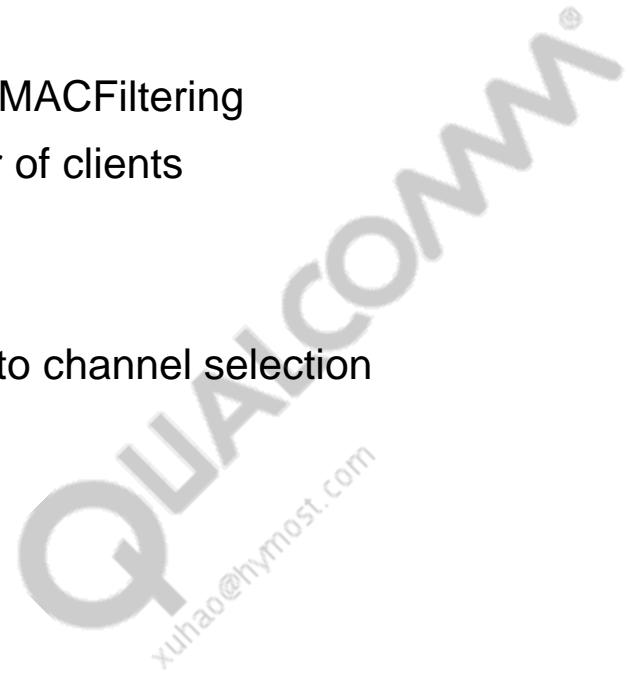
REDEFINING MOBILITY

SoftAP



## ■ SoftAP features

- AccessControlLists or MACFiltering
- Restricting the number of clients
- Auto-Shutdown mode
- Energy Detect mode
- Channel range and auto channel selection
- QCMobileAP APIs
- CLI Cmds for testing



## AccessControlLists or MACFiltering

REDEFINING MOBILITY

# Dynamic Configuration of ACLs

- Dynamic configuration of AccessControlLists (ACLs)
  - MAC filtering is done using ACLs
  - ACLs comprise Black List and White List
  - White List is the list of MAC addresses allowed to associate to the QCMobileAP
  - Black List is the list of MAC addresses not allowed to associate to the QCMobileAP
  - This feature provides the ability to disconnect any connected client without disrupting rest of clients; this includes applying the changes to the MAC filter table immediately without requiring a restart of AP mode.

# Dynamic Configuration of ACLs – Black List/White List

This feature provides the following functionality through private IOCTLs and event

- Dynamically add or remove a MAC address from the ACLs (Black List and White List) using IOCTL.
  - IOCTL is supported on QCMobileAP interface
  - Support both Black List and White List at the same time
  - macaddr\_acl value in hostapd.conf need to be configured as
    - ◆ 0 – For Black List
    - ◆ 1 – For White List
    - ◆ 2 – For both lists
- Driver sends IWEVCUSTOM event (JOIN\_UNKNOWN\_STA and MAC address) to the user space/framework if QCMobileAP receives a request from a client whose MAC address is not present in either of the ACLs
  - User space/framework can use this event to decide whether the unknown client can be added or removed from ACLs

# Dynamic Configuration of ACLs – Black List/White List (cont.)

- Disassociate a MAC address using an IOCTL
  - IOCTL is supported on QCMobileAP interface
  - User space/framework needs to ensure that disassociate IOCTL is followed by addition to Black List or deletion from White List (of an associated STA).
  - This enables users to disconnect Wi-Fi clients based on dynamic modifications done to the ACLs.
  - If the STA is not associated, the driver ensures that Disassociate IOCTL acts as a No-Op (No Operation).
- User space/framework needs to ensure that a MAC Address is not present in both the initial lists (Black List and White List) at the same time
  - Thereafter, if any MAC address is dynamically added, the driver ensures that this MAC address, if present in the other list than the one where the user wants it to be added, gets deleted from that list and then gets added to the user-requested list.

# Dynamic Configuration of ACLs – Black List/White List (cont.)

- The following private IOCTLs are supported for this feature
  - See \external\hostap\hostapd\qc\_sap\_ioctl.h for IOCTL definitions

IOCTL	Parameters
To configure the ACL parameters dynamically  QCSAP_IOCTL MODIFY_ACL (SIOCIWFIRSTPRIV+18)	Peer MAC address <ul style="list-style-type: none"><li>A string specifying the MAC address of the peer STA</li><li>It is inputted as a 6 octet MAC address with each octet inputted in hex, e.g., 0x00 0x0a 0xf5 0x89 0x89 0x90 for mac addr 000af5898990</li></ul> List type <ul style="list-style-type: none"><li>Specifies the type of list where the MAC address needs to be added or deleted</li><li>Valid values are 0 and 1</li><li>0 indicates white list</li><li>1 indicates black list</li></ul> Cmd type <ul style="list-style-type: none"><li>Specifies the type of operation (Add/Delete) to be performed on the passed MAC address</li><li>Valid values are 0 and 1</li><li>0 indicates that the MAC address passed has to be added to the list passed</li><li>1 indicates that the MAC address passed has to be deleted from the list passed</li></ul>
To disconnect a STA  QCSAP_IOCTL_DISASSOC_STA (SIOCIWFIRSTPRIV+11)	Peer MAC address <ul style="list-style-type: none"><li>A string specifying the MAC address of the peer STA</li><li>It is inputted as a 6 octet MAC address with each octet inputted in hex, e.g., 0x00 0x0a 0xf5 0x89 0x89 0x90 for MAC address 000af5898990</li></ul>

