DHD (DONGLE HOST DRIVER) OFFLOAD

Rev 0.1

5300 California Avenue • Irvine, CA 92617 USA •

Revision History

Revision Number	Date	Change Description
0.1	September 10, 2015	Initial revision.

TABLE OF CONTENTS

1 OVERVIEW

1.1 INTRODUCTION

DHD (Dongle Host Driver) offload feature refers to the capability of system to offload the traffic classification, modification and forwarding to/from WLAN to hardware accelerator, thus reducing the host CPU utilization.

.

1.2 SCOPE

The scope of this document is restricted to high level description of the feature, configuration parameters and its usage for the users.

- Applicability of "DHD Offload" term within the scope of this document is as follows:
  - Offload specifically means "Host side processing offload"
  - Does not apply to "NIC" mode WLAN drivers (including WLAN Dongles used in NIC mode).
  - Dongle refers to a single WLAN card (43602, 4366 + variants and such future devices).
  - "Dongle Offload" and "DHD Offload" terms may be used interchangeably within this document.
- DHD offload is currently supported only on following Runner hardware accelerator based devices – BCM963138 and BCM96838 (and their other chip variants).
- Mechanism of flow learning/provisioning by flow-cache into the Runner accelerator is outside the scope of this document.
- This document does not intend to highlight the exact code execution path but limited to briefly provide the high level packet/processing flow.

1.3 ABBREVIATIONS

DHD	Dongle Host Driver
AC	Access Category

2 BUILD

By default the DHD Offload feature is enabled in 963138/96838 gateway profiles starting in release 4.16L.04

Note: DHD Offload requires relatively more memory than non-offload case and this memory needs to be reserved at system boot time. Careful consideration is required to enable this feature on low memory devices.

To enable/disable the DHD offload feature, use the *make menuconfig* command on Linux command prompt before build.

```
$ make menuconfig
```

Then navigate to menuconfig→Packet Acceleration →[ ] DHD Runner Acceleration Support

3 MEMORY RESERVATION

The amount of memory to be reserved for DHD Offload is specified in CFE boot loader parameters as shown in example below from 963138 reference board. These values are passed to Linux during system boot to reserve the memory.

```
Base MAC Address                  : 02:10:18:89:82:01
PSI Size (1-128) KBytes           : 48
Enable Backup PSI [0|1]           : 0
System Log Size (0-256) KBytes    : 0
Auxillary File System Size Percent: 0
MC memory allocation (MB)         : 4
TM memory allocation (MB)         : 44
DHD 0 memory allocation (MB)      : 14
DHD 1 memory allocation (MB)      : 7
DHD 2 memory allocation (MB)      : 0
```

DHD 0/1/2 refers to the radio index that gets assigned to the specific WLAN dongle during PCIe scan (Refer to "Radio Number Assignment" section). Specified memory will be reserved prior to Linux boot and will not be available for Linux usage. In the above example, first WLAN dongle will be allocated 14MB while 7MB for second WLAN dongle. (Refer to "Flow-Ring Allocation Controls" section for more details).

Using **default configuration** (Refer to "Flow-Ring Profile" section) and offload for all access categories, each WLAN station would require ~216KB of memory. So, based on the maximum number of clients/stations supported by the dongle, required memory will be:

- Dongle with 32 station support: ~7MB
- Dongle with 64 station support: ~14MB

Note#1: In the above example, "TM Memory allocation" for 963138 platforms is also changed to 44MB from the default 20MB. This increase in TM memory is **recommended** to increase the total number of Runner packet buffers to accommodate the large buffering requirements towards WLAN dongle.

Note#2: For 96838 platforms, **recommended** TM memory requirement is 36MB.

4 DHD OFFLOAD

4.1 INTERACTION WITH DONGLE

All communication with Dongle is done using descriptor rings irrespective of offload or not. Dongle firmware remains same in both the cases and dongle gets configured accordingly to work in offload/non-offload modes. Following type of descriptor rings are used:

- Packet Receive Path related rings
  - **RxBuf Post Ring (RX_POST)**: DHD/Runner puts the descriptor for packet receive buffer in this ring. Dongle uses these buffers to receive the packets from WLAN Clients and passes them back to DHD/Runner (using RxPkt Complete Ring). Only one RX_POST ring per Dongle is created.
  - **RxPkt Complete Ring (RX_CMPL)**: Dongle puts the descriptors for received packets in this ring and signals the DHD/Runner to process the packet(s). Only one RX_CMPL ring per Dongle is created.
- Packet Transmit Path related rings
  - **TxPkt Post Flow-ring (TX_POST/Flow-Ring):** DHD/Runner puts the descriptor for packets they need to transmit towards Dongle (i.e. WLAN Station) and signals the Dongle to process the packet(s). Up to four flow-rings are created per station for each access category (VO, VI, BE, BK). Each flow-ring could be of different size and the size is configurable through boot time profile (Refer to section "Flow-Ring Allocation Controls"). There is also a flow-ring for Broadcast/Multicast traffic for every BSS/SSID.
  - **TxPkt Complete Ring (TX_CMPL)**: Dongle puts the descriptors related to transmitted packets in this ring and signals the DHD/Runner to indicate completion of transmission, so DHD/Runner can free up those packet buffers. Only one TX_CMPL ring per Dongle is created.

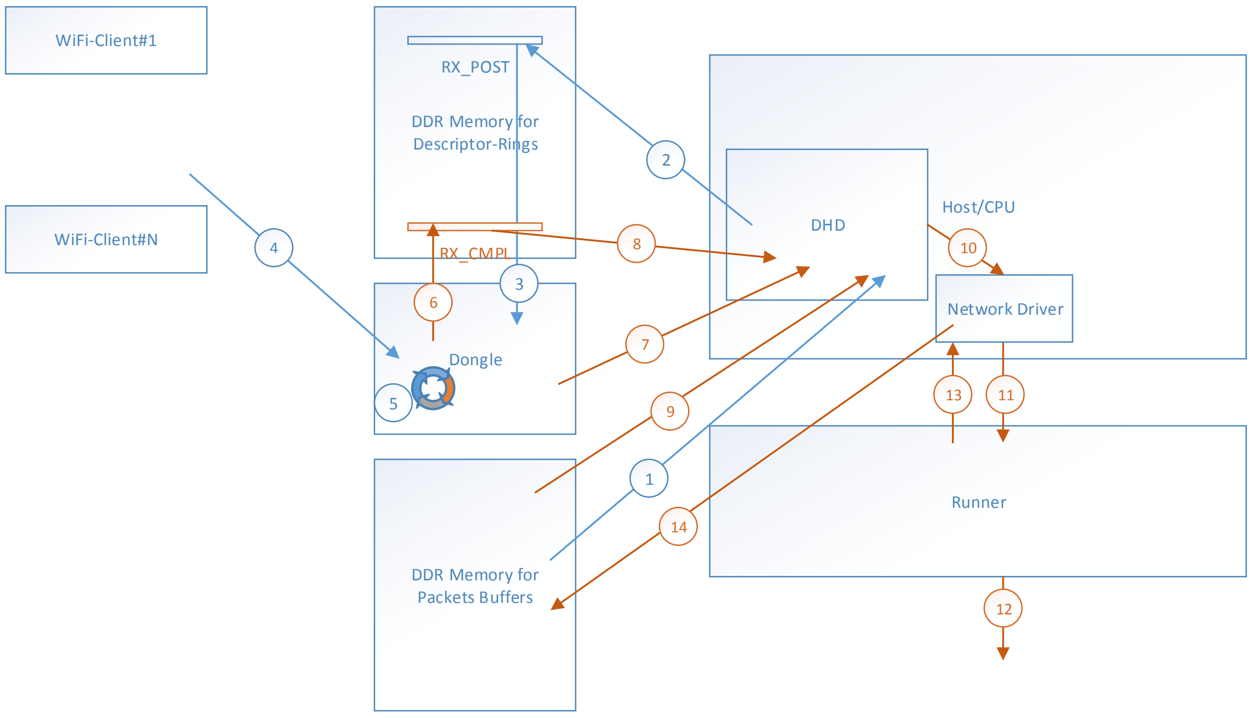
4.2 DESCRIPTOR RING ALLOCATION

DHD allocates all the rings at initialization time based on the configured sizes and provides the required information to Dongle.

