目录

[General description 2](#_Toc40972357)

[缩写词/术语 2](#_Toc40972358)

[TX datapath 4](#_Toc40972359)

[RX datapath 10](#_Toc40972360)

|  |  |  |
| --- | --- | --- |
| 作者 | 蓝麒 | |
| 版本信息/修改记录 |  | |
| 版本 | 描述 |
| 2020-05-22 | 1.0 | 初始版本 |
|  |  |  |

## General description

以文字框图形式描述 datapath，为设计硬件加速提供指导。 分析需要尝试配合 802.11 spec 中 MAC data plane architecture 框图。分析内容包含从IP stack 到 WMAC 的报文发送，以及无线报文从WMAC 到IP Stack. 分析内容以语言进行逻辑描述而非代码描述，并适当穿插针对 设计硬件加速 的思考点。具体硬件加速需要实施的内容取决于合作方的实现方案，而本文针对整体平台无关的设计上具有参考意义 。

* 分析 SW Datapath 操作比较耗时的操作，可用硬件加速替代
* 哪些特殊帧需要作为HW Datapath 的例外，交付给CPU 处理
* 硬件加速可能破坏潜在的一些功能
* 硬件需要执行哪些功能模块

浅绿色框内是对软件实现的描述。灰色框是针对硬件加速设计的思考讨论。

## 缩写词/术语

|  |  |
| --- | --- |
| DA | Destination *MAC* Address |
| SA | Source *MAC* Address |
| dip, sip | Destination IP address, source IP address |
| Port | 在本文需要根据前后上下文，可能指物理的或虚拟的端口，也可能是TCP/UDP port. |
| VAP | virtual AP. 软件上的虚拟AP，和 multi-BSS 相关  有个OS struct net\_device{} 与之对应。 |
| Node | 代表一个station. 对端station成为 peer node. 本地成为 local node. 每个VAP 可以看做一个station, 即每个VAP 一个local node。 |
| TID | Traffic identification. 支持QoS 的station(node) 具有16个TID, 用以区分steam 的优先级。某些实现中会额外添加第17个TID，用于发送管理报文。 |
| Radio device | 一个Radio设备工作于2.4GHz band 或者 5GHz band。 可支持多个 VAP。有个OS struct net\_device{} 与之对应。 |
| 高优先级帧 | EAPOL, DHCP, ARP, QoSNULL |
| WMAC | Wireless MAC, 属于硬件范畴 |
| 异步/同步 | 本文中同步是指 针对一个报文发送的过程的顺序执行。异步是指针对一个报文发送的过程发生中断。中断往往是由于node 状态、由QoS 引入的queue 或者AMSDU/AMPDU 聚合导致暂缓发送。 |
| DPI | Data path interface  硬件提供给软件的接口，用于灵活设置一些数据报文的datapath. 例如某些bit 匹配目标模式，将报文送至CPU处理 或者丢弃。 |
| packet/frame | 本文没有区分 packet, frame，统一理解为 报文。本文也混用报文，封包这种用语。 |
| NAWDS | Non-association WDS. 是比较早期流行的一种无线桥模式。目前比较多用无线桥是 Repeater 模式。 |
| 组播 | Multicast. 也称多播。本文中组播包含了广播，如有排除广播的情况需特别指出。 |
| Datapath | 数据报文从设备入口 到设备出口需要经过许多处理节点。所有这些节点连接构成的流程称为datapath. |
| SW datapath | 针对单个数据报文的处理需要CPU参与的datapath |
| HW datapath | 针对单个数据报文的处理不需要CPU参与的datapath |

## TX datapath

TX datapath 大致分了三个阶段：

阶段一： bridge 或说IP stack 转发到VAP

阶段二： VAP 中寻找node，将skb->dev 切换成 radio device

阶段三： Radio device 层的发包处理。准备好 DMA descriptor 并提交给 WMAC.