# Test Procedure – Dynamic Configuration of ACLs

## ■ Syntax for modify\_acl IOCTL

- iwpriv wlan0 modify\_acl <6 octet mac addr> <list type> <cmd type>
- List type
  - ◆ BLACK\_LIST – 0
  - ◆ WHITE\_LIST – 1
- Cmd type
  - ◆ ADD MAC – 0
  - ◆ REMOVE MAC – 1
- MAC address will be accepted as a 6 octet value with each octet in hex representation
  - ◆ 00:0a:f5:11:22:33 will be represented as 0x00 0x0a 0xf5 0x11 0x22 0x33 while using this IOCTL

## ■ Example

- To ADD a MAC Address 00:0a:f5:89:89:90 to the Black List
  - ◆ iwpriv wlan0 modify\_acl 0x00 0xf5 0x89 0x89 0x90 0 0
- To DELETE a MAC Address 00:0a:f5:89:89:90 from the White List
  - ◆ iwpriv wlan0 modify\_acl 0x00 0xf5 0x89 0x89 0x90 1 1

# Test Procedure – Dynamic Configuration of ACLs (cont.)

## ■ Syntax for disassoc\_sta IOCTL

- iwpriv wlan0 disassoc\_sta <6 octet mac address>
- MAC address will be accepted as a 6 octet value with each octet in hex representation
  - ◆ 00:0a:f5:11:22:33 will be represented as 0x00 0x0a 0xf5 0x11 0x22 0x33 while using this IOCTL

## ■ Example

- Disassociate STA with MAC address 00:0a:f5:11:22:33 from QCMobileAP
  - ◆ iwpriv wlan0 disassoc\_sta 0x00 0x0a 0xf5 0x11 0x22 0x33

# Test Procedure – Dynamic Configuration of ACLs (cont.)

S.No.	macaddr_acl (in hostapd.conf)	ACL used	CLI cmd(s) to dynamically add or remove STAs from Lists	Comments
1	0	BlackList	Add to BlackList <code>iwpriv wlan0 modify_acl &lt;6 octet mac addr&gt; 0 0</code> Remove from Blacklist <code>iwpriv wlan0 modify_acl &lt;6 octet mac addr&gt; 0 1</code> <b>Note :</b> Disassoc IOCTL needs to be executed after addition to BlackList <code>iwpriv wlan0 disassoc_sta &lt;6 octet mac address&gt;</code>	<ul style="list-style-type: none"> <li>- wlan0 is the interface name in hostapd.conf</li> <li>- SoftAP will ACCEPT connections from ALL STAs UNLESS they are DENIED in BlackList.</li> </ul>
2	1	WhiteList	Add to WhiteList <code>iwpriv wlan0 modify_acl &lt;6 octet mac addr&gt; 1 0</code> Remove from WhiteList <code>iwpriv wlan0 modify_acl &lt;6 octet mac addr&gt; 1 1</code> <b>Note :</b> Disassoc IOCTL needs to be executed after removal from WhiteList <code>iwpriv wlan0 disassoc_sta &lt;6 octet mac address&gt;</code>	<ul style="list-style-type: none"> <li>- wlan0 is the interface name in hostapd.conf</li> <li>- SoftAP will DENY connections from ALL STAs UNLESS they are ACCEPTED in WhiteList.</li> </ul>

# Test Procedure – Dynamic Configuration of ACLs (cont.)

S.No.	macaddr_acl (in hostapd.conf)	ACL used	CLI cmd To dynamically add or remove STAs from lists	Comments
3	2	Both lists	All above cmds apply	All above comments apply  Driver sends IWEVCUSTOM Event (with "JOIN_UNKNOWN_STA" and MAC Address) to the UserSpace/Framework if SoftAP receives a request from a client whose MAC Address is NOT present in either of the Lists.

## Restricting the Number of Clients

REDEFINING MOBILITY

# Dynamic Configuration of MaxNumberOfClients

- This feature provides the ability to dynamically change the maximum number of clients supported by the QCMobileAP without restarting the AP and hence not affecting the already connected clients.
- Parameters are configured dynamically through a private IOCTL
  - IOCTL is supported on QCMobileAP interface
  - IOCTL – QCSAP\_IOCTL\_SETPARAM (SIOCIWFIRSTPRIV+0)
  - SUBIOCTL – QCSAP\_PARAM\_MAX\_ASSOC
  - See \external\hostap\hostapd\qc\_sap\_ioctl.h for IOCTL Definitions
- IOCTL requires the Max\_allowed\_clients parameter.
  - Max\_allowed\_clients – An integer specifying the soft limit for the maximum number of associations allowed to the QCMobileAP; range is 0 to 10
  - If user passes a value less than 0, an invalid status is returned back to user
  - If the user passes a value more than 10, number of clients supported will be automatically set to 10 by the driver
- If the current number of associations are already more than the new value of supported max clients set by the user, those associations will continue to exist. However, any new STA trying to join the QCMobileAP will be denied until the number of associations go below the new set user limit.