4.3 NON-OFFLOAD/DHD PATH

When DHD Offload feature is disabled, based on the number of stations supported by dongle, DHD allocates the Flow-Rings of size 512 each. When required, DHD has mechanism to provide additional buffering for each station in addition to these 512 size flow-rings.

4.3.1 PACKET RECEIVE (DONGLE TO DHD)

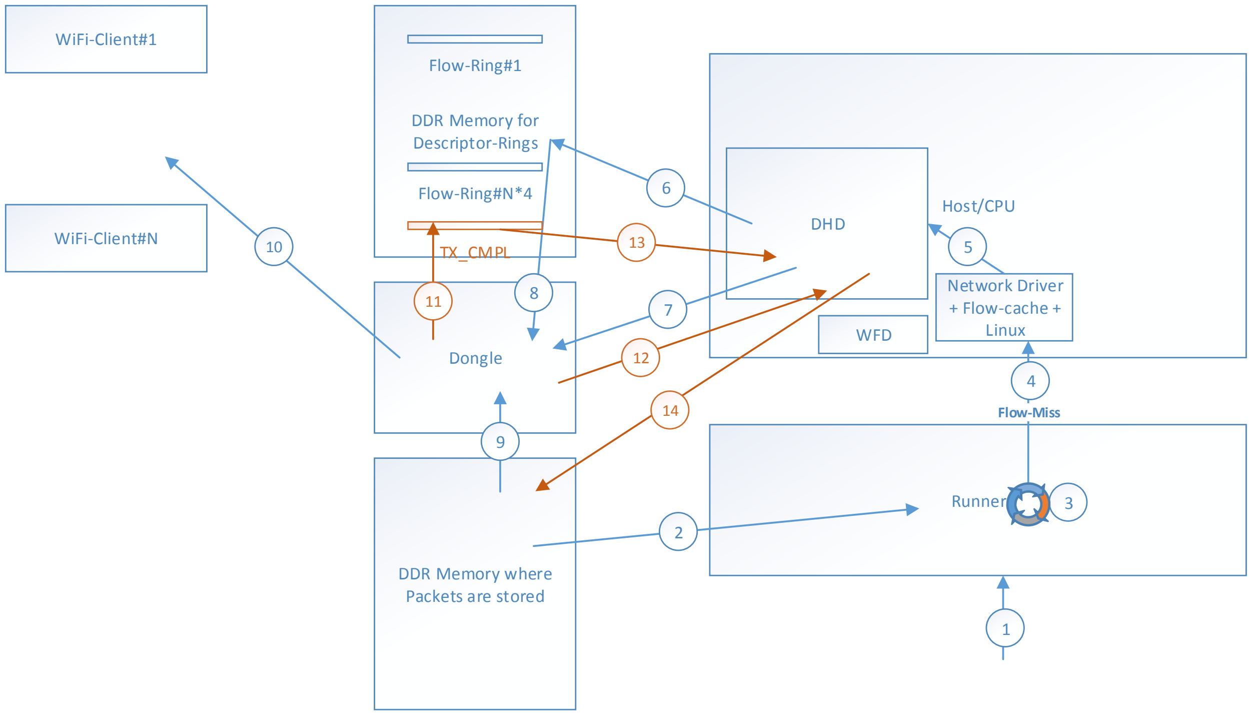


1. DHD allocates packet buffers to receive the packet from Dongle.
2. DHD puts the Receive packet buffer information in RX_POST Descriptor ring.
3. Dongle fetches the receive packet buffer information from the RX_POST descriptor ring.
4. Dongle receives the packet from WiFi-Client into the packet buffer.
5. Dongle processes the packet.
6. Dongle puts the information about received packet in RX_CMPL descriptor ring.
7. Dongle notifies the DHD about packet receive.
8. DHD fetches the descriptor from the RX_CMPL ring.
9. DHD fetches the packet from the packet buffers.
10. DHD forwards the packet to Network Driver (either through Linux or flow-cache).
11. Network Driver gives the packet to Runner for transmission.
12. Runner transmits the packet towards LAN/WAN.
13. Runner sends the packet buffer back to Network Driver to free.
14. Network Driver frees up the buffer back to Packet buffer memory.

4.3.2 PACKET TRANSMISSION (DHD TO DONGLE)

Packet transmission towards dongle can take two path based on flow being in runner or not.

