RW-WLAN-nX UMAC SW

Functional Specifications

RW-WLAN-nX-UMAC-SW-FS/0.04

Version 0.04

2016-03-01

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
0.01	2014-09-09	Initial Release	Steven L'Her
0.02	2015-05-28	Added SM module	Steven L'Her
0.03	2015-06-29	Added APM and RC description	Steven L'Her
0.04	2016-03-01	Updated rate control description	Flavia Vanetti

Changes between a version and the previous one is reflected by the addition of **change bars**, like for the line below:

TBD	Date entered	Description	Status

Items to be determined in the future versions of this document

Table of Contents

List of Figures

List of Tables

1 Overview

1.1 Document overview

1.1.1 Purpose

The document deals with the high level overview of the RW-WLAN-nX Upper MAC SW (Hence forth referred as UMAC SW). The purpose of the document is to give high level design of the UMAC SW to MAC development and verification engineers.

1.1.2 Scope

The scope of this document is to provide a high level description of the UMAC SW blocks and the functionality handled by each of the blocks. The exact "C"/assembly language functions/sub-routines, local variables etc. are not defined here.

1.1.3 Overview

This document describes the design in two levels in the ascending order of detail provided.

Level 1: *UMAC SW design overview*

Modules of UMAC-SW and their interfaces are defined at a high level. The operation of UMAC-SW is illustrated by explaining the interaction of these modules.

Level 2: *Functional blocks*

Sub-modules/functional blocks of the main modules defined in *UMAC SW design overview* are identified. A description of the operation of each functional block is given.

The design of each functional block is illustrated by explaining state machines, logic/algorithm at a higher level

High level data structures created/used by functional blocks are specified. Messages and message flow sequences between functional blocks are illustrated.

1.1.4 Abbreviations and Acronyms

AC	Access Category
ACK	Acknowledgment
ACM	Access Category Mandatory
AID	Association Identifier
AP	Access Point
APSD	Automatic Power Save Delivery
A-MPDU	Aggregate MAC Protocol Data Unit
A-MSDU	Aggregate MAC Service Data Unit
BA	Block Acknowledgement
BAR	Block Acknowledgement Request
BSSID	Basic Service Set Identifier
CF	Contention Free

CSI	Carrier State Information
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
EDCA	Enhanced Distributed Channel Access
FIFO	First-In-First-Out
FC	Frame Control
FCS	Frame Check Sequence
HCCA	HCF Controlled Channel Access
HT	High Throughput
IBSS	Independent Basic Service Set
ICV	Integrity Check Value
IV	Initialization Vector
LLC	Logical Link Control
L-SIG	Legacy (Non-HT) Signal Field
MAC	Medium Access Control
MEM-BAR	Memory Base Address Register
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
MLME	MAC Sub Layer Management Entity
MMPDU	MAC Management Protocol Data Unit
MPDU	MAC Protocol Data Unit
MSDU	MAC service data unit
NAV	Network Allocation Vector
OS	Operating System
PC	Point Coordinator
PHY	Physical layer
PLME	PHY Sub Layer Management Entity
PS	Power Save
PSMP	Power Save Multi-Poll
QAP	QoS Access Point
QoS	Quality of Service
QSTA	QoS Station
RD	Reverse Direction
RDG	Reverse Direction Grant
RSNA	Robust Security Network Association

RTS	Request to Send
RX	Receiver
SSID	Service Set Identifier
STA	Station
STBC	Space-Time Block Coding
TID	Traffic Identifier
TIM	Traffic Indication Map
TS	Traffic Stream
TU	Time Unit
TX	Transmitter
TX_REQ	Transmit Request message received from UMAC-SW
U-APSD	Unscheduled Automatic Power Save Delivery
WEP	Wired Equivalent Privacy
WLAN	Wireless LAN
WM	Wireless Medium

Table 1: Abbreviations and Acronyms

2 High level design overview

2.1 Design objectives

The UMAC SW design objectives are to:

- ✓ Perform the SME/MLME operations required to associate with peer WiFi devices.
- ✓ Perform the WiFi encapsulation/decapsulation of data packets.
- ✓ Drive the RW Lower MAC.

2.2 Relationship to external environment

Refer to [2] for the description of the interface between the host (driver) and the UMAC SW.

Refer to *[3]* for the description of the interface between the UMAC SW and the LMAC SW.

2.3 UMAC SW design overview

Figure 1 below shows the high level blocks of the UMAC SW as well as the dependencies with the LMAC SW. It has to be noted that the UMAC SW is not a standalone SW module. It has to be compiled along with the RivieraWaves LMAC SW to form a "FullMAC" SW (FMAC SW). For details about the LMAC SW, refer to [3].

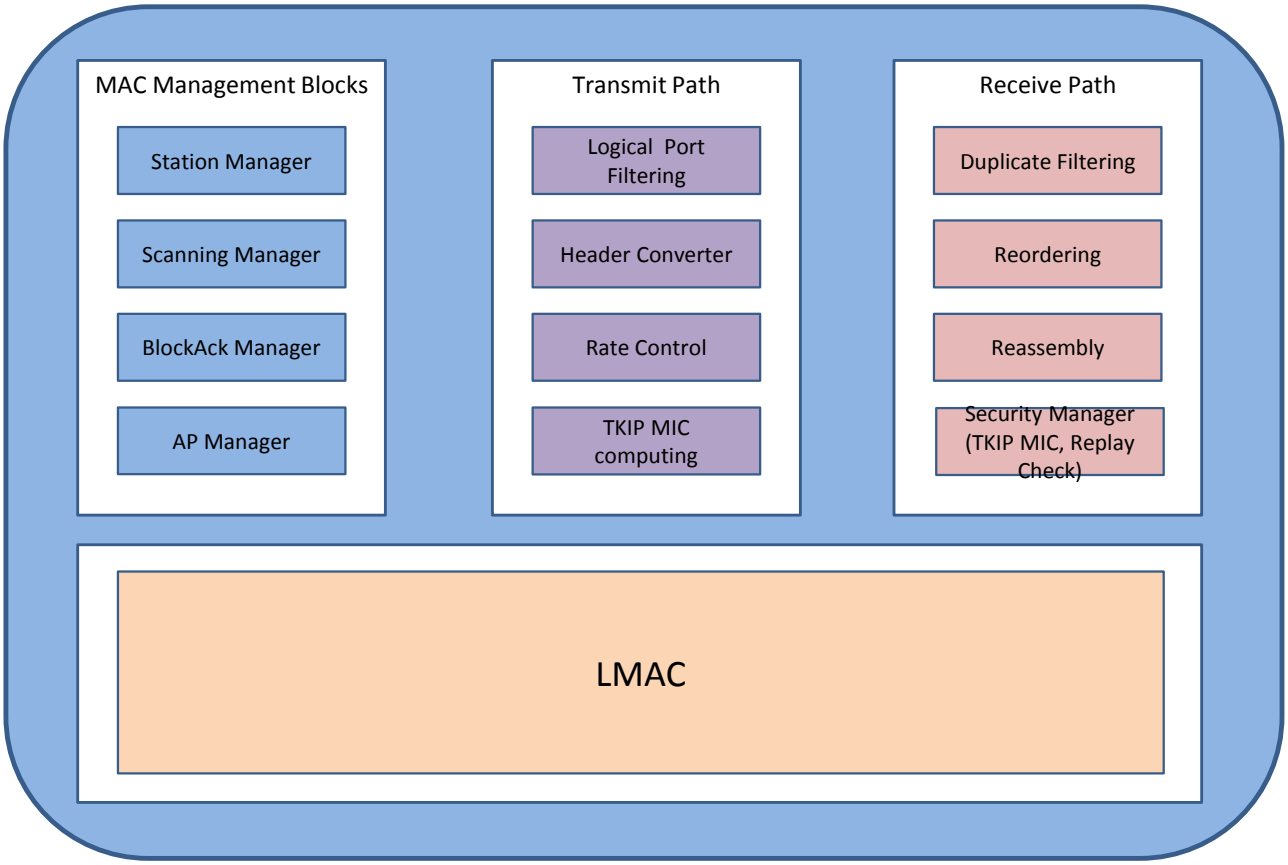

Figure 1: UMAC SW and LMAC SW overview

2.3.1 UMAC SW

The UMAC SW is responsible for the MAC HW configuration and the chaining of descriptors for the transmission and reception. The UMAC SW modules are grouped into four blocks.

- ✓ MAC Management Block
- ✓ TX Path Block
- ✓ RX Path Block

2.3.1.1 MAC Management Block

This block is responsible for the SME/MLME operations of the MAC:

- ✓ Management of the scanning requests coming from the host
- ✓ Station operation to associate with an Access Point
- ✓ Access Point configuration and operation
- ✓ Management of the transmission and reception Block Acknowledgment agreements

2.3.1.2 TX Path Block

This block is mainly responsible for the WiFi encapsulation of the packets that need to be transmitted. The following operations are performed by this block:

- ✓ Logical Port Filtering
- ✓ MAC header and security header creation
- ✓ Rate control
- ✓ TKIP MIC computation
- ✓ Interface with the LMAC TX path

2.3.1.3 RX Path Block

This block is responsible for the reception of the packets from the LMAC RX path and of their indication to the upper layers. The following operations are performed by this block:

- ✓ Filtering of the duplicate frames
- ✓ Reordering of data packets received under a BlockAck agreement
- ✓ Security checks (TKIP MIC, replayed frames, etc.)
- ✓ Reassembly of fragmented packets
- ✓ Header conversion (from MAC to 802.2 header)

2.4 Functional blocks

2.4.1 Introduction

This section describes the functionality of individual modules of UMAC SW blocks. The Figure 2 shows the UMAC SW modules.