## Dynamic Configuration of MaxNumberOfClients (cont.)

- If the value passed is 5 by the user to the IOCTL and the number of clients connected to QCMobileAP is four
  - Driver will only allow one more client to associate after which it will reject association requests from new STAs
- If the value passed is 5 by the user to the IOCTL and the number of clients connected to QCMobileAP is six
  - Driver will *not* allow any clients to associate
  - Driver allows clients to associate only when two or more clients disassociate to make the current associated clients less than five.
- Driver does not take care of race conditions, i.e., if the IOCTL to change the maximum number of supported clients and an association request from a new STA come at almost the same time, the request that reached the driver first gets executed first
- Get the current soft limit for maximum number of clients supported through Private IOCTL
  - IOCTL is supported on QCmobileAP interface
  - IOCTL - QCSAP\_IOCTL\_GETPARAM (SIOCIWFIRSTPRIV+1)
  - SUBIOCTL - QCSAP\_PARAM\_MAX\_ASSOC
  - See \external\hostap\hostapd\qc\_sap\_ioctl.h for IOCTL definitions

# Test Procedure – Dynamic Configuration of MaxNumOfClients

Steps	Procedure	CLI cmd (ADB Shell)	Comments
1	To get the current maximum soft limit for maximum no of associations allowed on softap	<code>iwpriv wlan0 getMaxAssoc</code>	wlan0 is the interface name in hostapd.conf
2	To set the soft limit for maximum no of associations allowed on softap	<code>iwpriv wlan0 setMaxAssoc &lt;limit&gt;</code>	Range is 0 -10 <b>Example :</b> If the value passed is 5 by the user to the IOCTL and the no. of clients connected to softap are 6 then -Driver will NOT allow any clients to associate. -Driver allows clients to associate only when 2 or more clients disassociate to make the current associated clients less than 5

## Auto-Shutdown Mode

REDEFINING MOBILITY

# Auto-Shutdown

- Auto-Shutdown
  - Disable AP and turn off Wi-Fi after a user-configurable period of inactivity, i.e., no clients connected
  - Reduces unnecessary power drain when no clients are connected
  - User must manually restart the AP after auto-shutdown
- This feature can be configured by changing the following param in WCNSS\_qcom\_cfg.ini
  - gAPAutoShutOff
  - Value of 0 disables AutoShutdown feature
- Driver sends IWEVCUSTOM Event (AUTO-SHUT.indication) to UserSpace. QCMobileAP needs to be stopped from UserSpace.
- QCMobileAP needs to be restarted after shutdown

## Energy Detect Mode

REDEFINING MOBILITY

# Energy Detect Mode

- Energy Detect mode
  - Turn on only the RF energy detector instead of the complete Rx chain when listening for packets
  - Reduces power consumption when no packets are being received
  - Possible performance impact, as range is reduced up to 65%
  - Impacts both QCMobileAP and STA modes
- This feature can be configured by changing the following param in WCNSS\_qcom\_cfg.ini
  - gEnablePhyAgcListenMode

## Energy Detect Mode (*cont.*)

- gEnablePhyAgcListenMode in WCNSS\_qcom\_cfg.ini
  - Valid values are 0 to 128
  - 128 means disable Energy Detect feature
  - 10 to 128 are reserved
  - The EDET threshold mapping for 0 to 9 is as follows in 3-dB steps:
    - ◆ 0 = -60 dBm, 1 = -63 dBm,..... 7 = -81 dBm, 8 = -84 dBm, 9 = -87 dBm
    - ◆ 7 is recommended value from system if feature is to be enabled
    - ◆ Setting 0 would yield the highest power saving (in a noisy environment) at the cost of more range
- The range impact is approximately calculated as
  - Range Loss (dB) = EDET threshold level (dBm) + 97 dBm

## Channel Range and Auto Channel Selection

REDEFINING MOBILITY

# Channel Range and Auto Channel Selection

- The QCMobileAP Channel Range feature restricts QCMobileAP Auto channel selection algorithm to select the best channel out of a given input candidate channel range.
- This feature gives OEMs better control to decide the operating channel of QCMobileAP.
- This feature can be used for QCMobileAP 5-GHz regulatory compliance.
- This will enable OEMs to avoid channels restricted by regional regulations and achieve adherence to regulatory restrictions.
- Different regulatory domain has different set of operating channels, so OEMs can restrict the QCMobileAP channel as per their regulatory domain requirement.
- OEMs can configure channel range and operating band to avoid using Radar channels.

# Channel Range and Auto Channel Selection – Config Params

Feature	WCNSS_qcom_cfg.ini	Example	Comments
Auto Channel Selection	gApAutoChannelSelection	gApAutoChannelSelection=1	AutoChannel Selection is set through ini file
Channel Range	gAPChannelSelectStartChannel gAPChannelSelectEndChannel gAPChannelSelectOperatingBand <ul style="list-style-type: none"><li>■ 0 – 2.4 GHz ,</li><li>■ 1 – LOW, 5 GHz</li><li>■ 2 – MID, 5 GHz</li><li>■ 3 – HIGH, 5 GHz</li><li>■ 4 - 4.9 GHz</li></ul>	gAPChannelSelectStartChanne l=149 gAPChannelSelectEndChannel =165 gAPChannelSelectOperatingBa nd=3	AutoChannel selection algorithm selects best channel among channels 149 to 165