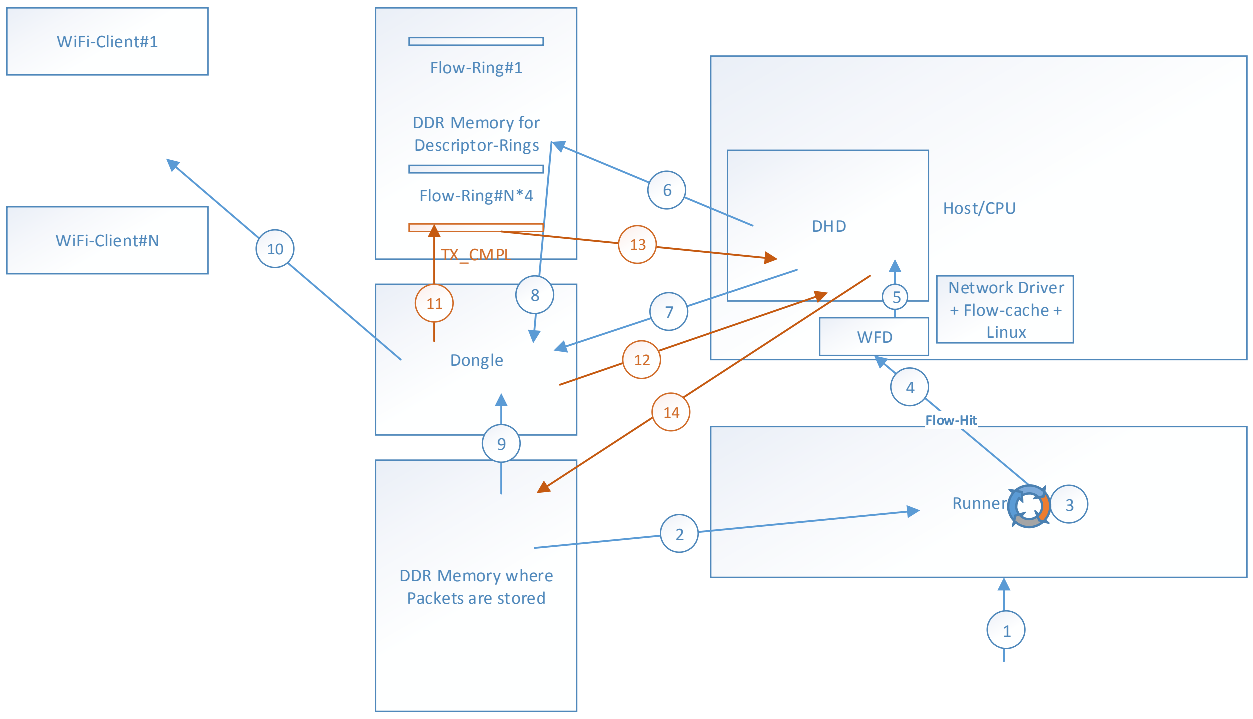
4.3.2.1 Transmit Flow-miss Case



1. Packet is received by Runner from LAN/WAN port.
2. Packet is received in the DDR memory buffer.
3. Runner processes the packets and makes the forwarding decision (flow-miss).
4. Runner sends the packet to Host CPU Network driver.
5. Network driver receives the packet and forwards it to DHD Driver (either through Linux Kernel, or Flow-cache).
6. DHD updates the TX_POST descriptor in the flow-ring with required information.
7. DHD notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the DHD about the TX_CMPL ring update.
13. DHD takes the packet buffer information from the TX_CMPL ring.
14. DHD frees the packet buffer back to the packet pool (from wherever it came from).

4.3.2.2 Transmit Flow-hit Case

Difference between Flow-miss/hit cases is highlighted below.



1. Packet is received by Runner from LAN/WAN port.
2. Packet is stored in the DDR memory.
3. Runner processes the packets and makes the forwarding decision (flow-hit).
4. Runner sends the packet to Host CPU WFD.
5. WFD receives the packet and forwards it to DHD Driver.
6. DHD updates the TX_POST descriptor in the flow-ring with required information.
7. DHD notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the DHD about the TX_CMPL ring update.
13. DHD take the packet buffer information from the TX_CMPL ring.
14. DHD frees the packet buffer back to the packet pool (from wherever it came from).

4.4 DHD OFFLOAD PATH

As mentioned above, interface with Dongle remains the same irrespective of offload or host/CPU DHD path.

DHD offload provides a way such that descriptor rings could be managed by Runner (instead of DHD). This capability enables Runner to directly send/receive the packets to/from Dongle without DHD/Host involvement.

Allocations of descriptor rings remain the responsibility of DHD and the required information is passed to Runner to manage those rings.

4.4.1 RING ALLOCATIONS

When DHD Offload feature is enabled in the build profile, **receive path from dongle is always handled by Runner**. Details about each ring type management are below:

- RX_CMPL: Always managed by Runner.
- RX_POST: Always managed by Runner.
- **TX_POST/Flow-Rings**: Allocation and Management of flow-rings (used to transmit data packets towards Dongle) is determined by the memory reservation done at system boot time (information that is supplied by the boot loader – see section "Memory Reservation").
    - For the remainder of this document and better understanding, following nomenclature will be used:
        - Number of flow-rings managed by Runner = "N"
        - Number of flow-rings managed by DHD = "M"
        - This means the total number of flow-rings required to support all the stations on a Dongle is "N+M".
    - If the memory isn't enough to offload all flow-rings, DHD will allocate some flow-rings "N" in the reserved memory for Runner to manage and will allocate remaining "M" from Linux to be managed by DHD.
    - If there is enough memory reserved to offload all the flow-rings to Runner, DHD will allocate all flow-rings from the reserved memory and hand over the management to Runner (i.e. M=0).
    - If there is NO memory reserved for flow-rings, DHD will allocate all flow-rings from Linux memory pool (as it would do in offload disabled builds) and manage the flow-rings as well (i.e. N=0).
- TX_CMPL: This ring is managed by Runner if N > 0, otherwise managed by DHD.

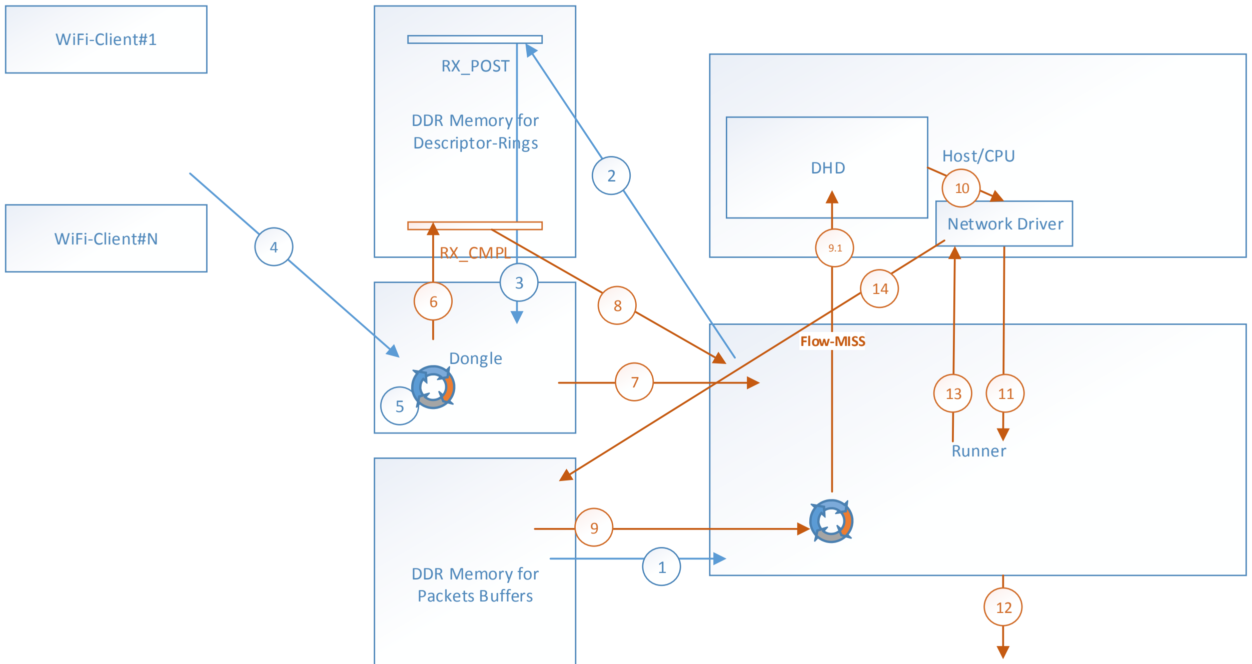
4.4.2 PACKET RECEIVE (DONGLE TO RUNNER)

When DHD offload is enabled, all packet from Dongle are always received by the Runner (accelerated or not).

4.4.2.1 Receive Flow-Miss case

Flow-miss path (packet trapped to DHD for processing) will occur whenever runner does not have the flow to process/forward the packet. This mechanism remain same as any other packet acceleration done by Runner for other interfaces (LAN/WAN).

The changes from the DHD receive path are highlighted in the text description to emphasis the fact that Dongle does not care if the packet is received by the Runner or DHD.