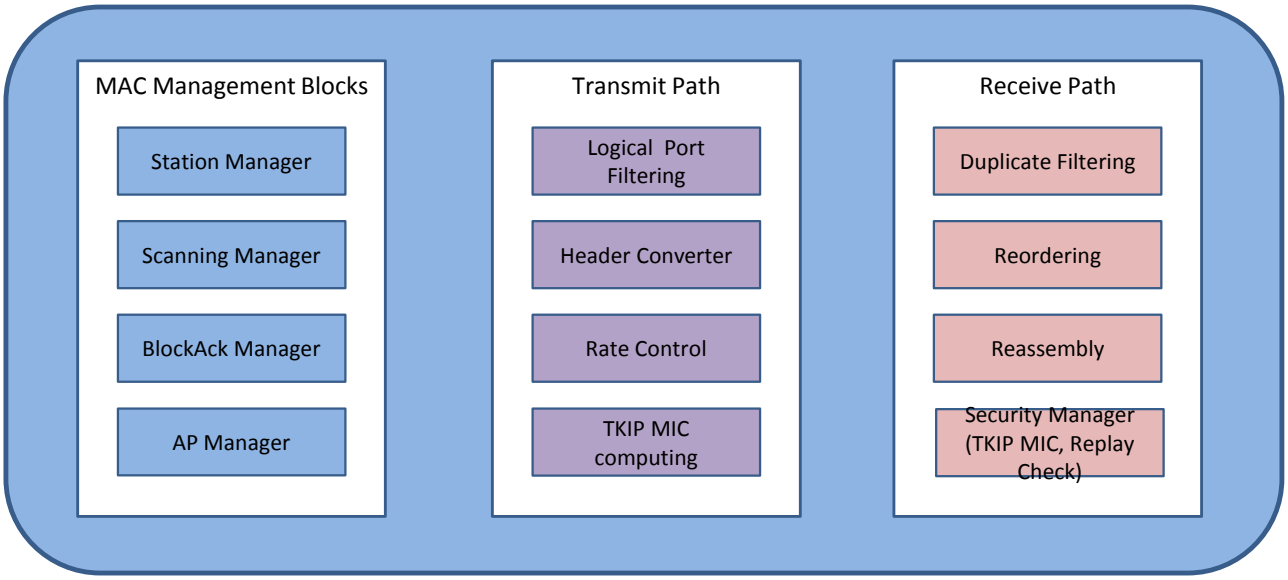

Figure 2: Functional block diagram of UMAC SW

The following sections give a detailed description of the constituent modules of each of these blocks.

2.4.2 MAC Management Blocks

2.4.2.1 Scanning Manager (SCANU)

This block is responsible for performing 2 types of actions:

- ✓ The discovery of networks (scanning)
- ✓ The synchronization with the network the Station Manager is attempting to connect to (joining)

This block is implemented as a kernel task called SCANU that can interface with both the host system (for scanning) and the Station Manager (for scanning and joining). The SCANU task implements 2 different states:

- ✓ IDLE: no procedure is ongoing
- ✓ SCANNING: the SCANU module is currently involved in a scanning or joining procedure

Additionally to its kernel message interface, the SCANU provides a function based API that allows the Station Manager module to perform searches in the scan data base. The functions available for that are the following:

*struct mac_scan_result *scanu_search_by_bssid(struct mac_addr const *bssid);*

This function searches for a specific BSSID through the scan data base. It returns the pointer to the corresponding scan result if found, and NULL otherwise.

*struct mac_scan_result *scanu_search_by_ssid(struct mac_ssid const *ssid);*

This function searches for a specific SSID through the scan data base. It returns the pointer to the corresponding scan result if found, and NULL otherwise. If several scan results have the same SSID, then the function returns the result having the highest RSSI.

2.4.2.1.1 Scanning procedure

The purpose of the scanning procedure is to discover the networks available in the neighborhood. The SCANU modules keeps only a small set of parameters of the discovered networks in order to limit the amount of memory required for the database. The set of parameters stored in the SCANU environment during a scanning procedure is the following:

- ✓ The channel on which the network is discovered
- ✓ The RSSI of the received frame
- ✓ The BSSID of the network
- ✓ The SSID of the network
- ✓ The capabilities of the network
- ✓ The beacon period

The information required for a connection with a specific network will then be retrieved using the joining procedure.

2.4.2.1.1.1 SCANU_START_REQ handling

Upon the reception of SCANU_START_REQ message the SCANU module will prepare the list of channels to be scanned and proceed to the scanning on the first band:

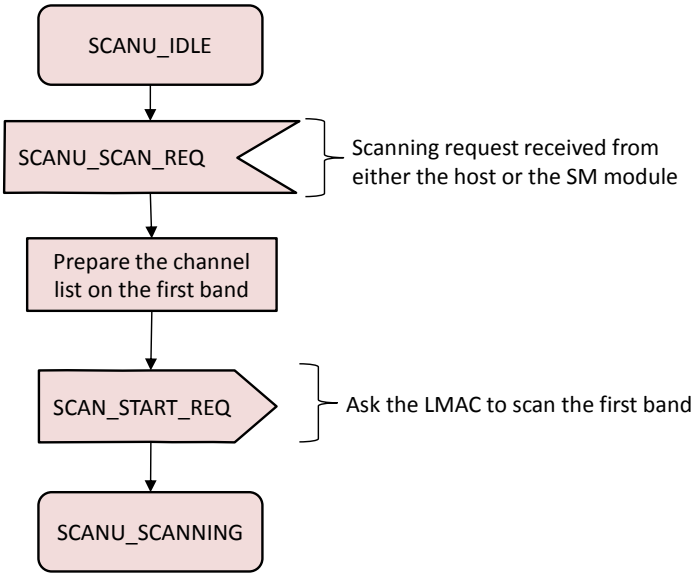

Figure 3: SCANU_START_REQ handling

2.4.2.1.1.2 SCANU_SCANNING state

The SCANU module will then wait for the confirmation of the scanning startup from LMAC SW:

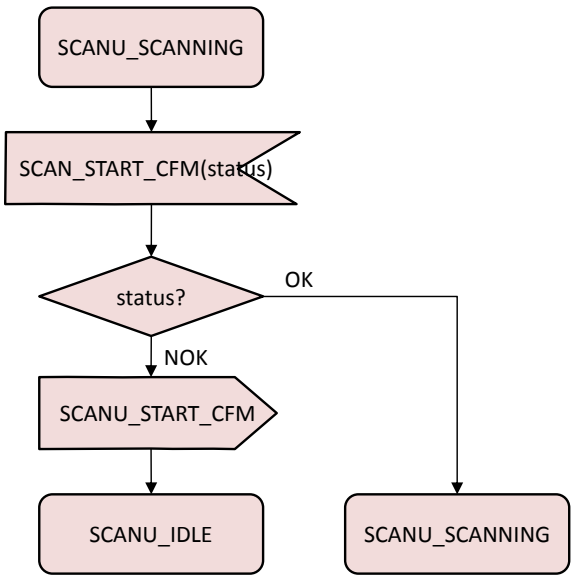

Figure 4: Confirmation of the scanning start from the LMAC

If the LMAC confirmation indicates that the scanning was not started, then the procedure stops immediately and the SCANU module goes back to IDLE. Otherwise it stays in SCANNING state, waiting for packet indications (beacons and probe responses) from the RX path:

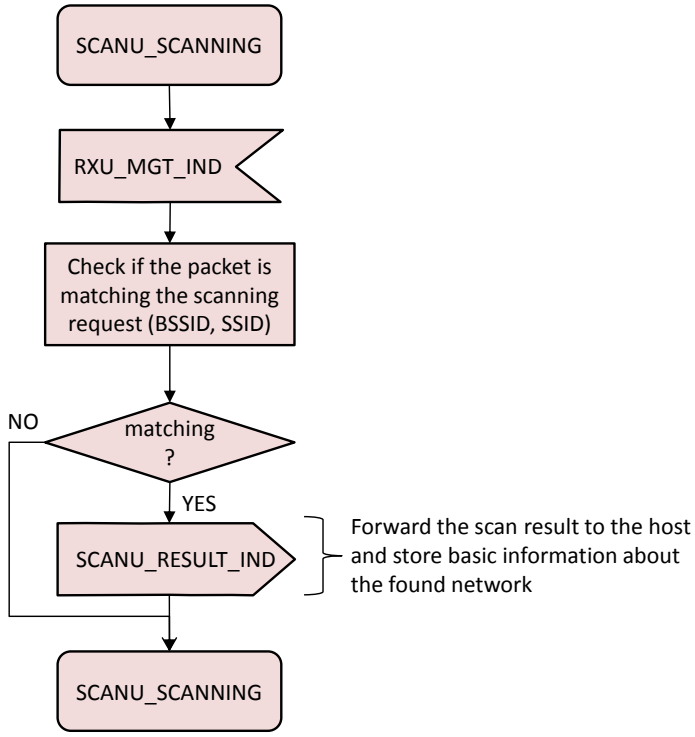

Figure 5: Getting scan results

The scanning on the first band terminates upon the reception of SCAN_DONE_IND from the LMAC. The SCANU module then finishes the procedure or scans the second band if required:

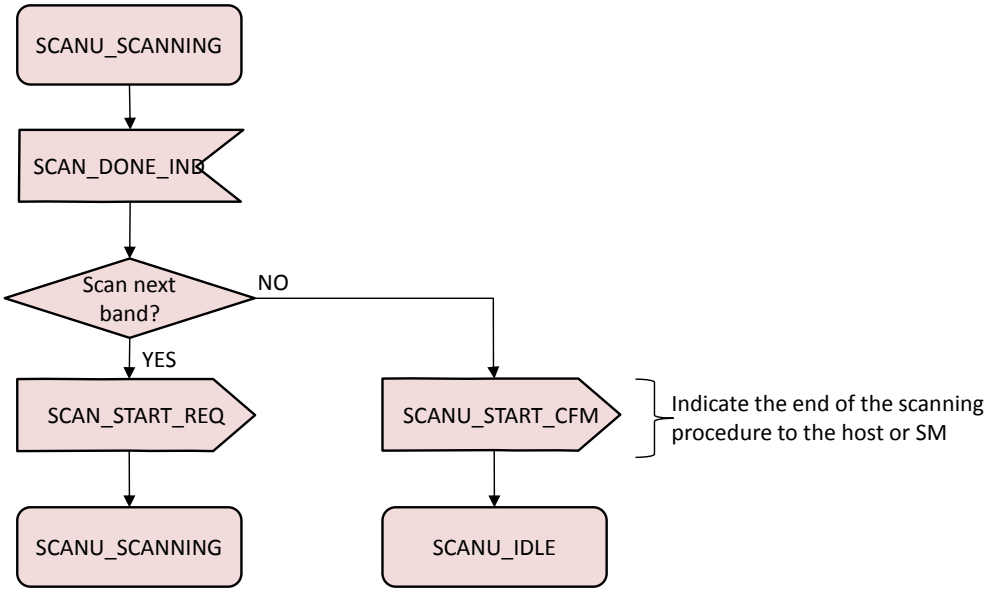

Figure 6: Scan next band or finish scanning

2.4.2.1.2 Joining procedure

The joining is invoked by the SM module prior to the association with a specific network. It is used to retrieve the missing information about the network that will allow proceeding with the association. The additional information retrieved is the following:

- ✓ The legacy rate set used by the network
- ✓ The HT parameters

- ✓ The QoS parameters

The procedure consists in the same steps as the scanning procedure, except that it is invoked using the SCANU_JOIN_REQ and finished by a SCANU_JOIN_CFM. The SCANU_JOIN_REQ request parameters have to contain a BSSID different from the Wildcard BSSID.

2.4.2.2 Station Manager (SM)

This block is responsible for the operations required to associate with an Access Point. The major steps of such a procedure are the following:

- ✓ Optionally start a scanning procedure, if no scan result is matching the requested SSID to connect to
- ✓ Join the network (see 2.4.2.1.2)
- ✓ Create the channel context that will be used for this connection
- ✓ Add the STA entity that will be used to store information about the peer device
- ✓ Set the BSS parameters
- ✓ Authenticate and Associate with the network

The SM module also manages the disconnection from an Access Point.

This block is implemented as a kernel task called SM that interfaces with the host system to get the connection/disconnection requests. The SM task implements the following states:

- ✓ SM_IDLE: no procedure is ongoing
- ✓ SM_SCANNING: the SM has requested a scan to the SCANU
- ✓ SM_JOINING: the SM has requested a joining to the SCANU
- ✓ SM_CHAN_CTX_ADDING: the SM has requested the addition of a channel context to the LMAC
- ✓ SM_STA_ADDING: the SM has requested the addition of a new STA entry to the LMAC
- ✓ SM_BSS_PARAM_SETTING: the SM has sent the BSS parameters to the LMAC and waits for the completion
- ✓ SM_SCHEDULING_CHAN_CTX: the SM has requested the LMAC to schedule the channel context previously allocated.
- ✓ SM_AUTHENTICATING: the SM has sent an authentication message to the AP and is waiting for the response
- ✓ SM_ASSOCIATING: the SM has sent an association request to the AP and is waiting for the response
- ✓ SM_DISCONNECTING: the SM is currently involved in a disconnection procedure

2.4.2.2.1 Connection procedure

2.4.2.2.1.1 SM_CONNECT_REQ handling

The connection procedure is invoked by the host using the SM_CONNECT_REQ message. The handling of this message is done as follows:

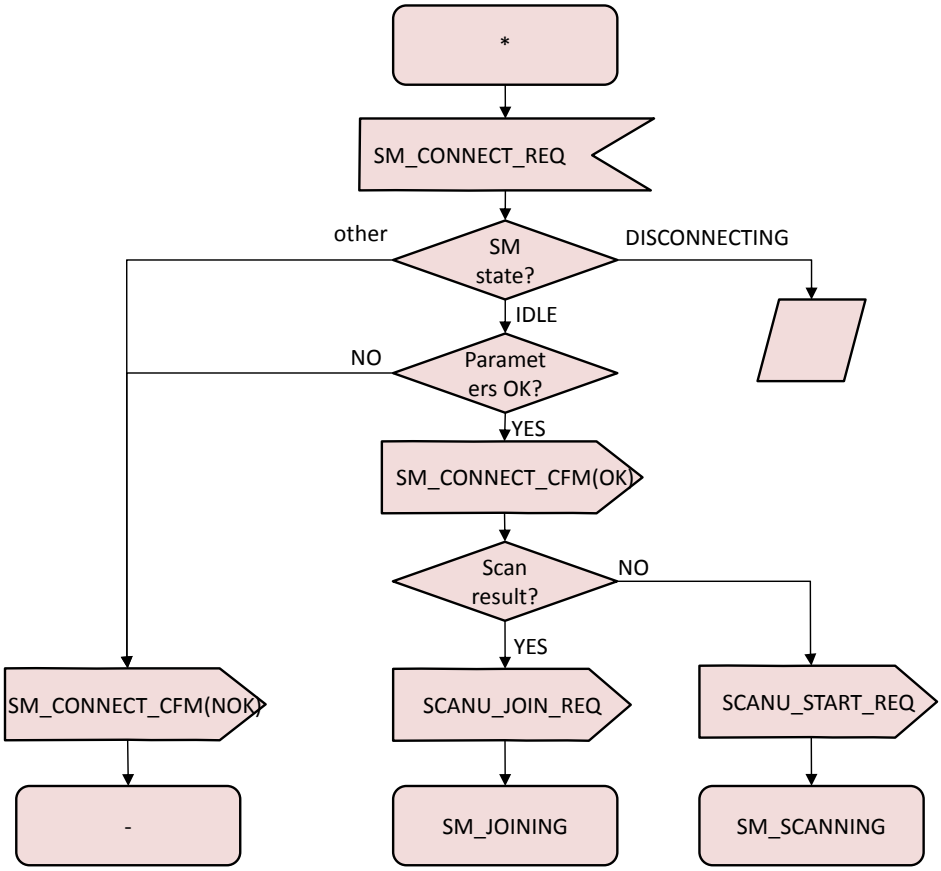

Figure 7: SM_CONNECT_REQ handling

The test determining if the parameters are OK consists of the following checks:

- ✓ Check that the interface for which the connection is requested is of STA type
- ✓ Check that this interface is not already connected to an Access Point

The scan result check then consists in verifying that for the requested SSID to connect to there is a scan result entry available (this search is done via the function API of the SCANU module – see 2.4.2.1). This entry allows knowing the BSSID and channel on which the joining procedure should be performed.

2.4.2.2.1.2 SM_SCANNING state

When a scanning has been requested, the SM task will proceed as below at the end of the procedure:

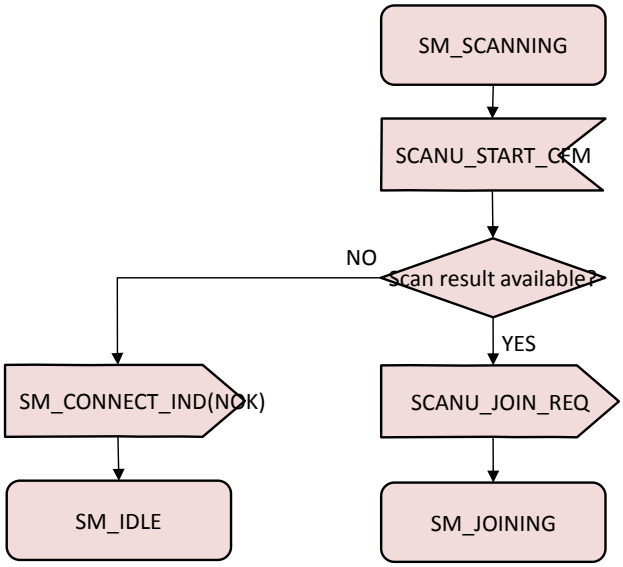

Figure 8: SM_SCANNING state

If the scanning procedure did not allow getting information about the SSID we are required to connect to, then the connection procedure terminates with the sending of a SM_CONNECT_IND message to the host, indicating a failure status. Note that when a SM_CONNECT_IND is sent (in this state or states below) with a failure status, the resources previously allocated for the connection (channel context, STA entry, etc.) are released.