# Channel Range and Auto Channel Selection – Private IOCTLs

- Auto Channel Selection and Channel Range features can be set through Private IOCTLs
- Private IOCTLs are supported on STA interface
  - iwpriv wlan0 setsapchannels <start\_channel> <end\_channel> <band>
  - iwpriv wlan0 setAutoChannel 1
- These IOCTLs need to be called after Driver load and before hostapd start
- Example
  - insmod /system/lib/modules/prima/prima\_wlan.ko
  - iwpriv wlan0 setAutoChannel 1
  - iwpriv wlan0 setsapchannels 153 165 3
  - hostapd -ddd /data/misc/wifi/hostapd.conf



**32 STAs Support Using Virtual STAs**

REDEFINING MOBILITY

# 32 STAs Support using Virtual STAs

- Virtual Station
  - Software based station supported by FW and Host SW
  - No fixed/assigned station descriptor in HW
  - If Virtual Station support is enabled then SoftAP allows up to 32 STAs
  - Multiplexing of HW resources is used to support additional num of STAs
- Config Param for Virtual Station support
  - gEnableVSTA=1 need to be added in WCNSS\_qcom\_cfg.ini
- SoftAP supports only 10 STAs if Virtual Station support is not enabled
  - This is due to HW resource constraint

# 32 STAs Support using Virtual STAs

(cont.)

- General Purpose Station

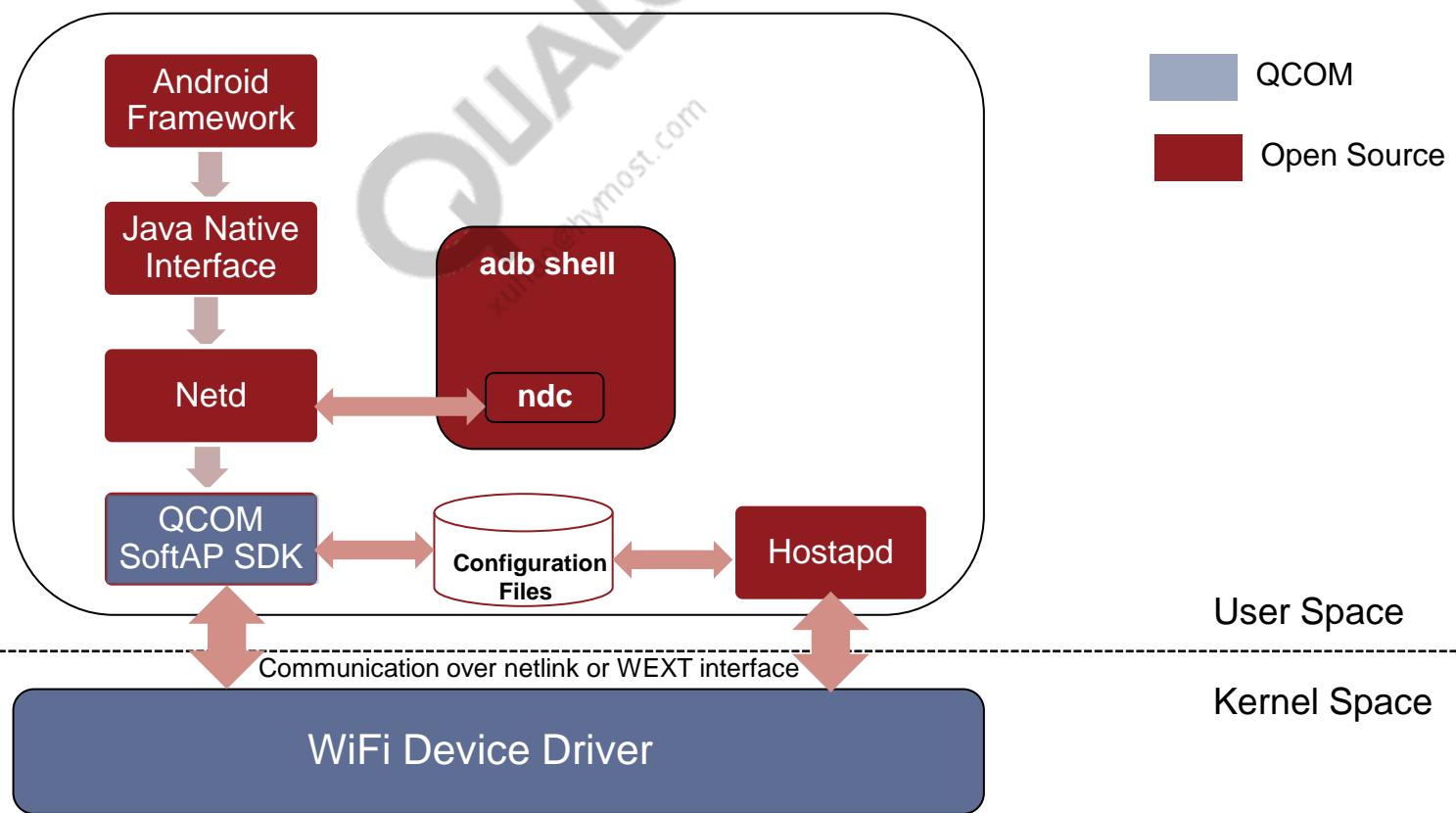
- A HW station which can be used for any virtual station one at a time
- Used for multiplexing multiple Software stations during Swap In and Swap Out procedures
- **Swap In:** Bringing in the virtual station configuration to General Purpose Station configuration
- **Swap Out:** Pulling out the station from General Purpose station to its corresponding virtual station.