在报文发送路径上，存在一些原因使得报文发送过程中存在异步。

* 1. 从OS statck 开始处理
     1. 设备状态判断。如果设备 （VAP/Radio device）处于 down 的状态，那么丢弃封包而停止发送。
     2. 如果当前channel 遇到 radar ，那么丢包而停止发送

|  |
| --- |
| 硬件加速模块：驱动能够设置硬件，暂停将报文推送到PCIe 总线。暂停原因不需要硬件提供可设置的flag，由软件提供。 |

* + 1. 计算skb head 部分空间是否满足驱动要求，如果不满足则进行扩展

|  |
| --- |
| 硬件加速模块：在设计buffer 时要充分考虑所处理的 报文size 范围。 因而不需要考虑软件这种动态扩展。 |

* + 1. 处理 WNM Traffic filter service, 如果不是station 所指定的packet，那么丢包

|  |
| --- |
| 硬件加速模块：属于WiFi范畴里 比较高层的feature. 如果需要支持WNM的这个feature并且该feature的执行存在于SoC 端, 在 硬件加速模块设计处理上可能会比较棘手。 |

* + 1. 处理 802.11v ARP Proxy.

驱动检测并处理ARP, DHCP, 还有IPv6 的 ICMP. 如果AP驱动介入适当的处理完这些packet, 那么停止发送流程。

|  |
| --- |
| 硬件加速模块： 探测 ARP, DHCP, IPv6 ICMP，交给 CPU 处理。硬件加速模块可以设计DPI提供给软件，以提供灵活性。  需要考虑DPI 能够处理帧内容多长的bit，和硬件成本相关。通常的需求是解析匹配Ethernet header 、IP header、TCP header 和 UDP header. |

* + 1. WFA hostspot 2.0 DGAF 相关功能 （TBD）

好像类似 WNM 的 traffic filter service 功能，可以在VAP 级进行过滤指定组播报文.

|  |
| --- |
| 硬件加速模块： 提供 DPI (data path interface), 可编程 规则，例如丢弃 匹配规则的 封包。灵活性 和 硬件设计复杂性之间可能存在此消彼长的关系，受硬件成本限制。 |

* + 1. 可选的 Vendor 实现的流控

|  |
| --- |
| 如果 packet 是组播（非广播），并且 1）其DA 存在于 snoop table；2）当前所关联的client 数量不为0；那么这个 packet 被转换为单播。根据 multicast enhancement 的模式设置， 转换方式可能是 Translate way 或者 Tunneling way.  Translate way: 比较简单，直接将 组播地址DA 换成 peer node 的 MAC address.  如果是peer node 是网桥设备，丢失了 原有的 组播地址DA. 这个应该好处理，  如果是4地址，就保留DA = 组播DA.  Tunneling way：添加了ether header 和 私有字段。优势是支持接收端是个无线桥设备。缺点是受兼容性约束。  注意点：仅对 IPv4 multicast address 进行 snoop convert. IPv6 ARP 是组播，其DA 为33:33:xx:xx:xx:xx,  Snoop 应该让这些frame 通过。 |

|  |
| --- |
| 硬件加速模块：考虑 硬件加速模块 在组播转单播 方面的实现   1. WiFi chip 自身是否会处理组播。如果会，交给WiFi chip 处理是否会导致性能低下。 2. 禁用WiFi chip 的组播处理，完全由SoC 端处理，该如何设计。 3. 有参考设计方案是 是将组播都交给CPU 处理。 |

* + 1. 通过 DA 查找目标node. 如果查找不到，那么停止发送流程。如果找到将 node 与 packet 进行关联。

|  |
| --- |
| 硬件加速模块：需要确定 WiFi 模块 服务的 都是 Ethernet frame.  硬件加速模块：提供 查找表  注意点，如果DA 是一个 组播地址，那么返回 local node. 每一个VAP 都有一个代表自己的node 数据结构。  结合上面步骤，可能要设计一个组播表。参考swtich 组播的设计，实现上应该没有太难的，需要考虑几张表的关联。  QCA软件上 将普通peer node table 和 wds peer node table 进行了区分。硬件设计应该可以统一使用一张表，针对一个DA 不应该会出现 既在 普通 peer node table, 又在 wds peer node table 的情况。Entry 含有flag 用以表示是否为 4地址模式。  组播是一个比较不好处理的case (TBD – 专题研究). |