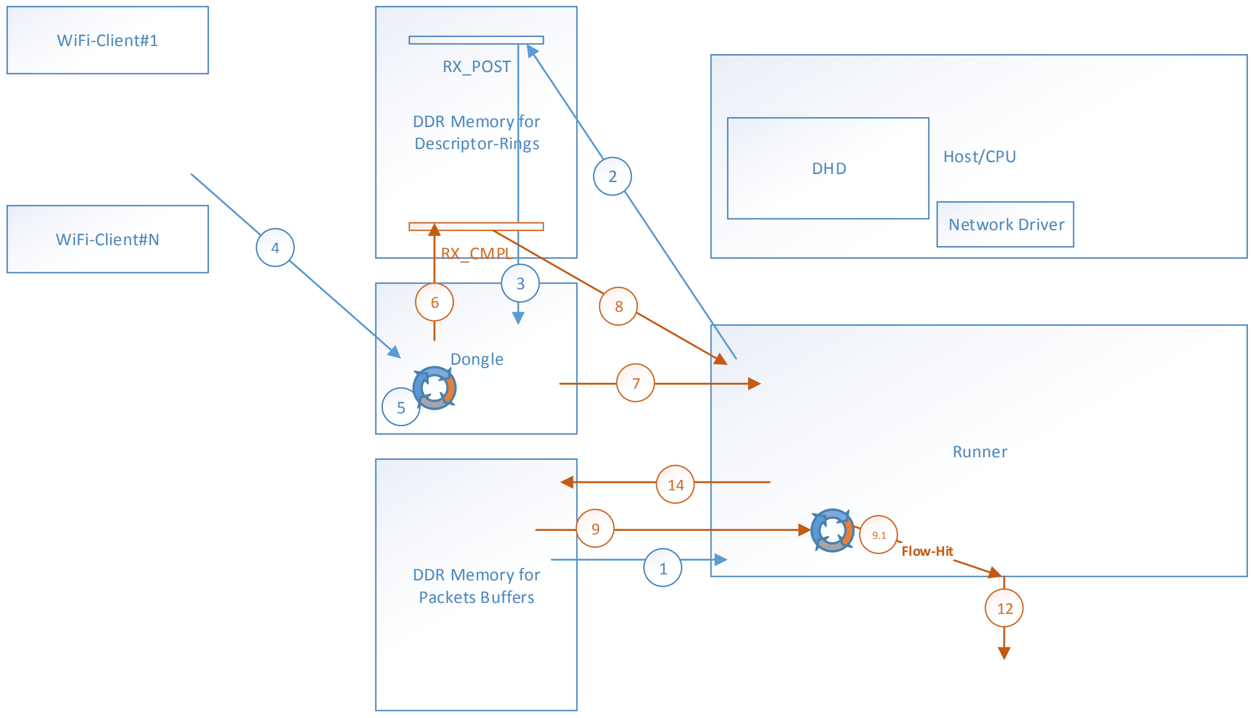


1. Runner allocates a packet buffer to receive the packet from Dongle.
2. Runner puts the Receive packet buffer information in RX_POST Descriptor ring.
3. Dongle fetches the receive packet buffer information from the RX_POST descriptor ring.
4. Dongle receives the packet from WiFi-Client into the packet buffer.
5. Dongle processes the packet.
6. Dongle puts the information about received packet in RX_CMPL descriptor ring.
7. Dongle notifies the Runner about packet.
8. Runner fetches the descriptor from the RX_CMPL ring.
9. Runner fetches the packet from the packet buffers.
   9.1 Runner has no flow for this packet → Flow-miss → packet trapped to DHD
10. DHD forwards the packet to Network Driver (either through Linux or flow-cache).
11. Network Driver gives the packet to Runner for transmission.
12. Runner transmits the packet towards LAN/WAN.
13. Runner sends the packet buffer back to Network Driver to free.
14. Network Driver frees up the buffer back to Packet buffer memory.

4.4.2.2 Receive Flow-hit case

Flow-hit path will occur whenever runner has the flow to process/forward the packet. This mechanism remain same as any other packet acceleration done by Runner for other interfaces (LAN/WAN).

Flows are pushed to Runner by flow-cache the same way as done for any other interface. The changes from the "Flow-miss" receive path are highlighted in the text description below.



1. Runner allocates a packet buffer to receive the packet from Dongle.
2. Runner puts the Receive packet buffer information in RX_POST Descriptor ring.
3. Dongle fetches the receive packet buffer information from the RX_POST descriptor ring.
4. Dongle receives the packet from WiFi-Client into the packet buffer.
5. Dongle processes the packet.
6. Dongle puts the information about received packet in RX_CMPL descriptor ring.
7. Dongle notifies the Runner about packet.
8. Runner fetches the descriptor from the RX_CMPL ring.
9. Runner fetches the packet from the packet buffers.
   9.1 Runner has flow for this packet → Flow-hit → packet forwarded to egress interface.

~~10. DHD forwards the packet to Network Driver (either through Linux or flow-cache).~~
~~11. Network Driver gives the packet to Runner for transmission.~~
12. Runner transmits the packet towards LAN/WAN.
~~13. Runner sends the packet buffer back to Network Driver to free.~~
14. Runner frees up the buffer back to Packet buffer memory.

4.4.3 PACKET TRANSMISSION (RUNNER/DHD TO DONGLE)

Packet transmission path in DHD Offload enabled builds can take three different paths depending upon Flow-ring allocation and flow-hit/miss in Runner (refer to section "Ring Allocations").

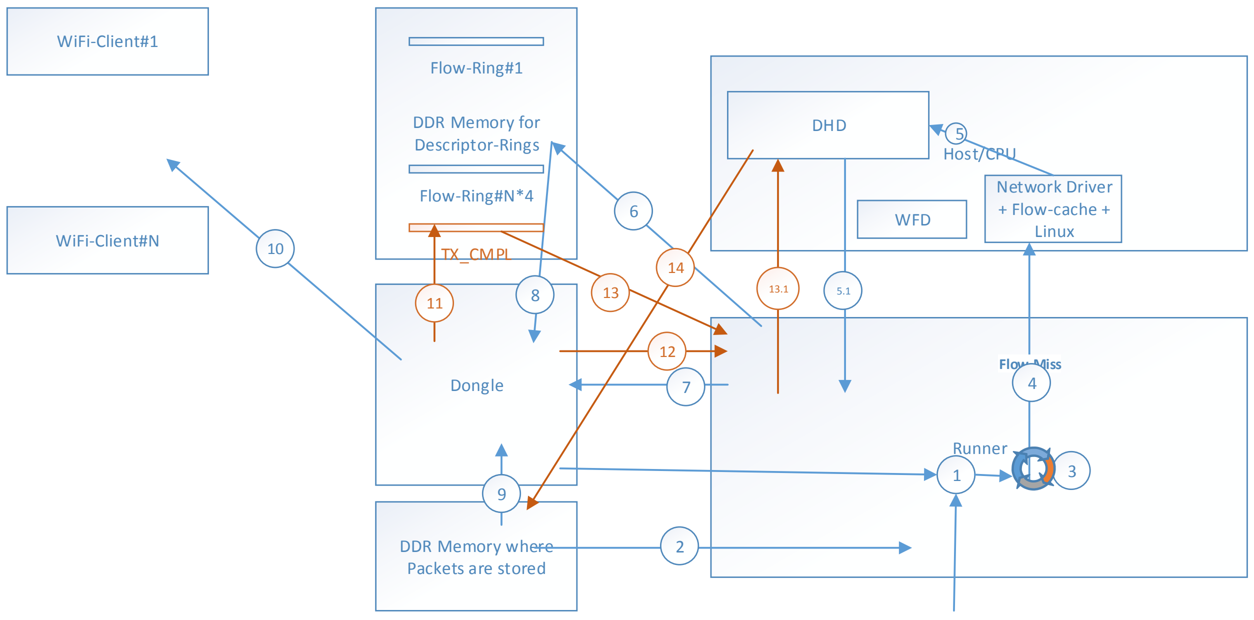
- Flow-ring managed by Runner (one among those "N" flow-rings): All traffic towards this flow-ring must go through runner.
- Flow-ring managed by DHD (one among those "M" flow-rings): All traffic towards this flow-ring must go through DHD.