REDEFINING MOBILITY

## QCMobileAP APIs

# QCOM SoftAP SDK

- QCOM SoftAP SDK processes configuration commands sent by the GUI Application for SoftAP
- SDK code is located in system/qcom/softap/sdk
- JNI interface is located in system/qcom/softap/jni

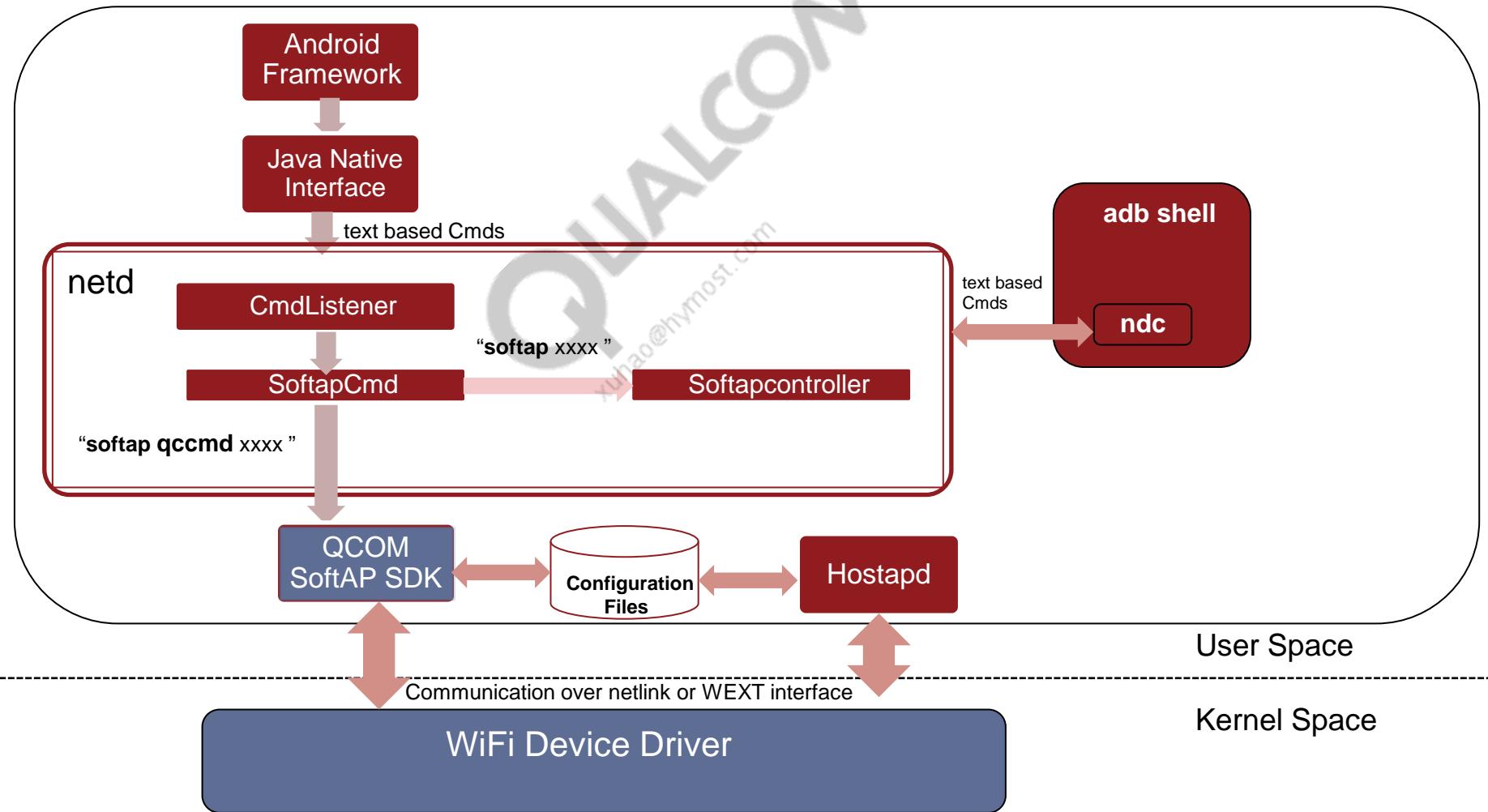


# Netd System Daemon

- The Android net daemon (Netd) application receives text-based configuration commands from the user applications or from the framework
- Netd executes the commands based on the text present in the command
- system/netd has CommandListener.cpp which routes the commands to a related controller
- SoftapController.cpp is the default controller in Android for processing Softap cmds
  - Text commands with “softap” at the beginning are routed to SoftapController.cpp
- Code embraced by QCOM\_WLAN defined in Netd are changes made by Qualcomm to process cmds using Qualcomm SoftAP SDK
  - Text commands with “softap qccmd” at the beginning are routed to SoftAP SDK