* + 1. 计算 packet 的优先级，获得node 的 AC/TID 参数。

|  |
| --- |
| 实现细节：有个称为 DSCP override 的功能：  解析帧获得IP header，判断是否是igmp报文, 提取TOS 字段  驱动可能针对 igmp 帧 设定 固定的 tid. (可选)  驱动可能针对 特定 dip 设定固定的 tid （可选）  如果是 EAPOL 帧，设置EAOPL flag， 并设置tid = 6 (VO queue)  其他就用TOS 字段映射 TID. （注意 IPv6 frame 使用了 flow label 字段）  <vlan 的处理>  获得tid 再映射到 ac.  注意：实现上 会比较 IP DSCP 和 VLAN 各自获得的AC，使用优先级较高的一个。  如果 peer node 是非qos , 那么设置 tid = 0, ac=BE |

|  |
| --- |
| 硬件加速模块： 能够解析tos字段，推算出tid，并将tid 填充到 packet的描述符 |

* + 1. 如果报文关联的node是peer node, 但是该peer node 还未通过认证，并且packet 不是个EAPOL 帧，那么做丢包处理。

|  |
| --- |
| 未认证之前， 硬件加速模块 只处理EAPOL报文并且转交给 CPU 处理。其余报文做丢包处理。 |

* + 1. Vendor 的一种丢包策略 （Intelligent QoS for user experience）

描述：执行了一种状态机，当处于 BLOCKING, PROBING 状态时， UDP packet的发送是被阻塞的，即丢包。这是为了防止链路状况不佳的 client 影响总体性能。

|  |
| --- |
| 硬件加速模块： 这种功能可由软件实现 上述状态机，根据状态设置硬件过滤模块对UDP packet 进行丢包处理。 这个需求应该不太重要。 |

* + 1. Power save 处理（1）

如果报文所属的 ac 不是 UAPSD delivery，并且 node 处于 paused 状态，

那么将 packet 放入 node 专属的 power save queue. 并且适当设置 TIM bit.

这里注意 node 专属的 power save queue 有两个， 分别用于 data frame 和 mgmt frame.

这里小节视为空操作，QCA在UMAC, LMAC层 都有实现Power save Queue, 选择LMAC，继续往下走。

另外，802.11 spec MAC data plane architecture 给出的参考中，Power save 的操作是放在 AMSDU 之后。

* + 1. NAWDS 的处理。如果是个组播包，那么遍历各个 NAWDS node, 复制一份packet 发送到 NAWDS link. 需要注意的是要检查SA, 避免造成广播风暴。

|  |
| --- |
| 硬件加速模块： 如何处理 NAWDS feature ? 这里应该不需要关注是 NAWDS 还是 repeter, 找到这种node 都发一份。还是回到组播的处理上。 |

* + 1. 开始调用 LMAC 的 发送接口
  1. LMAC 的发送处理

UMAC 的发送 和 LMAC 的发送 存在 Qdisc, 这意味着存在一次异步调度。

这里所处理的是一个MSDU.

|  |
| --- |
| 硬件加速模块： 不存在且不需要这种Qdisc。 QoS 主要还是由 WiFi chip 的 EDCF 模块执行竞争 以及 TID queue 的调度实现。 |