4.4.3.1 Runner Managed Flow-rings

This case illustrates the data path flow for two scenarios (1) flow-miss, and (2) flow-hit, when the Flow-ring is managed by the Runner.

4.4.3.1.1 Flow-Miss Case

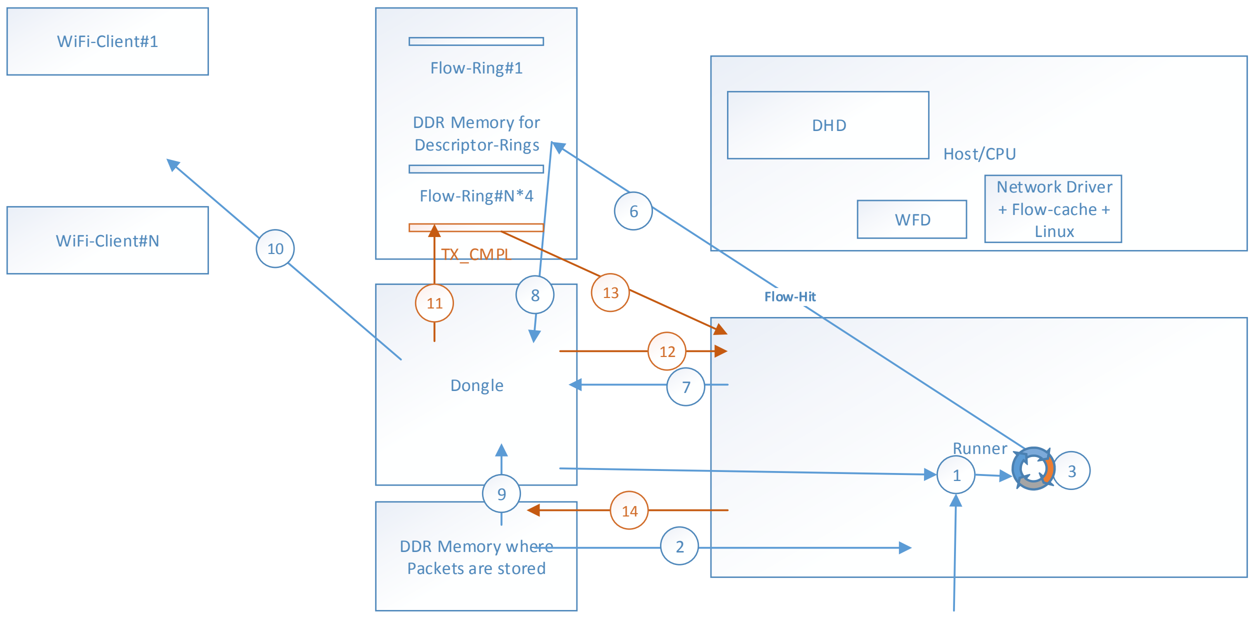
Differences from non-offload flow-miss case are highlighted below.



1. Packet is received by Runner from LAN/WAN port or WLAN-Dongle.
2. Packet is received in the DDR memory buffer.
3. Runner processes the packets and makes the forwarding decision (flow-miss).
4. Runner sends the packet to Host CPU Network driver.
5. Network driver receives the packet and forwards it to DHD Driver (either through Linux Kernel, or Flow-cache).
    5.1. DHD determines the flow-ring is managed by Runner, so forwards the packet to Runner for transmission.
6. Runner updates the TX_POST descriptor in the flow-ring with required information.
7. Runner notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the Runner about the TX_CMPL ring update.
13. Runner takes the packet buffer information from the TX_CMPL ring.
    13.1 Runner sends the TX_CMPL information to DHD.
14. DHD frees the packet buffer back to the packet pool (from wherever it came from).

4.4.3.1.2 Flow-Hit Case

Differences from non-offload flow-hit case are highlighted below.



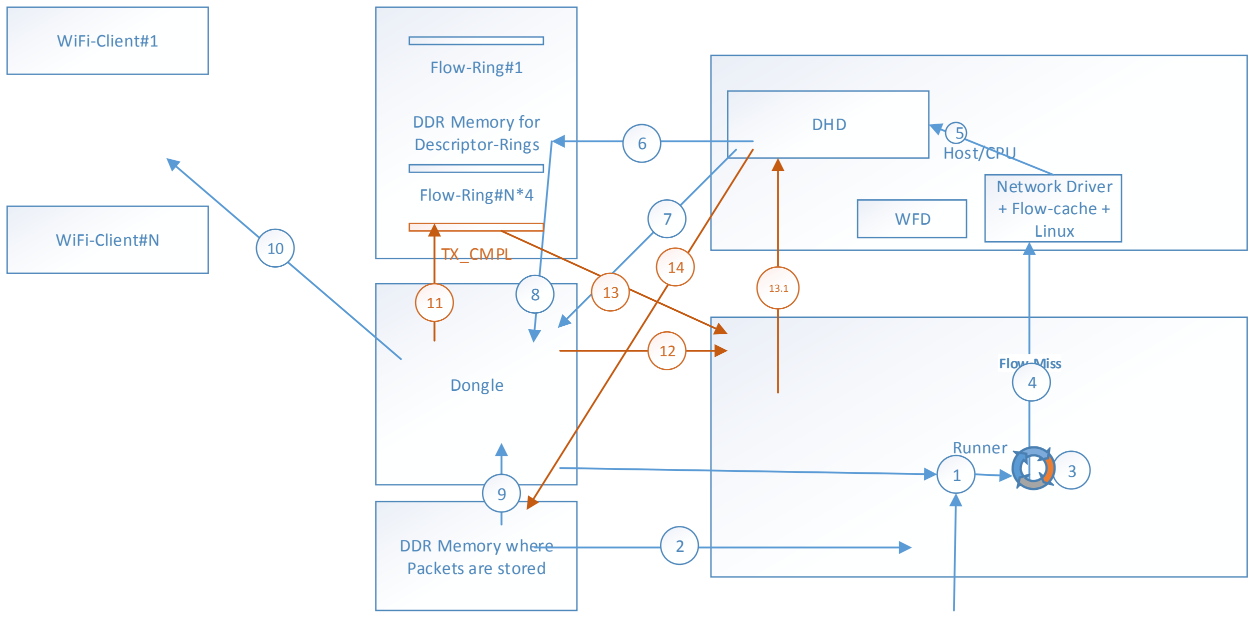
1. Packet is received by Runner from LAN/WAN port or WLAN ~~Dongle~~.
2. Packet is stored in the DDR memory.
3. Runner processes the packet and makes the forwarding decision (flow-hit).
4. ~~Runner sends the packet to Host CPU WFD.~~
5. ~~WFD receives the packet and forwards it to DHD Driver.~~
6. Runner updates the TX_POST descriptor in the flow-ring with required information.
7. Runner notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the Runner about the TX_CMPL ring update.
13. Runner takes the packet buffer information from the TX_CMPL ring.
14. Runner frees the packet buffer back to the packet pool (from wherever it came from).