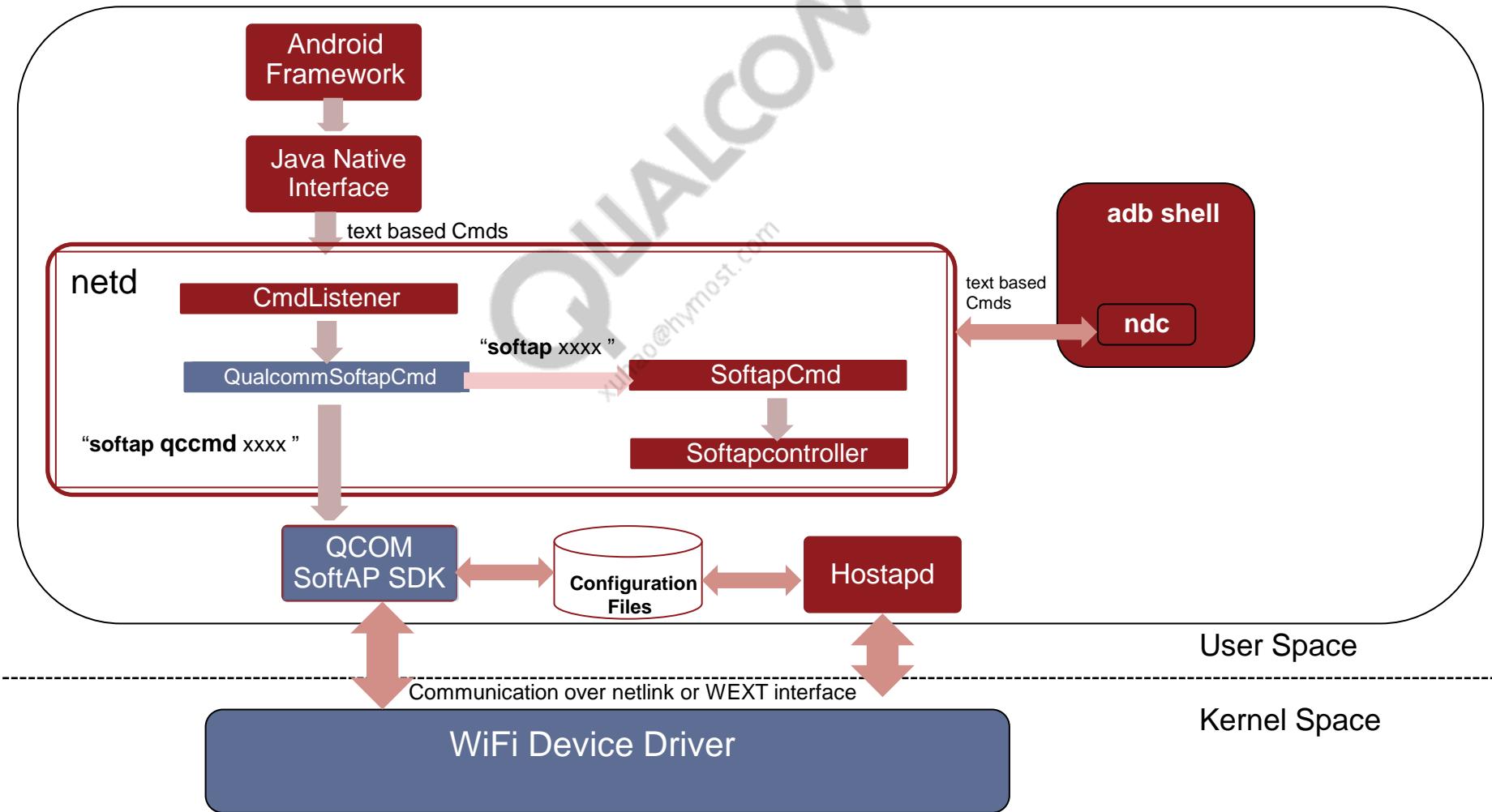
# SoftAP Command Processing

- SoftapCmd in netd forwards the cmds to SDK if they start with “softap qccmd”
- Otherwise the cmds are forwarded to Softapcontroller for processing



# SoftAP Command Processing in jb\_mr1 branch

- Qualcomm-specific changes in CmdListener.cpp are moved to QualcommSoftapCmd.cpp in jb\_mr1
- QualcommSoftapCmd forwards the cmds to SDK if they start with “softap qccmd”
- Otherwise, the cmds are forwarded to Softapcontroller for processing



# QCMobileAP APIs

- QCMobileAP APIs
  - This section contains information on the APIs supported by QCMobileAP SDK
  - QCMobileAP APIs can also be tested from the adb shell using the NDC tool
- Example
  - `ndc softap qccmd get channel`
  - “get channel” is the QCMobileAP API processed by the QCMobileAP SDK

## QCMobileAP APIs (cont.)

S.No.	API functionality	ndc command for QCMobileAP API
1	Enable SAP	ndc softap startap
2	Disable SAP	ndc softap stopap
3	Reset SAP	ndc softap stopap ndc softap startap
4	Changing SSID	ndc softap qccmd set ssid=qualcomm123412341234123412341234 ndc softap qccmd get ssid
5	Changing channels	ndc softap qccmd set channel=x ndc softap qccmd get channel
6	Beacon interval	ndc softap qccmd set beacon_int=1000 ndc softap qccmd get beacon_int
7	DTIM period	ndc softap qccmd set dtim_period=5 ndc softap qccmd get beacon_int