* + 1. 通过 packet 的控制结构（例如skb）获取 预存 ac/tid, 找到对应的 HW queue 编号 以及 HW queue 对应的软件控制数据结构。
    2. 执行早期的丢包处理

|  |
| --- |
| 描述：  某些情况下要尽早丢包， 例如 tx buffer 数量少于一个预设的 阈值时，进行丢包。  对于某些关键 控制性质 的 帧，不受这个规则约束。  列举 这些高优先级的 帧： qosnull, eapol, dhcp, arp |

|  |
| --- |
| 描述：  如果当前 Block Ack aggrement 协商还未完成， 并且目标 HW Queue 中存在 aggregation 封包，并且不属于高优先级帧 （qosnull, eapol, dhcp, arp）， 那么进行丢包处理。  设计逻辑： 在 启用聚合的node 和 legacy node 共存的情况下，避免 legacy frame 和 非聚合HT frames 拉低总体性能。但是不能丢弃像 EAPOL 这种高优先级帧，否则会导致 解关联。 |

|  |
| --- |
| 描述：  如果是组播包， 并且当前可用的 TX描述符 低于预设阈值，并且不是高优先级帧，那么丢弃。  为了给 高优先级帧 预留16个TX 描述符，发送流程进行如下操作：  如果不是高优先级帧，并且可用的 TX 描述符少于 相应HW queue的最低描述符+16， 那么丢弃。 |

* + 1. UAPSD power save 检查

如果所属的 AC 启用了UAPSD，添加标记，在后续的处理中会放置到 UAPSD 相关的 queue. 适当设置 TIM.

* + 1. AMSDU 处理

如果node 启用了AMSDU, 相应的BA aggrement 也启用了AMSDU，那么做下面的AMSDU 处理。

聚合前检查：

1. 如果设置 UAPSD flag 或设置了 more data flag, 停止AMSDU 聚合。
   1. UAPSD flag 和 more data flag 都表示当前封包需要进入power save queue (普通的 power save queue或者UAPSD queue)。然而标准协议好像并没有对AMSDU 这方面有所约束。标准协议提供的参考是在做完AMSDU 聚合之后执行PS defer Queueing. 理论上来说，既然当前处于power save 状态而不能发送出去，那么应该说这时候执行AMSDU 是合理的。
2. 如果不是 IP/TCP 报文，停止 AMSDU 聚合。 （标准协议并未对要求IP/TCP做规范）

以上 A.B 两种情况下，如果当前AMSDU 缓存队列已经存在数据，先将这些已完成的AMSDU 数据发送出去。然后继续后面的发送流程。

要做聚合分以下两种情况处理：

1. 当前报文将是AMSDU是第一个subframe。

执行 AMSDU 封装，然后将该 AMSDU 加入 AMSDU调度器。释放原 报文。

如果还未启动amsdu flush timer，启动它。

1. 当前报文不是AMSDU 的第一个subframe

检查现存的AMSDU 剩余空间，如果不够了，那么先发出这个AMSDU, 再对新来的报文按普通发送流程处理。（新来的报文应该也可以成为一个新的 AMSDU 的第一个subframe, 标准并未约束。根据执行方便。）

这里存在一个具体实现的细节处理。如果当前rx phy rate 太低， 或者 tx queue 刚好有发送的报文，并且 AMSDU 空间还够，那么将新来的报文附加到AMSDU, 成为一个新的subframe。

否则 新来的报文将是最后一个 subframe （不做padding）, 附加到AMSDU 之后，然后释放原报文，最后将整个 AMSDU 发送出去。这个设计逻辑是，既然无线媒体资源可用，那就不必等待更多的MSDU而浪费无线媒体资源。

如果AMSDU聚合成功，那么这里停止发送流程。这里造成一个 异步调度。

如果当前packet不合适AMSDU聚合, 继续后面的流程。

|  |
| --- |
| 硬件加速模块： 软件上组建AMSDU 需要分配空间，然后将一个个subframe拷贝拼凑成AMSDU 报文。  思考如何通过描述符串接各个subframe，由硬件对多个内存地址存放的subframe 顺序进行DMA, 从而避免内存分配和拷贝操作。  了解有的实现：subframe的帧封装(添加LLC, padding) 是由软件实现的，硬件仅仅执行DMA. 如果纯硬件加速，需要考虑实现AMSDU queue, MSDU 到 AMSDU subframe 的封装。 |

* + 1. 对 MSDU 进行 MPDU 封装

需要关注的信息有：

是否是 QoS data frame, 是否使用4地址模式，这将影响 802.11 header size.