4.4.3.2 DHD Managed Flow-Rings

This case illustrates the data path flow for two scenarios (1) flow-miss, and (2) flow-hit, when the Flow-ring, where packet needs to be transmitted, is managed by the DHD.

4.4.3.2.1 Flow-miss Case (N > 0)

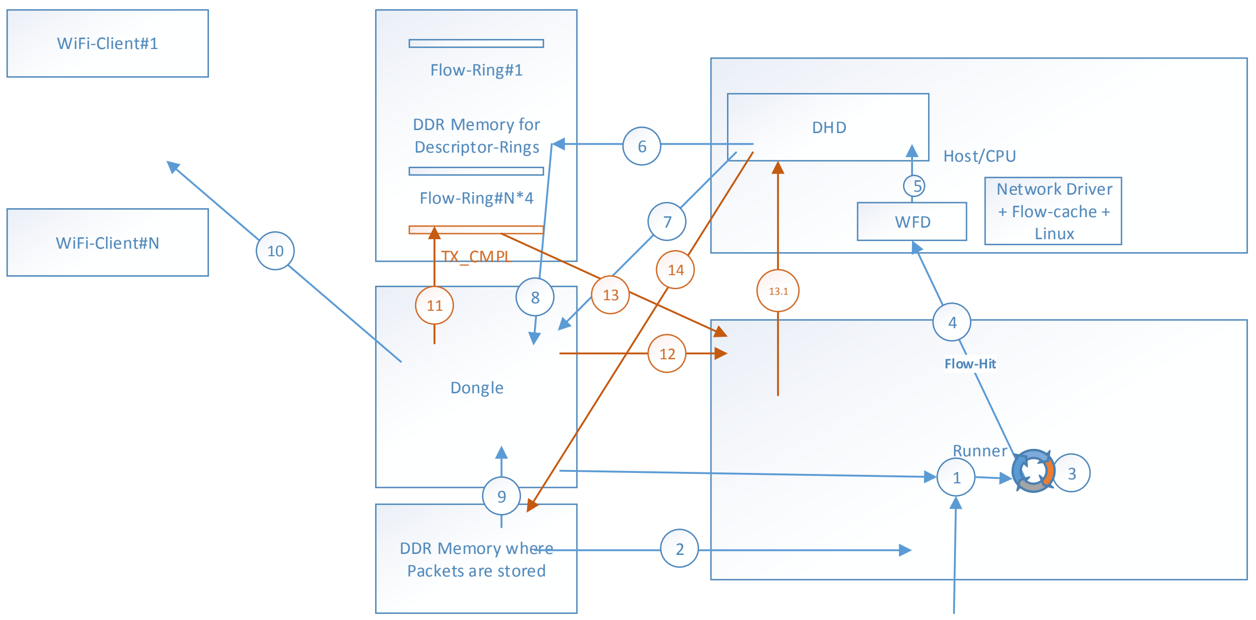
Differences from non-offload flow-miss scenario are highlighted below.



1. Packet is received by Runner from LAN/WAN port or WLAN-Dongle.
2. Packet is received in the DDR memory buffer.
3. Runner processes the packets and makes the forwarding decision (flow-miss).
4. Runner sends the packet to Host CPU Network driver.
5. Network driver receives the packet and forwards it to DHD Driver (either through Linux Kernel, or Flow-cache).
6. DHD updates the TX_POST descriptor in the flow-ring with required information.
7. DHD notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the Runner about the TX_CMPL ring update.
13. Runner takes the packet buffer information from the TX_CMPL ring.
    13.1 Runner sends the TX_CMPL information to DHD.
14. DHD frees the packet buffer back to the packet pool (from wherever it came from).

4.4.3.2.2 Flow-Hit Case (N>0)

Differences from non-offload flow-hit scenario are highlighted below.



1. Packet is received by Runner from LAN/WAN port or WLAN-Dongle.
2. Packet is stored in the DDR memory.
3. Runner processes the packets and makes the forwarding decision (flow-hit).
4. Runner sends the packet to Host CPU WFD.
5. WFD receives the packet and forwards it to DHD Driver.
6. DHD updates the TX_POST descriptor in the flow-ring with required information.
7. DHD notifies the Dongle about the packet.
8. Dongle reads the TX_POST descriptor from flow-ring and gets the required information.
9. Dongle grabs the packets from DDR.
10. Dongle transmits the packets towards Wifi-Client.
11. Dongle updates the descriptor with transmitted packet buffer information in TX_CMPL Ring.
12. Dongle notifies the Runner about the TX_CMPL ring update.
13. DHD take the packet buffer information from the TX_CMPL ring.
    13.1 Runner sends the TX_CMPL information to DHD.
14. DHD frees the packet buffer back to the packet pool (from wherever it came from).

4.4.3.2.3 No Runner Managed Flow-rings

When runner is not managing any flow-rings (N=0; meaning DHD memory in boot loader was set to 0), TX_CMPL ring is also managed by DHD.

In such case, the data path flow would be same as “DHD Managed Ring” except that TX_CMPL signal will be handled by DHD directly (instead of Runner).

4.5 IPTV MULTICAST HANDLING

Multicast traffic acceleration/offload takes the same data path as unicast i.e. based on the destination flow-ring either it takes the WFD (flow-ring managed by DHD) or direct Runner (flow-ring managed by Runner) path.

When multiple clients join the same stream and the flow-rings are managed by DHD, Runner only forwards one packet per SSID to DHD and the replication to all the clients is done by DHD.

Multicast flows are configured (not learned by default) in flow-cache/Runner, so multicast traffic does not go through Linux iptable/ebtable rules/classification. By default all WLAN multicast traffic will be mapped to Video Access Category Flow-ring (AC_VI).

IPTV Multicast traffic priority can be changed by setting system level global multicast precedence:

1. This can be set using WebGUI and is displayed in "cat proc/net/igmp_snooping" output.
2. When global multicast precedence is set, it applies to all IPTV multicast traffic (including LAN clients as well).

Notes:

1. Multicast acceleration through Runner is only supported for WAN to LAN/WLAN traffic. LAN/WLAN to LAN/WLAN multicast acceleration is done by flow-cache in Host/CPU only.
2. If WMF (Wireless Multicast Forwarding) feature is NOT enabled for a given SSID/BSS, all multicast traffic towards clients on that SSID/BSS will go through WFD acceleration path.

4.6 INTER/INTRA BSS AND INTER RADIO TRAFFIC HANDLING

Inter/Intra BSS and Inter Radio traffic will always be accelerated using one of the following acceleration paths.

- ➢ Non-offload builds
  - o Flow-cache acceleration applies.
- ➢ Offload builds
  - o Runner will accelerate all the **received** traffic from Dongle as below:
    - ▪ Runner will forward the traffic through WFD path if TX flow-ring is DHD managed or destination is NIC WLAN.
    - ▪ Runner will directly forward the traffic (without host involvement) if TX flow-ring is managed by Runner.
  - o For all other scenarios Inter/Intra BSS and Inter Radio traffic will be learned & accelerated by flow-cache.

4.7 FLOW-RING ALLOCATION CONTROLS

The following controls are provided for configuring the N+M flow rings:

- Reserved memory (through boot loader)
- Flow-Ring Profile
  - Flow ring size (max items)
  - Flow ring allocation priority
  - Flow ring selection policy

All these controls are set per dongle radio.