## QCMobileAP APIs (cont.)

S.No.	API functionality	ndc command for QCMobileAP API
8	WEP 64	<code>ndc softap qccmd set security_mode=1</code> <code>ndc softap qccmd set wep_key0=1234567890</code> <code>ndc softap qccmd set wep_key1=1234567890</code> <code>ndc softap qccmd set wep_key2=1234567890</code> <code>ndc softap qccmd set wep_key3=1234567890</code> <code>ndc softap qccmd set wep_default_key=0</code>
9	WPA-TKIP	<code>ndc softap qccmd set security_mode=2</code> <code>ndc softap qccmd set rsn_pairwise=TKIP</code> <code>ndc softap qccmd set wpa_passphrase=qualcomm</code> <code>ndc softap qccmd get security_mode</code> <code>ndc softap qccmd get rsn_pairwise</code> <code>ndc softap qccmd get wpa_passphrase</code>
10	WPA2-AES	<code>ndc softap qccmd set security_mode=3</code> <code>ndc softap qccmd set rsn_pairwise=CCMP</code> <code>ndc softap qccmd set wpa_passphrase=qualcomm</code> <code>ndc softap qccmd get security_mode</code> <code>ndc softap qccmd get rsn_pairwise</code> <code>ndc softap qccmd get wpa_passphrase</code>

## QCMobileAP APIs (cont.)

S.No.	API functionality	ndc command for QCMobileAP API
11	AP mode	ndc softap qccmd set hw_mode=g ndc softap qccmd set hw_mode=b ndc softap qccmd set hw_mode=n ndc softap qccmd set hw_mode=g-only ndc softap qccmd set hw_mode=b-only ndc softap qccmd set hw_mode=n-only ndc softap qccmd get hw_mode
12	WPS PBC	ndc softap qccmd set security_mode=3 ndc softap qccmd set rsn_pairwise=CCMP ndc softap qccmd set wpa_passphrase=qualcomm ndc softap qccmd set wps_state=1 ndc softap qccmd set config_methods=0 (PBC)
13	Hidden SSID	ndc softap qccmd set ignore_broadcast_ssid=1
14	MAC Filter mode	ndc softap qccmd set macaddr_acl=2 ndc softap qccmd get macaddr_acl
15	802.11d	ndc softap qccmd set country_code=IN ndc softap qccmd get country_code

# CLI Commands for Testing

REDEFINING MOBILITY

# CLI Commands for Testing

## Procedure to bring up SoftAP on wlan0 interface

Steps	Procedure	CLI cmd (ADB Shell)	Comments
1	Install the WLAN Driver	<i>insmod /system/lib/modules/prima/prima_wlan.ko</i>	Ensure that WiFi is NOT turned ON from GUI.
2	Start Hostapd	<i>hostapd -ddd /data/misc/wifi/hostapd.conf</i>	wlan0 is the interface name in hostapd.conf
3	Configure the IPAddress	<i>ifconfig wlan0 192.168.43.1</i>	wlan0 is the interface name in hostapd.conf
4	Start DHCP Server	<i>system/bin/dnsmasq -x /data/dnsmasq.pid --no-resolv --no-poll --dhcp-range=192.168.43.10,192.168.43.100</i>	

# CLI Commands for Testing (cont.)

## Procedure to bring up SoftAP on softap.0 interface

Steps	Procedure	CLI cmd (ADB Shell)	Comments
1	Install the WLAN Driver	<i>insmod /system/lib/modules/prima/prima_wlan.ko</i>	Ensure that WiFi is NOT turned ON from GUI.
2	Create softap.0 interface	<i>iwpriv wlan0 initAP <u>Note : IOCTL is on wlan0 interface ONLY</u></i>	<b>softap.0 is the interface name in hostapd.conf</b>
3	Start Hostapd	<i>hostapd -ddd /data/misc/wifi/hostapd.conf</i>	softap.0 is the interface name in hostapd.conf
4	Configure the IPAddress	<i>ifconfig softap.0 192.168.43.1</i>	softap.0 is the interface name in hostapd.conf

# CLI Commands for Testing (cont.)

## Procedure to turn ON WPS on SoftAP using CLI cmds

Steps	Procedure	CLI cmd (ADB Shell)	Comments
1-4	Bringup SoftAP using CLI cmds	Refer the the table above for the <b>Procedure to bring up SoftAP using CLI cmds</b>	Ensure that hostapd.conf has WPS params set properly
5	Start hostapd_cli	<code>hostapd_cli -p/data/misc/wifi/hostapd -iwlan0</code>	wlan0 is the interface name in hostapd.conf
6	Trigger wpa_pbc from the CLI of hostapd_cli	<code>wps_pbc</code>	Start WPS on the STA also

REDEFINING MOBILITY

Hotspot 2.0



# Hotspot 2.0 – Overview

- Hotspot 2.0
  - Wi-Fi Alliance Specification
  - Wi-Fi network Discovery and Selection
    - To simplify and automate access to public Wi-Fi networks
    - STA identifies which access points are suitable for its needs
    - Authenticate to a remote service provider using suitable credentials
  - WPA-2 Enterprise security is required
  - Uses functionality from IEEE 802.11u
- IEEE Std 802.11u (Interworking with External Networks)
  - Allows STAs to access services provided by an external network
  - Generic Advertisement Service (GAS)
    - Enables STAs to discover network services
  - Access Network Query Protocol (ANQP)
- Wi-Fi Passpoint
  - Certification for Hotspot 2.0 STAs and Aps
- See [Q5]for details on the Hotspot 2.0 feature