需要填充 802.11 header, LLC。但是还未考虑加密需要的额外字节， 这个参考后面步骤。

注意点1： FC 字段标记 IEEE80211\_FC1\_WEP 影响后面的加密计算

注意点2： 这里赋予了 Sequence number. Sequence number 赋值应在分片之前完成。

* + 1. 考虑分片检查和执行分片。

如果报文长度（包含802.11header） 大于预设的分片阈值，并且不是组播报文，进行分片操作。如果需要分片，针对TKIP 加密必须软件计算 MIC（这里会消耗CPU资源）。为分片分配新的wbuf, 各个分片串在一起，除了最后一个分片，Frame control 字段设置 more flag bit.

|  |
| --- |
| 硬件加速模块：做分片? AMSDU 不做分片。分片具有很大的实际意义吗？ |

|  |
| --- |
| 硬件加速模块：软件上在做AMSDU 聚合 和 MPDU 封装步骤都有做 LLC 封装，硬件实现建议分下面步骤按次序进行。  Ethernet frame ===> Ethernet SNAP ==> AMSDU=>  |========|====> MPDU |

* + 1. 如果还未 发送 ADDBA建立 BA aggrement, 那么发送 ADDBA

|  |
| --- |
| 硬件加速模块：这属于软件功能，硬件可能需要软件来设置flag, 硬件参考这个flag 选择是否进行做AMPDU 聚合。 |

* + 1. 构建 tx controller 数据结构

包含 rate, power, rts，管理帧标记，pspoll 标记，BAR 标记，是否使用最低速率发送，是否是EAPOL 等信息，这些信息在 后面进一步转换为 HW Tx descriptor. Node 信息也暂存于 tx controller 结构体。

这里将计算完整的报文长度，包含不同加密模式下需要的字段（CCMP header ,IV，MIC, ICV等），还会加上用于FCS 的4字节。注意仅仅是计算，不会做报文扩充，填充工作应该是硬件发送时完成。

|  |
| --- |
| 硬件加速模块：目前还未遇见过全hw datapath 的实施方案。  QTN offload 主要是从Linux offload 到 F/W, 内容主要有AMSDU Aggregation, AMPDU Aggregation, De-aggregation 以及reorder; 硬件方面主要是执行DMA工作， 包含LLC, 802.11 header帧封装是由软件执行。  QCA IPQ807X 在 RX 方向似乎实现了完整HW datapath, 主要是 reorder 部分. 不清楚HW是否支持AMSDU De-aggregation. 在TX 方向 AMPDU 调度是由FW 实现，不清楚HW是否支持AMSDU Aggregation. |

* + 1. Power save 处理 – Legacy power save

如果 报文在前面处理中贴上了 legacy power save标记，那么缓存到相应的 power save queue中。停止发送处理。

Note- 802.11 spec参考给出的指导是 在作为AMSDU 之后，赋予Sequence number 直接进行”PS Defer Queuing” 处理。

* + 1. 对 payload 进行DMA 映射
    2. 将上面传来的 tx controller 数据 封装入 软件TX描述符 （ath\_buf）。
    3. Power save 处理 – UASPD

如果 Tx controll 中标记了 uapsd， 那么将 MPDU 缓存到 UAPSD queue.