Note: It is recommended to select proper flow-ring controls based on product requirements but carefully consider the impact and implications before changing the default values.

4.7.1 RADIO NUMBER ASSIGNMENT

During radio number assignment, dongle cards are scanned first. Radio numbers assignment start from 0 and incremented for each additional dongle card present.

Note: NIC cards are scanned after dongle cards.

Below is the scan sequence

- PCIe port #0
  - Port#0 … Port#N, if connected through a PCIe bridge
- PCIe port #1 (if present)
  - Port#0 … Port#N, if connected through a PCIe bridge

WL interface number will be same as DHD dongle radio number.

4.7.2 RESERVED MEMORY

Refer to section “Memory Reservation”.

4.7.3 FLOW RING PROFILE

Flow ring profile controls the max items and allocation weight/priority for each WLAN access category (AC) to determine if the created flow-ring should be managed by Runner or DHD.

Max items control determines how much memory is needed for flow rings. Allocation weight will be used to allocate runner accelerated flow rings to an access category.

There will be one flow ring profile per radio. Profile information is stored in Persistent Scratch Pad (PSP) memory and accessed using DHD command line application (dhdctl).

A profile is specified using profile identifier. DHD supports five built-in fixed profiles and three (one per radio) user defined profiles. Each profile will have the following format

<id> <weight>:<max_items> <weight>:<max_items> <weight>:<max_items> …

"<id>" = Refers to Profile identifier

"<weight>:<max_items>" = weight and max items combination is specified per access category (AC_BK, AC_BE, AC_VI, AC_VO) and for Broadcast/Multicast (BCMC) flow-ring per interface.

Based on the allocation weight DHD starts allocating runner managed flow rings to a particular access category. Any time during the allocation process, if there is no more reserved memory available, remaining flow rings are DHD managed.

Allocation weight specifies the number of flow-rings allocated during each AC (Access Category) scan.

Weight	Description
-1	Highest weight. All flow-rings are allocated at once.
0	Reserved for future use.
1	One flow-ring is allocated for each AC per scan. The scan is repeated until all required flow-rings are allocated.
2..N	"N" flow-ring is allocated for each AC per scan. The scan is repeated until all required flow-rings are allocated.

Field	values	Value description	Default values
<id>	0-6	0: Radio#1 user defined profile	0 1:1024 -1:2048 -1:1024 -1:512 1:512
		1: Radio#2 user defined profile	1 1:1024 -1:2048 -1:1024 -1:512 1:512
		2: Radio#3 user defined profile	2 1:1024 -1:2048 -1:1024 -1:512 1:512
		3: built-in profile (AC_BE, AC_VI priority)	3 1:1024 -1:2048 -1:1024 1:512 1:512
		4: built-in profile (each AC allocate first)	4 -1:1024 -1:2048 -1:1024 -1:512 1:512

		5: built-in profile (all AC same priority) 6: built-in profile (ascending AC priority) 7:built-in profile (all flow-rings same size, and priority)	5 1:1024 1:2048 1:1024 1:512 1:512 6 1:1024 2:2048 4:1024 8:512 1:512 7 1:2048 1:2048 1:2048 1:2048 1:2048
<weight>	-1, 1, 2, ….N	-1: Allocate all flow rings first 1-N: Allocate 'N' flow rings per AC per scan.	Based on profile id
<max_items>	128…..N	N: size of ring for that AC	AC_BK:1024, AC_BE: 2048, AC_VI:1024, AC_VO:512, BCMC:512

Flow-Ring Profile User Interface

"dhdctl" provides IOCTL's to list and set the flow ring profile for each radio. If no profile is stored in PSP, DHD uses a default built-in profile id 3.

To get the list of profiles available and current active profile use below command shown. The current active profile of the radio is the id with "*"

```
# dhd -i <IF#> flowring_profile
[id] [ ac_bk ] [ ac_be ] [ ac_vi ] [ ac_vo ] [ ac_bcmc]
  0   01:1024   -1:2048   -1:1024   -1:0512   01:0512
  1   01:1024   -1:2048   -1:1024   -1:0512   01:0512
  2   01:1024   -1:2048   -1:1024   -1:0512   01:0512
  3   01:1024   -1:2048   -1:1024   -1:0512   01:0512
 *4   -1:1024   -1:2048   -1:1024   -1:0512   01:0512
  5   01:1024   01:2048   01:1024   01:0512   01:0512
  6   01:1024   02:2048   04:1024   08:0512   01:0512
  7   01:2048   01:2048   01:2048   01:2048   01:2048

Where
<IF#> wl interface of radio (wl0, wl1, wl2)
```

Example to change the user defined profile settings

```
# dhd -i <IF#> flowring_profile <p> 1:1024 1:2048 1:1024 1:512 1:512

Where
<IF#> wl interface of radio (wl0, wl1, wl2)
<p>   user profile id (0-2)
```

A profile can be set active using a profile id or user profile id with <weight><items> settings. Below is an example to set the profile using profile id (no profile parameters)

```
# dhd -i <IF#> flowring_profile <p>

Where
<IF#>  wl interface of radio (wl0, wl1, wl2)
<p>    profile id (0-7)
```

[illegible]

4.7.4 FLOW-RING POLICY CONTROL

Flow ring policy is used to select runner offload flow ring or DHD managed flow ring at run time. This control is provided mainly to test the N+M flow ring feature and for tuning purpose. The following table shows the available flow ring selection policies. At a time one of the policies will be active.

Policy Name	Policy ID	values	Value description	Comments
Global	0	0-1	0: Prefer DHD flow ring first 1: Prefer Runner offloaded flow ring first	This is the default profile id with a default value of 1
Intfidx	1	0..N N < 16	Prefer Runner offloaded flow rings for all stations connected to interface index <= n	43602 dongle N<4 4366 dongle N<16
Clients	2	0..N N < 128	Prefer Runner offloaded flow rings for all stations until N clients	43602 dongle N<32 4366 dongle N<128
aclist	3	<ac>:<0/1>	Prefer DHD/Runner for each AC specified in the list	<ac> values 0:ac_bk 1: ac_be 2:ac_vi 3: ac_vo 4:ac_bcmc
Maclist	4	<mac>	Prefer Runner flow rings for all the stations from the mac address list	Mac address format `xx:xx:xx:xx:xx:xx` Up to 4 mac address can be specified
d11ac	5	0/1	1: prefer Runner flow rings for all 11AC stations, others DHD flow rings 0: prefer DHD flow rings for all 11AC stations, others Runner flow rings	Not available now. For future implementation

If no policy is stored in PSP, DHD uses a default user policy id#0.

“dhdctl” provides IOCTL’s to list and set the flow ring policy of each radio.