## Hotspot 2.0 – Overview (cont.)

- Wi-Fi Alliance Hotspot 2.0 specification needs only subset of IEEE 802.11u standard
  - IEEE 802.11u standard is different from Wi-Fi Alliance Hotspot 2.0 Specification
- IEEE 802.11u features that are supported
  - Network discovery and selection
  - Discovery of suitable networks through the advertisement of access network type, roaming consortium and venue information, via management frames
  - Selection of a suitable IEEE 802.11 infrastructure using advertisement services (e.g., Access Network Query Protocol (ANQP) or an IEEE 802.21 Information Server) in the BSS or in an external network reachable via the BSS.
  - Selection of an SSPN or external network with its corresponding IEEE 802.11 infrastructure
- IEEE 802.11u features that are NOT supported
  - Emergency services
  - Emergency Call and Network Alert support at the link level
  - QoS Map distribution
  - SSPN interface service between the AP and the SSPN

REDEFINING MOBILITY

TDLS



# TDLS

- Tunneled Direct-Link Setup (TDLS)
  - Based on IEEE Std 802.11z
  - Provides device-to-device connectivity
  - TDLS allows devices that are associated with the same AP to establish a direct link with each other
  - Wi-Fi direct allows devices to connect directly to each other without the need for an AP
- TDLS benefits
  - Data transmission is more efficient
  - Devices can use highest level of shared capabilities, regardless of AP's capabilities
  - Provides support for devices to negotiate an alternative channel
- See [[Q10](#)] for details on the TDLS feature

# WCN36x0 Common/Android Platform References

Ref.	Document
<b>Qualcomm</b>	
Q1	<i>Application Note: Software Glossary for Customers</i>
Q2	<i>Presentation: Overview of MSM8960 WLAN in Android</i>
Q3	<i>Application Note: WCNSS WLAN FTM PTT Messages</i>
Q4	<i>Presentation: WCN36x0 BT/WLAN/FM FTM Application Development for Windows</i>
Q5	<i>Presentation: Hotspot 2.0 Overview</i>
Q7	<i>Presentation : WCN36x0 Bluetooth® Overview for Android™ JellyBean</i>
Q8	WCNSS_Legacy_CCX_Fast_Roaming
Q9	QCA ANDROID WCN36X0 CENTRAL REGULATORY DOMAIN AGENT (CRDA)
Q10	Presentation: TUNNELED DIRECT LINK SETUP (TDLS)
Q11	MULTI CHANNEL CONCURRENCY (MCC) AND SINGLE CHANNEL CONCURRENCY (SCC)
Q12	WCN36x0_Android_WAPI_Integration_Testing
Q13	WCN3660™ Peer-to-Peer Interface Specification
Q14	Parameters_in_p2p_supplicant_conf

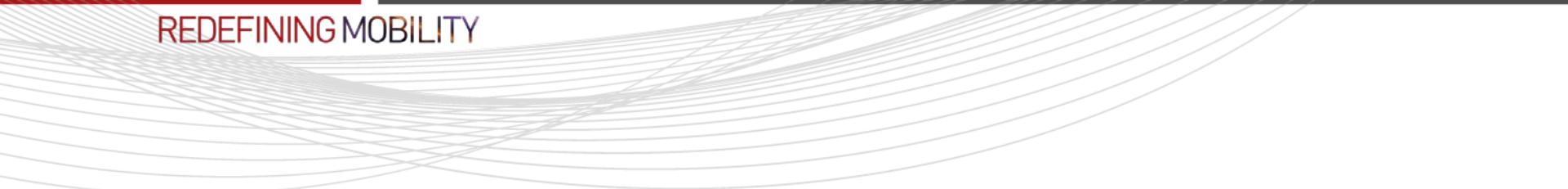
# WCN36x0 Windows Platform References

Ref.	Document	
<b>Qualcomm</b>		
Q15	<i>Application Note: Software Glossary for Customers</i>	CL93-V3077-1
Q16	<i>MSM8974-WCN36x0 WLAN Configuration and Debugging Guide for Windows Phone 8</i>	80-Y5143-2
Q17	<i>Presentation: WCN36x0 BT/WLAN/FM FTM Application Development for Windows</i>	80-N5047-22
Q18	<i>Presentation: MSM8974 FC Release Wi-Fi Software Report</i>	80-NA157-149
<b>Standard</b>		
S1	<i>Device Power States for Network Adapters</i>	<a href="http://msdn.microsoft.com/en-us/library/windows/hardware/ff546427(v=vs.85).aspx">http://msdn.microsoft.com/en-us/library/windows/hardware/ff546427(v=vs.85).aspx</a>
S2	<i>Understanding Connected Standby</i>	<a href="http://channel9.msdn.com/events/BUILD/BUILD2011/HW-456T">http://channel9.msdn.com/events/BUILD/BUILD2011/HW-456T</a>



Questions?

+uhao@hymost.com



REDEFINING MOBILITY