停止发送。

* + 1. AMPDU 处理 与 non-AMPDU 分支
       1. AMPDU 处理

如果满足以下条件，那么现将 MPDU 缓存到tid 的队列中， 然后将tid 加入到调度候选集。并适当的启动TX Queue调度。具体聚合的处理看调度部分。

* + - 1. non-AMPDU 分支 (有下面两个分支)
         1. 发送到 普通 HW TX Queue 的处理

这里有个处理细节，遇到分片的情形，暂缓发送，等待最后一个分片下发到这里再一起发送出去。

这种设计的考虑是：部分片段可能因为 tid paused 暂缓，而部分片段发送出去了，这将导致HW 忙于等待后续的因为tid paused暂缓的片段。

如果满足以下条件中的一个：

1. tid 中有挂起的 ath\_buf
2. tid 当前处于 paused
3. Hardware TX queue 中还有正在处理的报文
4. 当前报文是分片报文（所有的相关分片已经集结完毕），

执行下面的处理：

将MPDU串到 tid 的 bufer queue 中，然后将tid 加入到 对应 TX queue 的调度候选集。并启动这个Tx Queue调度， 参考下面1.2.12.3。 然后从这返回。剩下的工作是 HW TX queue 进行调度发送报文，再后面的工作就是 Tx Done 的处理。

下面是将要直接发送到 硬件的处理：

将软件TX描述符 中的表示 硬件描述的地址 写入WMAC 寄存器，启动硬件开始执行描述符解析、 发送报文，向CPU 报告TX Done.

* + - * 1. Power save 处理 （4）- 组播包

当 AP 关联有一个或多个 处于 PS mode 的client，或者当前 mcast queue 有组播报文缓存，将MPDU 缓存到 专用的 mcast queue. 这个mcast queue 是与VAP 相关的，因此存放在VAP 结构体中。自然在发送beacon 之后会将该队列中的包推送出去。

* + - 1. TX Queue 调度

注意点，这个 TX Queue 调度并没有产生软件上的异步。但是对于当前MPDU 的发送流程来说，是一个异步操作。具体一点就是发送的可能是另一个MPDU.

TX Queue 调度的逻辑描述：

从 TX Queue 中的 AC 队列取出第一个AC元素, 从AC 数据结构体的tid queue中取出第一个Tid 元素, 如果符合发送条件，就执行发包处理，如果该 tid 还处于paused 状态，那么放回到 等待发送队列的尾部。

发包处理中，如果 AMPDU聚合条件满足, 则进行AMPDU 发送，否则进行普通MPDU 发送。

* 1. TX Done 处理

WMAC 提供 TX status, 从Tx Status 获取 tx rate 信息，将用于rate adapter / rate algorithm. 收集RSSI, PER 信息。 处理完之后，将需要释放的 ath\_buf 释放，回收到ath\_buf 池中。

|  |
| --- |
| 硬件加速模块：需要研究目标WiFi chip SoC 端驱动。   1. PCIe 界面的SoC端 可能不需要涉及 TX Done这部分内容。 2. PCIe 的SoC 端可能需要 hold 下发的报文，直到 WiFi chip 端对报文进行了ACK.   具体还是要分析目标驱动 |

## RX datapath

RX datapath 从 RX Done 中断开始。硬件上报给CPU 的内容是 Rx Status + data, 中断上半部将 Rx Status 转化为软件抽象的 Rx descriptor， buffer 偏移掉 Rx Status 即作为skb->data 的值 – 指向MPDU。上半部将要处理的MPDU 放进 rxqueue, 下半部就是从这个rxqueue 中开始处理。

WMAC 呈现给软件的是MPDU, WMAC 已经完成了 A-MPDU De-aggregation, 也完成了MPDU Header + CRC的验证。WMAC会设有filter 功能（含有Addr1 地址过滤），未通过验证的报文可以被过滤掉而不上报给软件，从而降低面向软件的压力。并且WMAC以及完成了解密和完整性检查。

* 1. 中断上半部
     1. 收取单个MPDU的处理细节

首先获取一个buffer 描述符，将硬件上报的Rx Descriptor 解析成软件抽象的、统一的Rx Status.