To get the list of policies available and current active policy, use below command shown. The current active policy of the radio is the id in the list with “*”

```
# dhd -i <IF#> flowring_policy
[ Name  ] [id] [Policy]
  global    0
 intfidx    1
 clients    2
  aclist    3
 maclist   *4   00:90:4c:0f:50:93
   d11ac    5

where
<IF#> wl interface of radio (wl0, wl1, wl2)
```

Example to set the global policy for all runner flow rings

```
# dhd -i <IF#> flowring_policy 0 1

Where
<IF#> wl interface of radio (wl0, wl1, wl2)
```

Example to set the mac address list policy with 3 stations

```
# dhd -i <IF#> flowring_policy 4 00:90:4c:0f:50:91 00:90:4c:0f:50:92 00:90:4c:0f:50:93

Where
<IF#> wl interface of radio (wl0, wl1, wl2)
```

Example to set the ac list policy BE, VI to select runner offload and others to select DHD

```
# dhd -i <IF#> flowring_policy 3 0:0 1:1 2:1 3:0

Where
<IF#> wl interface of radio (wl0, wl1, wl2)
```

NOTE: New Setting is effective for all clients that connect after the set. It’s recommended to disconnect all the clients before the setting or reboot the system after the setting

4.8 ACCELERATED DATA PATH OVERVIEW

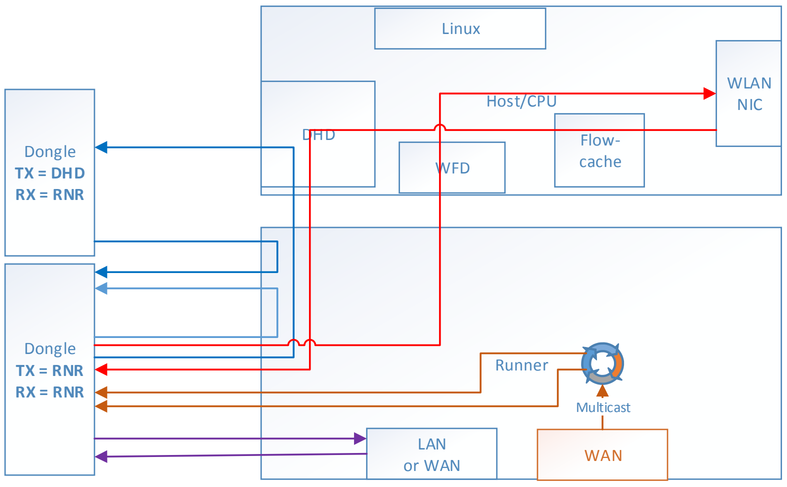
Shown below are all possible **accelerated** data path flows for Dongle when offload is enabled in the build. When arrows are traces from RX to TX, it shows what path packet will take (the module boxes it touches) for that kind of traffic scenario.

Following scenarios are presented:

1. Downstream Multicast with replication.
2. LAN/WAN to/from Dongle
3. Dongle to/from Dongle
4. Dongle to/from NIC

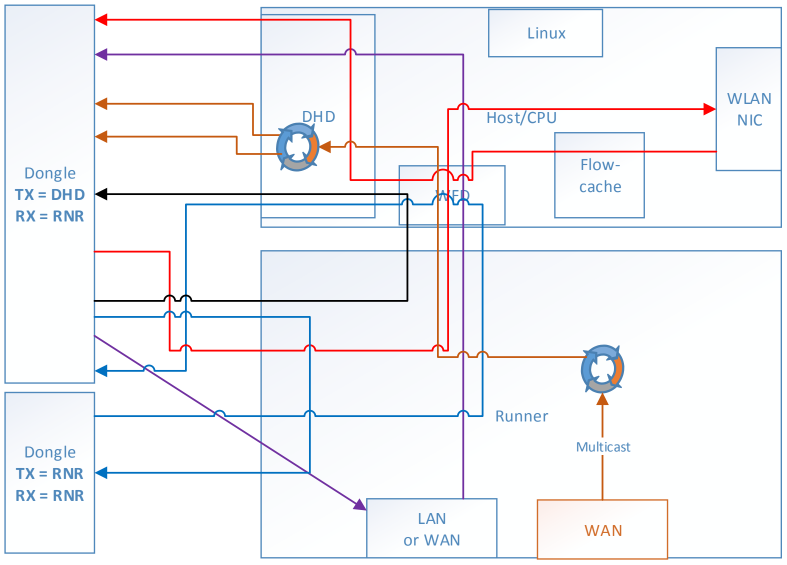
4.8.1 TRAFFIC FLOWS TO/FROM FULLY OFFLOADED DONGLE

This case shows the traffic flow when both RX and TX paths from/to Dongle are going through Runner.



4.8.2 TRAFFIC FLOWS TO/FROM RX-ONLY OFFLOADED DONGLE

This case shows the traffic flow when only the RX path is going through Runner and TX is going through DHD. This scenario will occur if user did not allocate any memory for that Dongle during system boot up (through boot loader).



4.9 DHD OFFLOAD DEBUGGING

While debugging any issues related to the DHD Offload, user should look for the following information carefully:

1. Check the amount of memory allocated through the boot loader (Refer to "Memory Reservation" section for details).

2. Make sure the correct memory is allocated during system boot (as configured in the boot loader). Look for following debug prints during system boot to find out the actual memory allocated.

```
creating a MT_MEMORY_NONCACHED device at physical address of 0x0d200000 to
virtual address at 0xcd200000 with size of 0x2c00000 byte for RDPA tm

creating a MT_MEMORY_NONCACHED device at physical address of 0x0c000000 to
virtual address at 0xcc000000 with size of 0xe00000 byte for DHD dhd0

creating a MT_MEMORY_NONCACHED device at physical address of 0x0b800000 to
virtual address at 0xcb800000 with size of 0x700000 byte for DHD dhd1
```

3. Get the output of following commands:

   "dhdctl –i wl0 dump"
   "dhdctl –i wl1 dump"

```
…
…
…

DHD Runner:
  Status      : radio#  0 tx_Offl 1  rx_Offl 1
  Profile     : prfl_id 3 id_valu  01:1024   -1:2048   -1:1024   -1:0512   01:0512
  Policy      : plcy_id 0 id_valu 1 (HW)
  tx_flowring: [ac_bk] sw 0 hw 64 [ac_be] sw 0 hw 64 [ac_vi] sw 0 hw 64 [ac_vo] sw 0 hw 64 [bc_mc] sw 0 hw
8
  h2r_notif   : tx_post 74 rx_cmpl 5 tx_cmpl 22
  r2h_req     : rx_cmpl 11 tx_cmpl 0 wk_dngl 85

CtrlPost: RD 37 WR 37
CtrlCpl:  RD 85 WR 85
RxPost:   RBP 0 RD 576 WR 10
RxCpl:    RD 11 WR 11
TxCpl:    RD 73 WR 74
active_tx_count 0  pktidmap_avail -1  rxbufpost 0
dhd cumm_ctr 0
Num: HW Flow If Prio :Dest_MacAddress: Qlen CLen L2CLen Pkts  Overflows   RD   WR  Acked tossed noack
  0.  1    2  0    0 33:33:00:00:00:16    0    0  0        0         0     69   69    NA     NA    NA
  1.  1  202  0    0 ac:9e:17:45:a1:ac    0    0  0        0         0      5    5    NA     NA    NA
```

Above dump provides the configuration information about:

- Flow-Ring Profile being used.
- Policy being used.
- RX_POST, RX_CMPL, TX_CMPL and TX_POST ring current Read/RD & Write/WR pointers.
- Provides information about the Flow-Rings as below:
  - Flow-Ring-ID "Flow" (e.g. 202)

- Is the flow-ring managed by HW (i.e. Runner) or SW (i.e. DHD) – check the “HW” column.
- Priority (or Access Category) of the flow-ring – check the “Prio” column.
- MAC Address of WLAN Client using that flow-ring – check “Dest MacAddress: column.

4. If the issue is related to unicast acceleration, dump the runner unicast flows.

5. If the issue is related to multicast acceleration, dump the running multicast flows along with WLAN multicast flow information using command “bs /b/e wlan_mcast”