* + 1. UAPSD triggers 的处理

如果是 UAPSD trigger frame, 从相应 UAPSD queue中响应数据报文。这里放置在中断上半部是为了做高优先级的快速响应。

* + 1. 将 MPDU 串待处理报文队列，留到中断下半部处理
  1. 中断下半部
     1. 适当初始化报文描述符(sk\_buff)处理
     2. RX rate，RSSI 信息收集
     3. 查找node，关联报文与node

通过 key ID 查找 node，或者通过802.11 header 中的TA 进行软件查找。如果没有找到node, 那么使用 ieee80211\_input\_all() 处理。

|  |
| --- |
| 硬件加速模块：这种未找到node的报文需要上报给软件。软件将遍历各个VAP，进行分别处理。例如，当STA首次发起Probe request等报文，本地还未建立相应的node，就需要走这个路径。 |

* + 1. 基本frame size 检查

|  |
| --- |
| 硬件加速模块：对于frame size 异常 可能由硬件丢弃，或者上报给CPU 处理，主要用于统计错误信息。 |

* + 1. RX datapath 分支
       1. Non-AMPDU RX process

这条路径上 没有做reorder.

* + - * 1. Power save state 检查。检查FC 中的 power mgmt bit.

|  |
| --- |
| 硬件加速模块：需要检查这个bit，如果发生了切换，需要报告给软件 。 |

* + - * 1. Bandsteer feature.

|  |
| --- |
| 硬件加速模块：硬件需要记录timestamp 信息。软件周期性检查 active/inactive 状态变化。 |

* + - * 1. WoW feature.

|  |
| --- |
| 硬件加速模块：Wake on wireles. 这个feature 是系统检查特征包以唤醒系统。这个特征包不是标准统一的。因此硬件匹配方面不好执行。  TBD: 帧格式是否是标准统一的, 如果是，弄清楚具体格式。  TBD: WiFi Chip 是否会执行这个feature, 以事件的方式通知SoC? |

* + - * 1. workaround.

|  |
| --- |
| 硬件加速模块：许多平台设计可能会有缺陷，后期软件上会有一些修复。列举在这里是要注意，目标SoC chip 的中驱动可能会有些workaround, 这个可能会被硬件加速覆盖而失效。 |

* + - * 1. 系统回调。

这是驱动提供给系统的一个回调接口。例如如果处理之后，得到的结果是要过滤，那么做丢包处理。

* + - * 1. Defragmentation

Node 结构体中有分片缓存队列，用于收集所有分片报文。收集完之后继续正常的流程。

|  |
| --- |
| 硬件加速模块：硬件何处理 这种分片包？ |

* + - * 1. AMSDU 处理

|  |
| --- |
| 硬件加速模块：硬件如何处理？ |

* + - * 1. MPDU 转成Ethernet frame
        2. VAP bridge处理

如果目标是VAP 下的另一个node，并且支持VAP bridge 功能，那么这里进行转发。相当于TX datapath 中 VAP 的hardstart() 开始，这将在其中的一个异步点返回，将跳过送至协议栈的步骤。

* + - * 1. 做最后的skb 设置，然后送至协议栈。
      1. AMPDU RX process
         1. BAR 处理。BAR 会推动re-order buffer 状态机。

|  |
| --- |
| 硬件加速模块：控制帧需要送至CPU |

* + - * 1. 如果是下面情况的一种, 使用上述”Non-AMPDU RX process” 一节描述的内容处理。

1. 组播
2. 非Data frame
3. 不是QoS Data frame
4. Block Ack aggrement 的建立还未完成
   * + 1. 找到 RX TID state 数据结构；从MPDU 中取出Sequence number；执行re-order buffer 的处理。

从re-order buffer 提交给上层的操作，基本是上述”Non-AMPDU RX process” 一节描述处理。

|  |
| --- |
| 硬件加速模块：re-order buffer 的实现  状态机推力来源：相关BAR, MPDU, 参考802.11-2012 9.21.4 & 9.21.7.6  高通IPQ807X 实现了硬件处理 re-order buffer. |