2.4.2.2.1.3 SM_JOINING state

When a joining has been requested, the SM task will proceed as below at the end of the procedure:

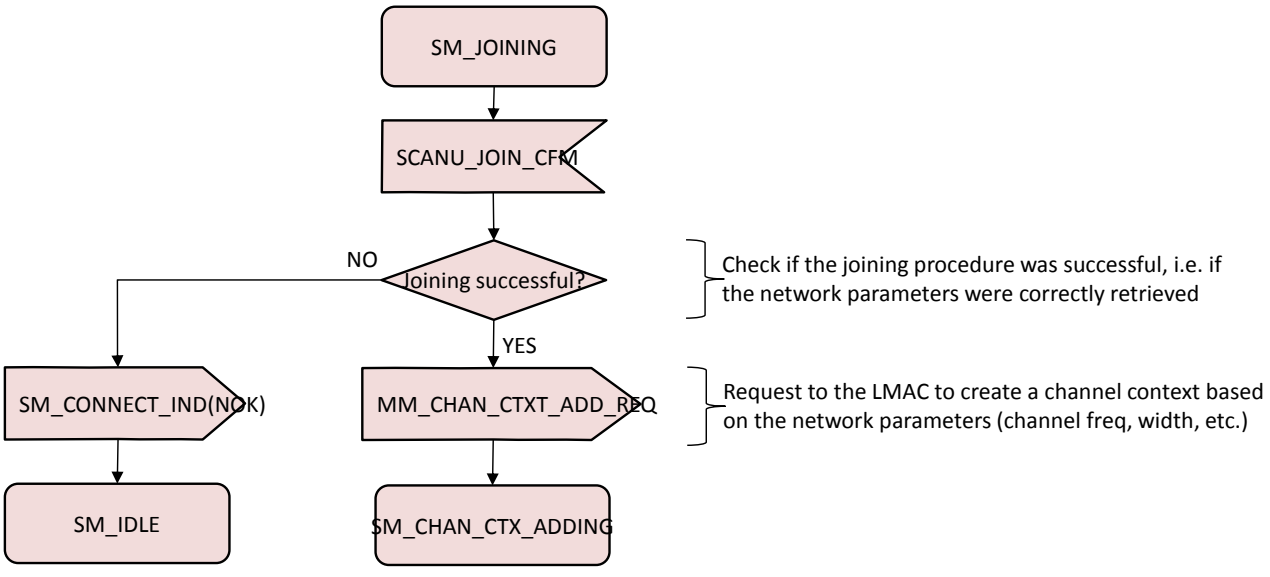

Figure 9: SM_JOINING state

Once the joining has been performed, a channel context creation is requested to the LMAC with the parameters retrieved from the network information.

2.4.2.2.1.4 SM_CHAN_CTX_ADDING state

Once the channel context addition has been performed by the LMAC, the SM adds an entry to the station database:

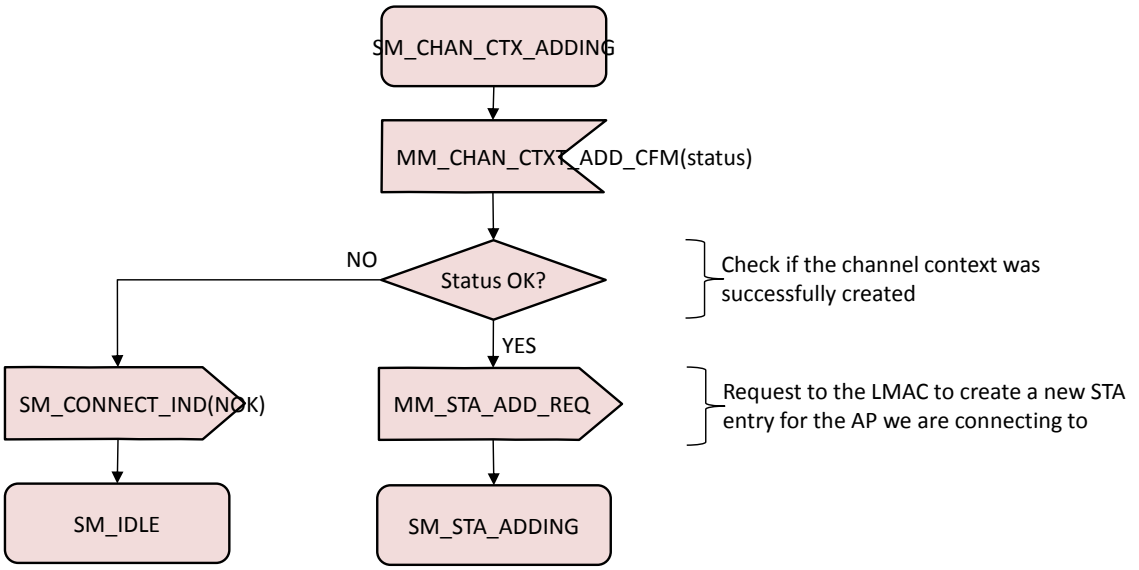

Figure 10: SM_CHAN_CTX_ADDING state

2.4.2.2.1.5 SM_STA_ADDING state

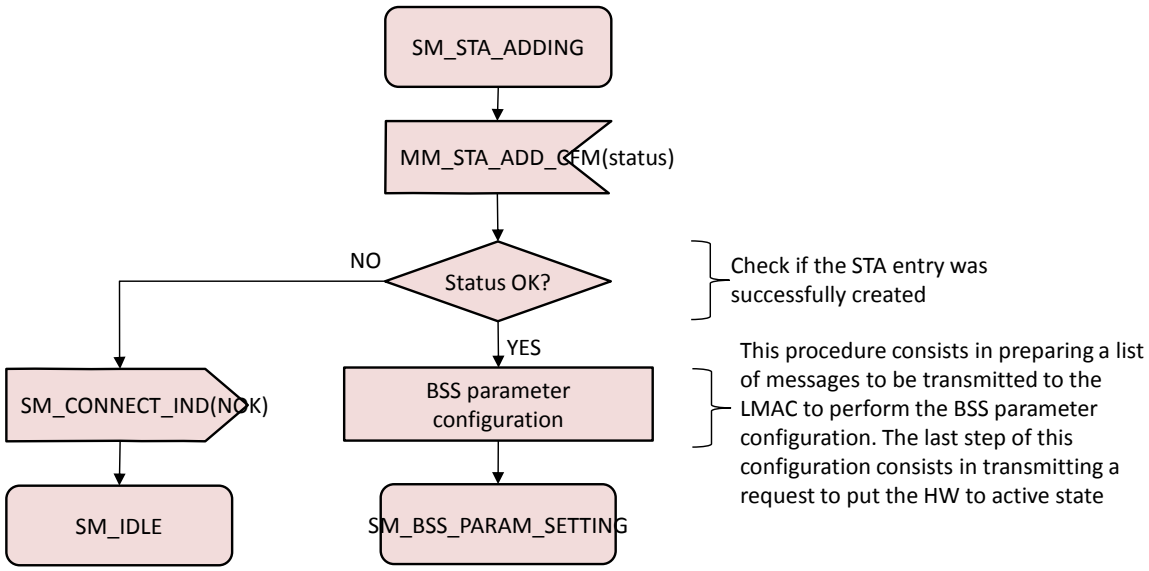

Figure 11: SM_STA_ADDING state

2.4.2.2.1.6 SM_BSS_PARAM_SETTING state

The SM will stay in this state until the completion of the parameters setting. Each time a BSS parameter setting confirmation is received, the next parameter is sent to the LMAC (the message is popped from the list that was built earlier):

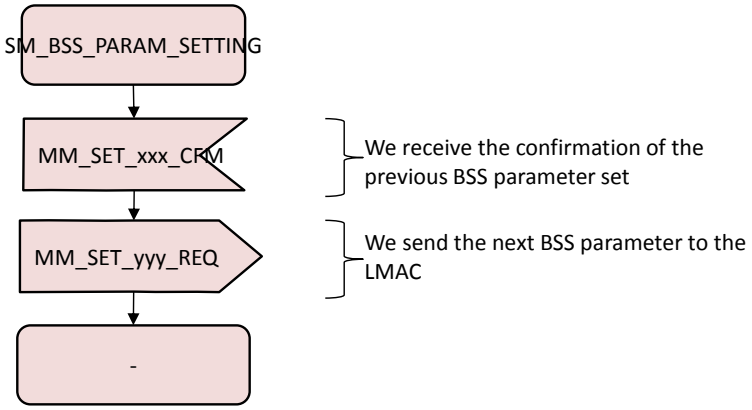

Figure 12: SM_BSS_PARAM_SETTING state – Next BSS parameter setting

Once the BSS parameters have been set and the system is put to active mode, the SM requests the LMAC to schedule the channel context that was previously allocated in order to get a "window" of availability on this channel to proceed to the association procedure:

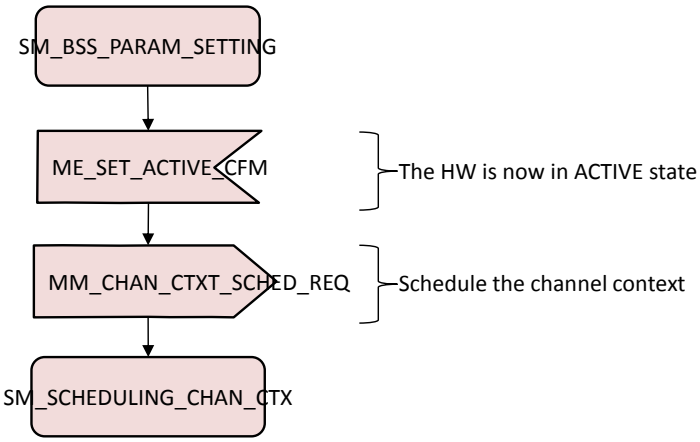

Figure 13: SM_BSS_PARAM_SETTING state – HW set to active

2.4.2.2.1.7 SM_SCHEDULING_CHAN_CTX state

Upon the indication of the end of channel scheduling by the LMAC, the SM can start the association procedure:

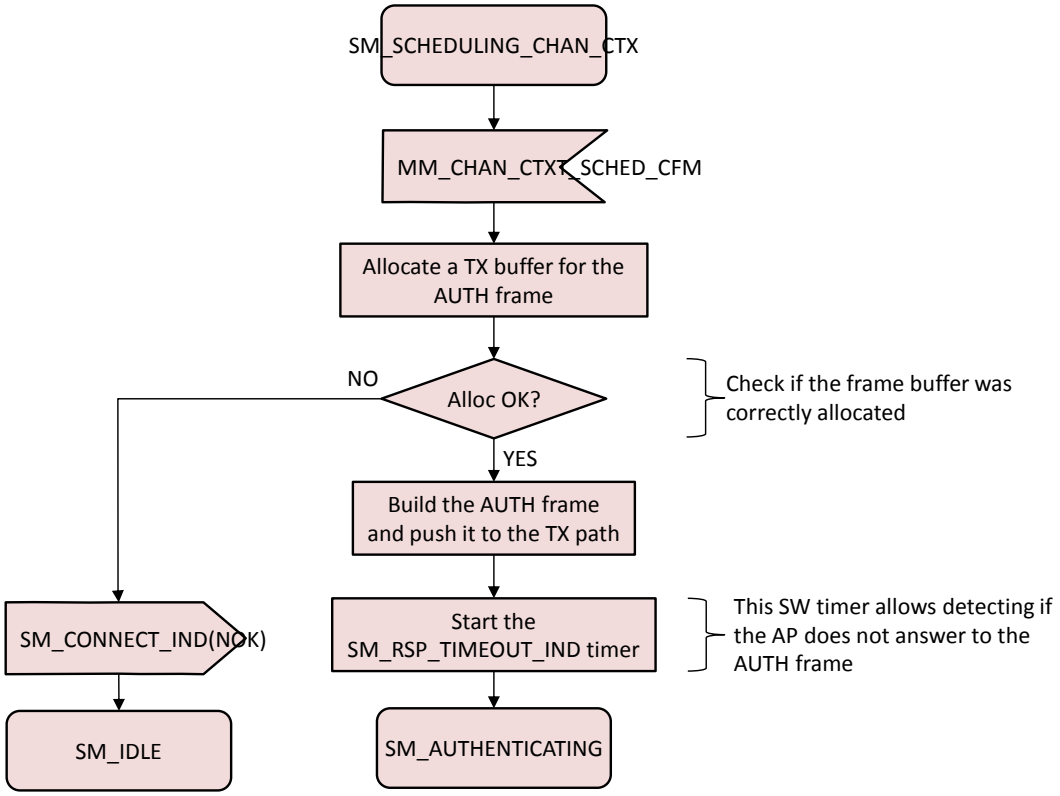

Figure 14: SM_SCHEDULING_CHAN_CTX state

2.4.2.2.1.8 SM_AUTHENTICATING state

In the authentication state different types of messages can be received. In a successful case the SM receives an AUTH frame in response to its own AUTH frame:

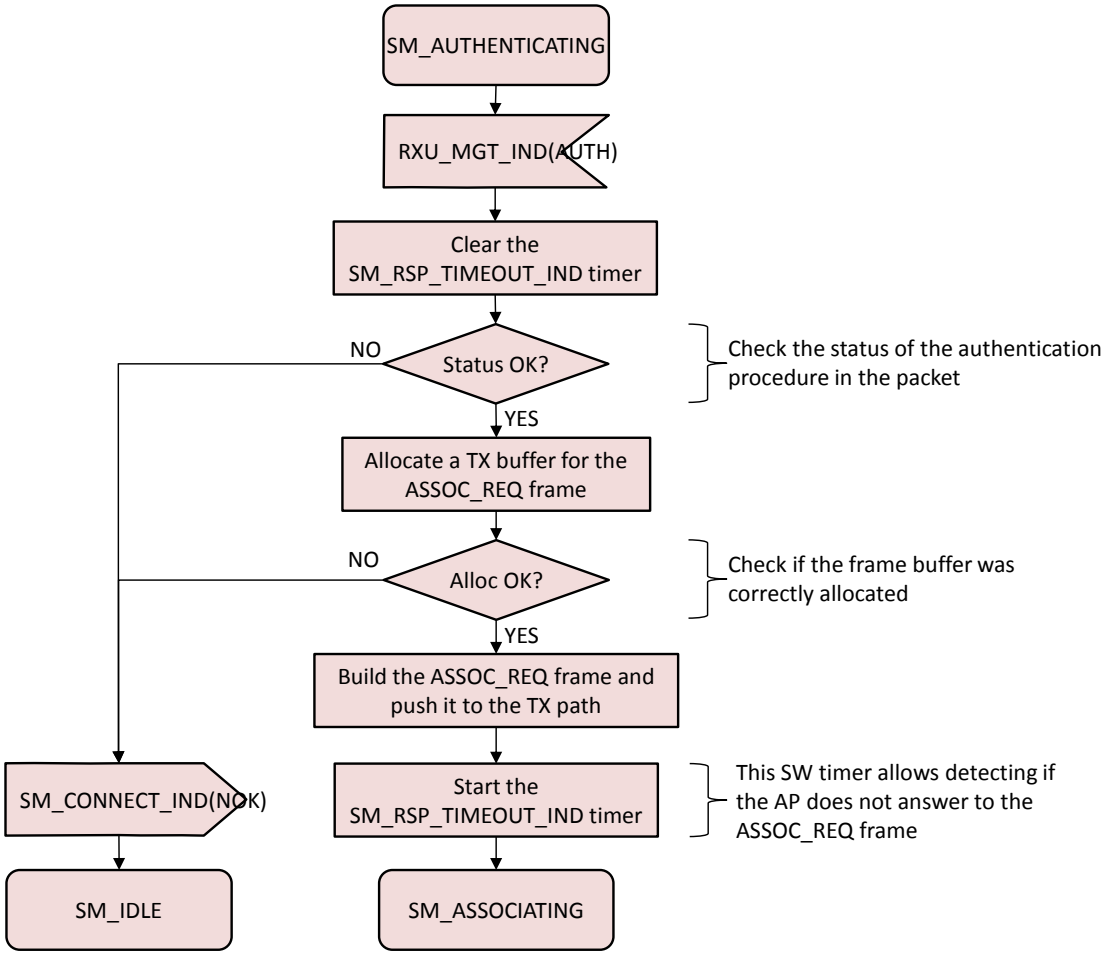

Figure 15: SM_AUTHENTICATION state - Reception of an AUTH frame

In case no response is received, the response timeout will expire:

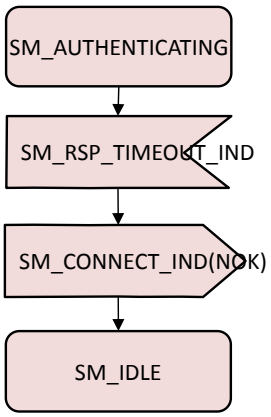

Figure 16: SM_AUTHENTICATION state - Response timeout

2.4.2.2.1.9 SM_ASSOCIATING state

This state is the final one before the completion of the connection procedure. Different types of messages can be received. In a successful case the SM receives an ASSOC_RSP frame in response to its own ASSOC_REQ frame:

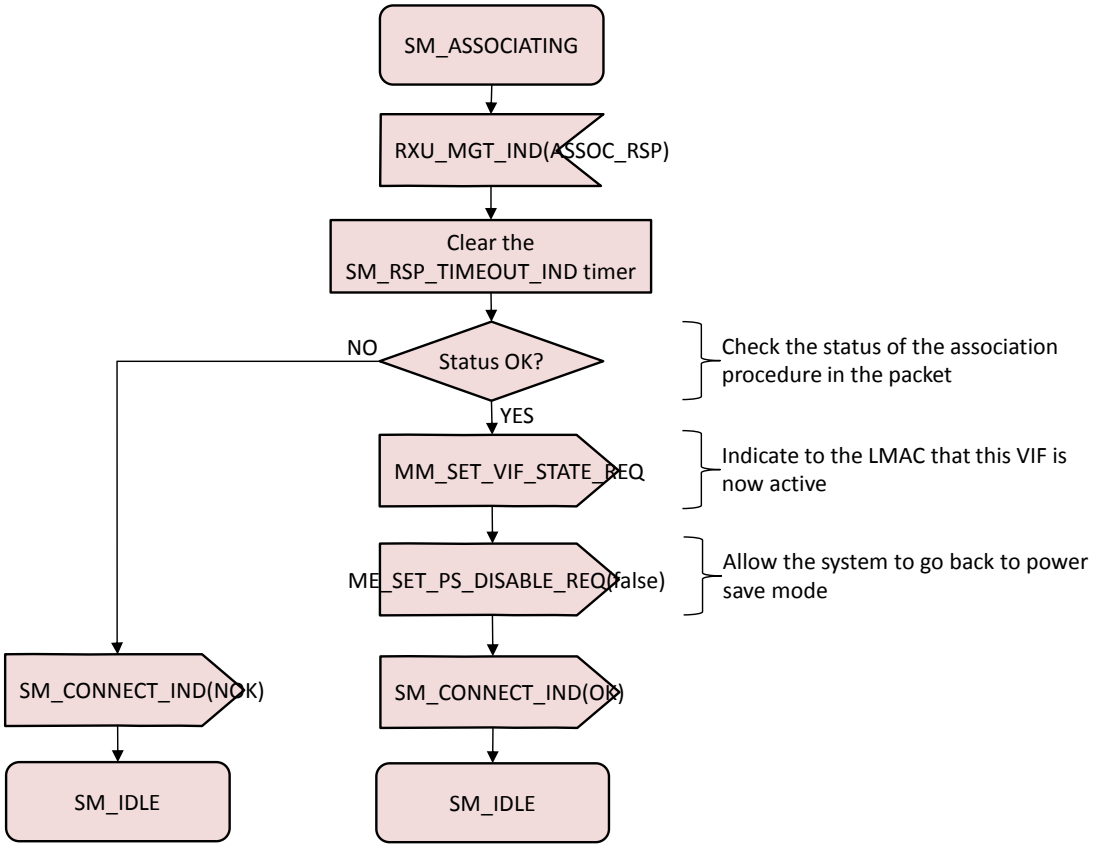

Figure 17: SM_ASSOCIATING state - Reception of an ASSOC_RSP frame

2.4.2.2.2 Disconnection procedure

2.4.2.2.2.1 SM_DISCONNECT_REQ handling

The disconnection procedure can be invoked by the host using the SM_DISCONNECT_REQ message. The handling of this message is done as follows:

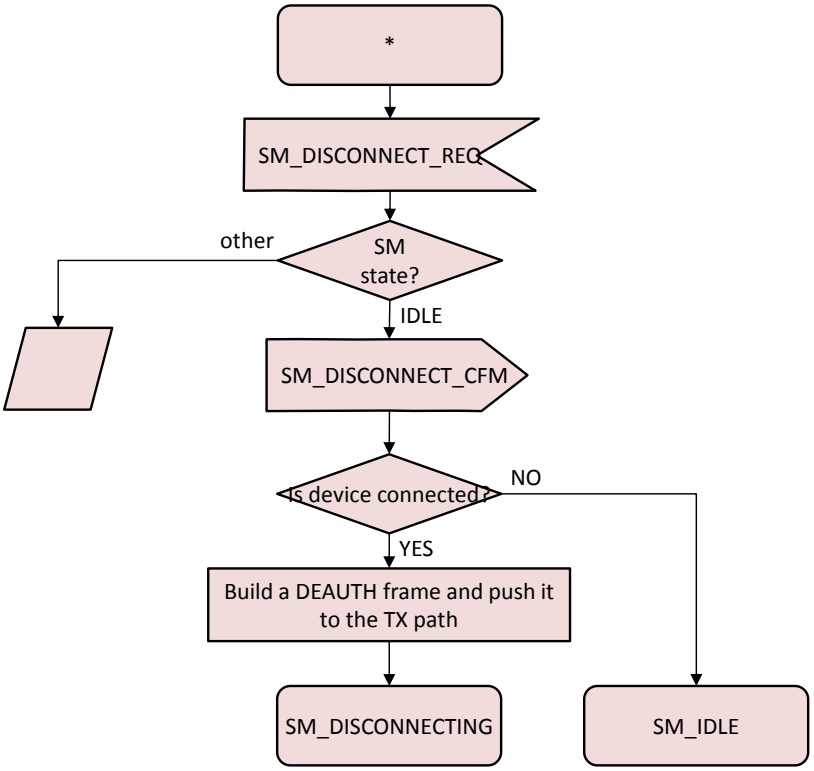

Figure 18: SM_DISCONNECT_REQ handling

The DEAUTH frame transmission will then be confirmed by a call to a callback function registered to the TX path when pushing the frame. The operations performed in this call back are the following:

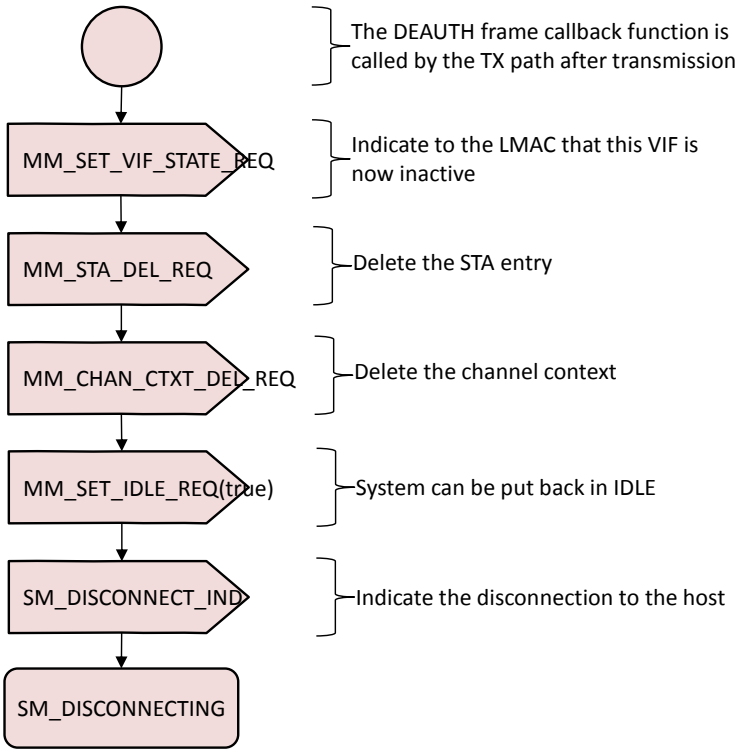

Figure 19: DEAUTH frame callback operations

2.4.2.2.2.2 Disconnection initiated by the peer

The disconnection procedure can also be invoked by the peer device. In that case a DEAUTH packet is received:

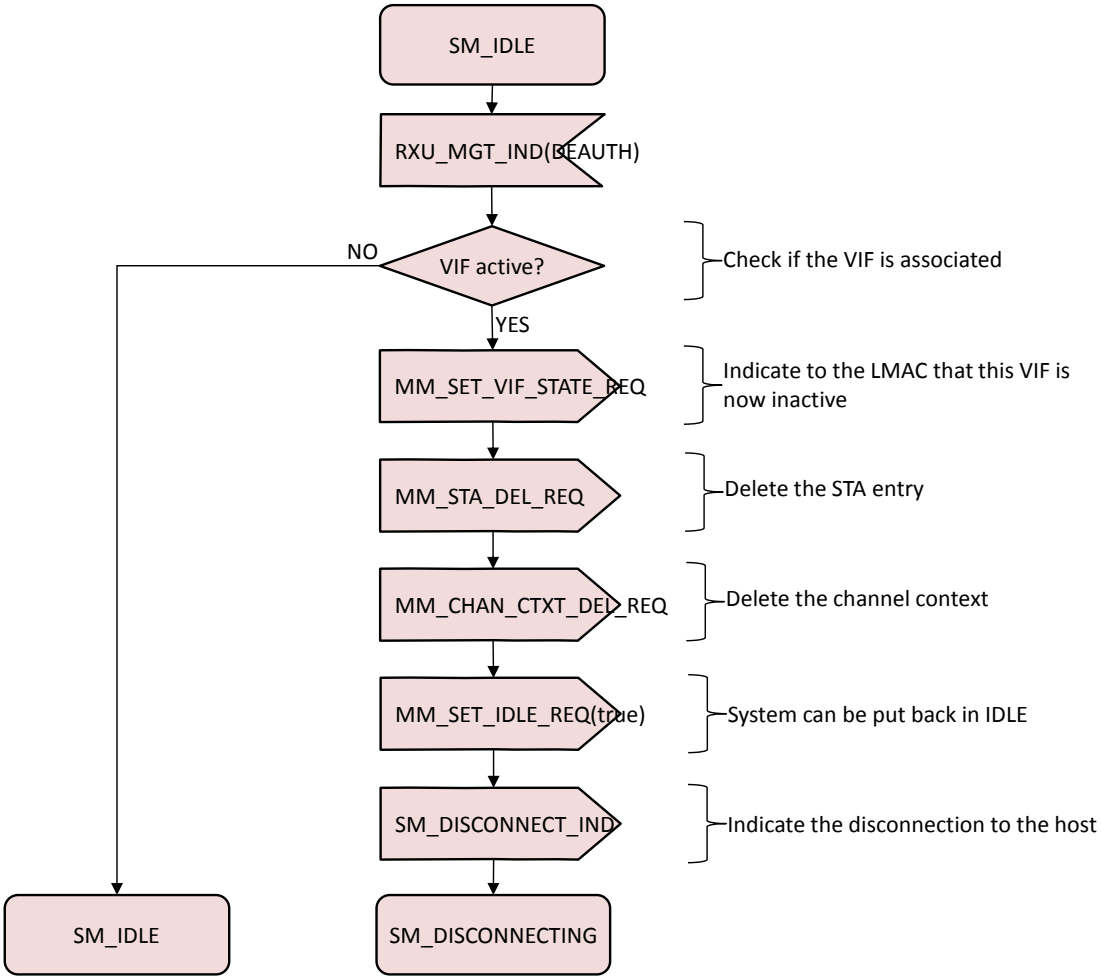

Figure 20: Disconnection initiated by the peer

2.4.2.2.2.3 Disconnection detected locally

The disconnection procedure can finally be invoked by the LMAC, when the connection monitoring module detects that the Access Point is not present anymore:

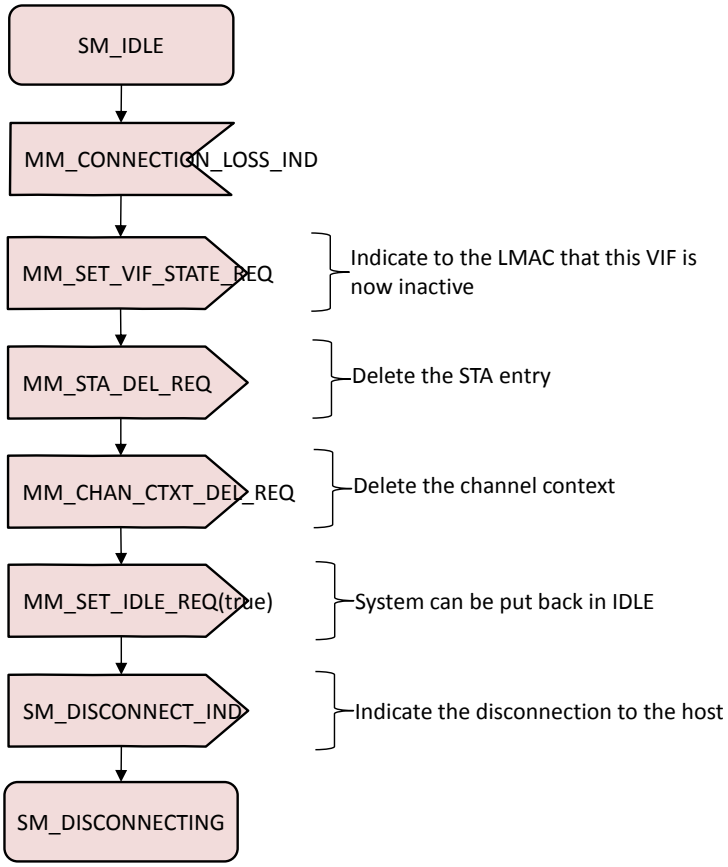

Figure 21: Disconnection detected locally

2.4.2.2.2.4 End of disconnection procedure

The completion of the disconnection procedure is triggered by the reception of the MM_SET_IDLE_CFM message while in SM_DISCONNECTING state:

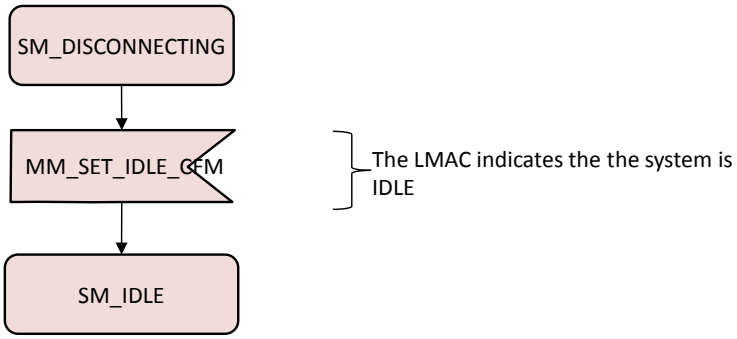

Figure 22: End of disconnection procedure

2.4.2.3 Access Point Manager (APM)

This block is responsible for starting an Access Point on a dedicated interface. The major steps of such a procedure are the following:

- ✓ Create the channel context that will be used for the BSS
- ✓ Set the BSS parameters
- ✓ Schedule the channel context previously created
- ✓ Setting the beacon parameters to the LMAC

The AP Manager also manages the Access Point termination.

This block is implemented as a kernel task called APM that interfaces with the host system to get the AP starting/stopping requests. The APM task implements the following states:

- ✓ APM_IDLE: no procedure is ongoing
- ✓ APM_CHAN_CTX_ADDING: the APM has requested the addition of a channel context to the LMAC
- ✓ APM_BSS_PARAM_SETTING: the APM has sent the BSS parameters to the LMAC and waits for the completion
- ✓ APM_SCHEDULING_CHAN_CTX: the APM has requested the LMAC to schedule the channel context previously allocated.
- ✓ APM_BCN_SETTING: the APM has pushed the beacon information to the LMAC

2.4.2.3.1 Access Point starting procedure

2.4.2.3.1.1 APM_START_REQ handling

The AP is created by the host using the APM_START_REQ message. The handling of this message is done as follows:

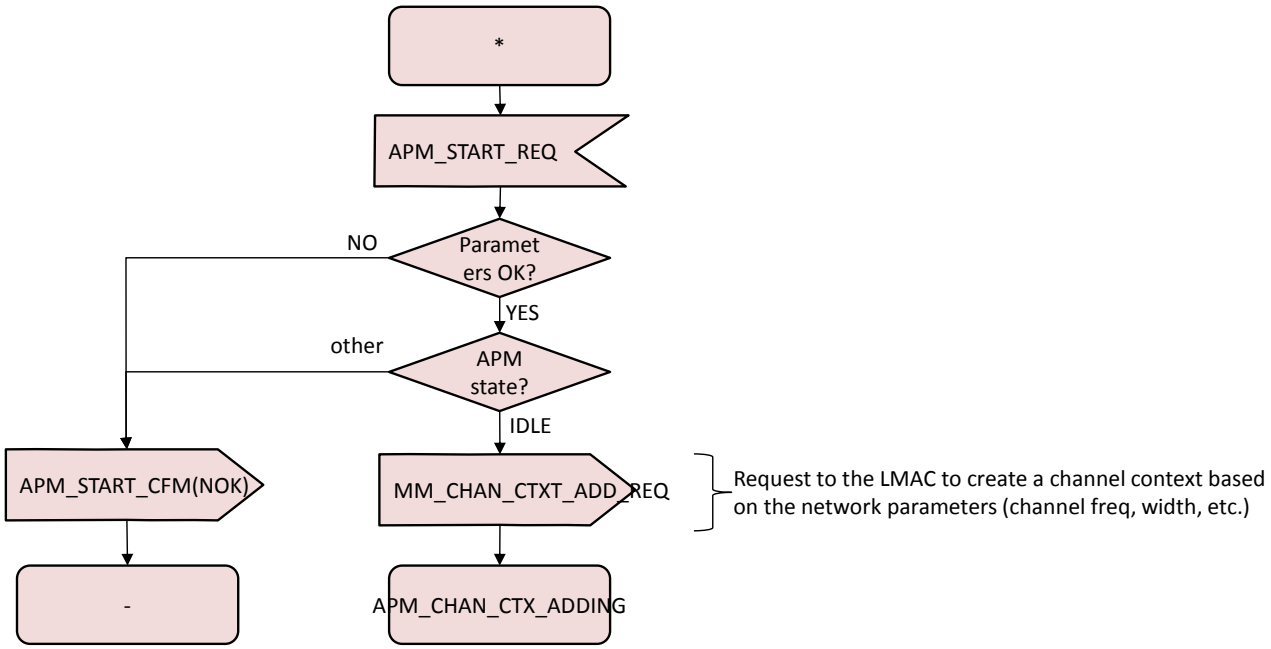

Figure 23: APM_START_REQ handling

2.4.2.3.1.2 APM_CHAN_CTX_ADDING state

In the APM_CHAN_CTX_ADDING state, the APM is waiting for the confirmation from the LMAC that the channel context required for the AP creation has been added:

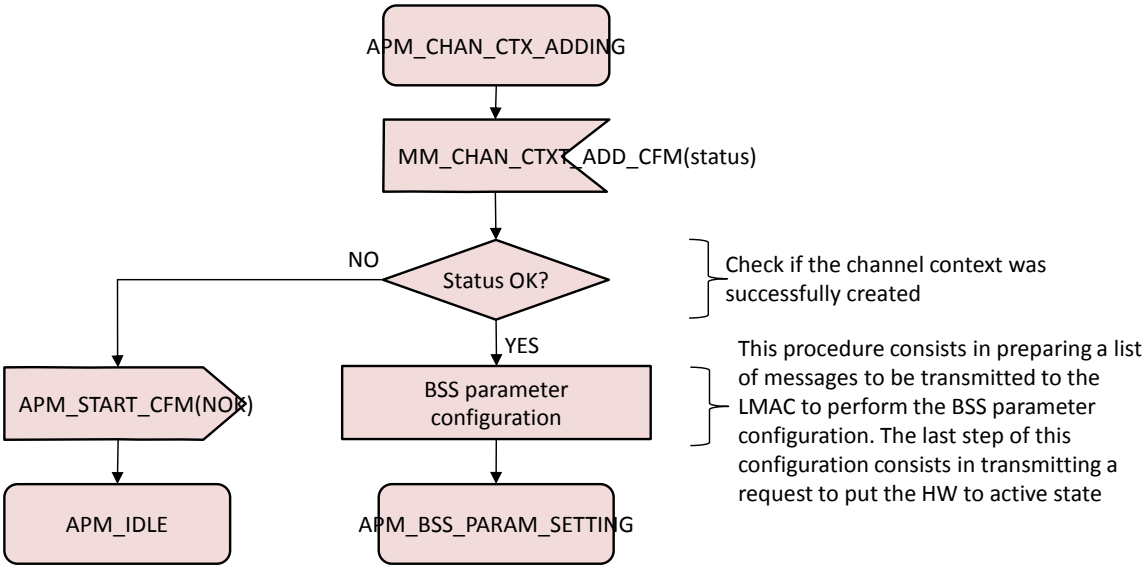

Figure 24: APM_CHAN_CTX_ADDING state

2.4.2.3.1.3 APM_BSS_PARAM_SETTING state

The APM will stay in this state until the completion of the parameters setting. Each time a BSS parameter setting confirmation is received, the next parameter is sent to the LMAC (the message is popped from the list that was built earlier):

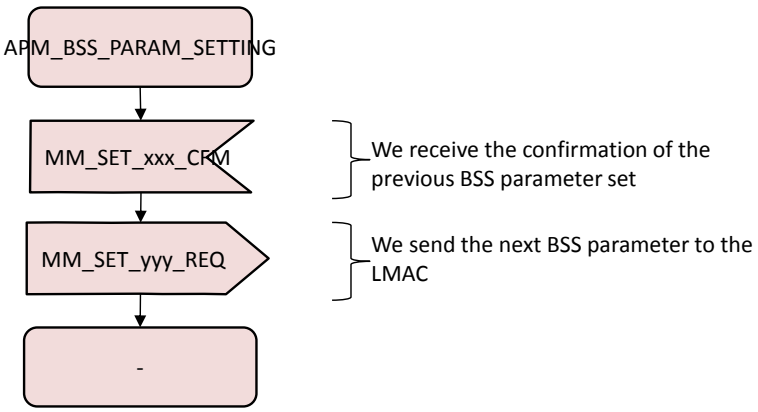

Figure 25: APM_BSS_PARAM_SETTING state – Next BSS parameter setting

Once the BSS parameters have been set and the system is put to active mode, the APM requests the LMAC to schedule the channel context that was previously allocated in order to get a "window" of availability on this channel to proceed to the association procedure:

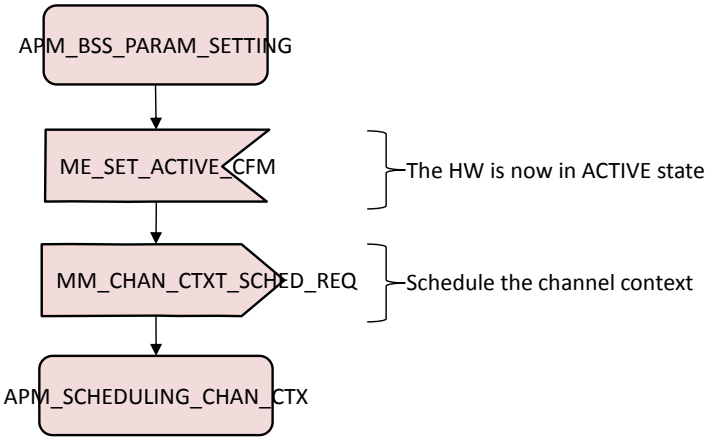

Figure 26: APM_BSS_PARAM_SETTING state – HW set to active

2.4.2.3.1.4 APM_SCHEDULING_CHAN_CTX state

Upon the indication of the end of channel scheduling by the LMAC, the APM sets the beacon parameters to the LMAC:

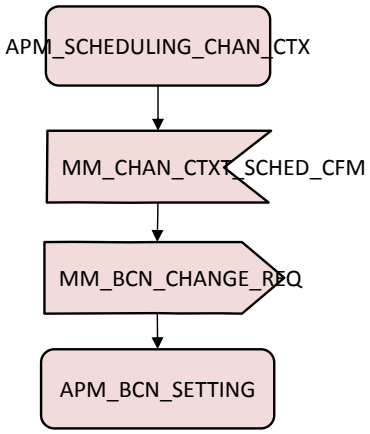

Figure 27: APM_SCHEDULING_CHAN_CTX state

2.4.2.3.1.5 APM_BCN_SETTING state

Once the beacon has been set to the LMAC, the APM terminates the starting procedure by setting the VIF to active, and it confirms the procedure to the host:

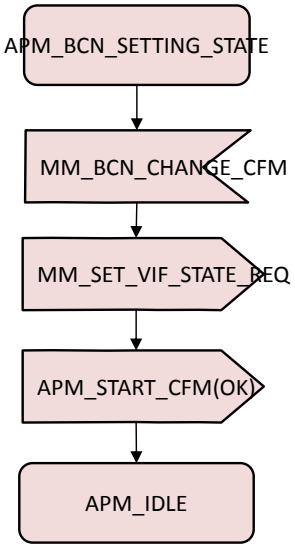

Figure 28: APM_BCN_SETTING state

2.4.2.3.2 Access Point stopping procedure

2.4.2.3.2.1 APM_STOP_REQ handling

When the Access Point needs to be stopped, the host sends the APM_STOP_REQ message to the APM task:

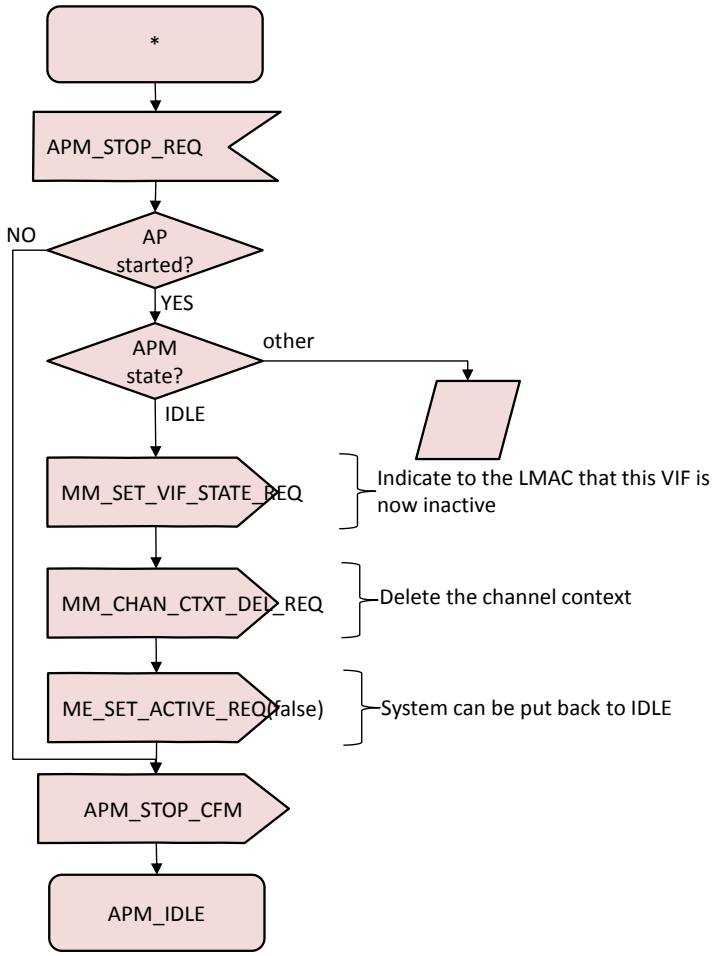

Figure 29: APM_STOP_REQ handling

2.4.3 Transmit Path

The Transmit Path of the UMAC SW is responsible for converting the Ethernet like frames received from the host into WiFi frames. It then passes the frame to the LMAC SW that will proceed to the chaining to the MAC HW.

The TX Path has been designed to minimize the buffer memory used during a packet transmission. In order to allow this, the payload data is downloaded from the host memory in the last steps of the transmission

The different steps in which the UMAC TX path is involved are the following:

- ✓ Get the transmit descriptors from the host, classified on an Access Category (AC) basis
  - ➢ Perform the logical port filtering
  - ➢ Compute what will be the frame length, based on the information present in the TX descriptor
  - ➢ Decide whether the frame could be part of an A-MPDU or not
  - ➢ Create BlockAck agreement with the peer device
  - ➢ Pass the TX descriptor to the LMAC SW (see [3] for more information about the operations performed by the LMAC)
- ✓ The LMAC, after allocating a buffer for the frame but prior to payload download, calls a UMAC TX path function that will build the frame headers
  - ➢ The MAC header
  - ➢ The security header if required
  - ➢ The LLC/SNAP header
- ✓ Once the payload has been downloaded, the LMAC calls another UMAC function to optionally compute the TKIP MIC (only if the frame is encrypted with TKIP)

The figure below gives an overview of the major steps of a transmission.

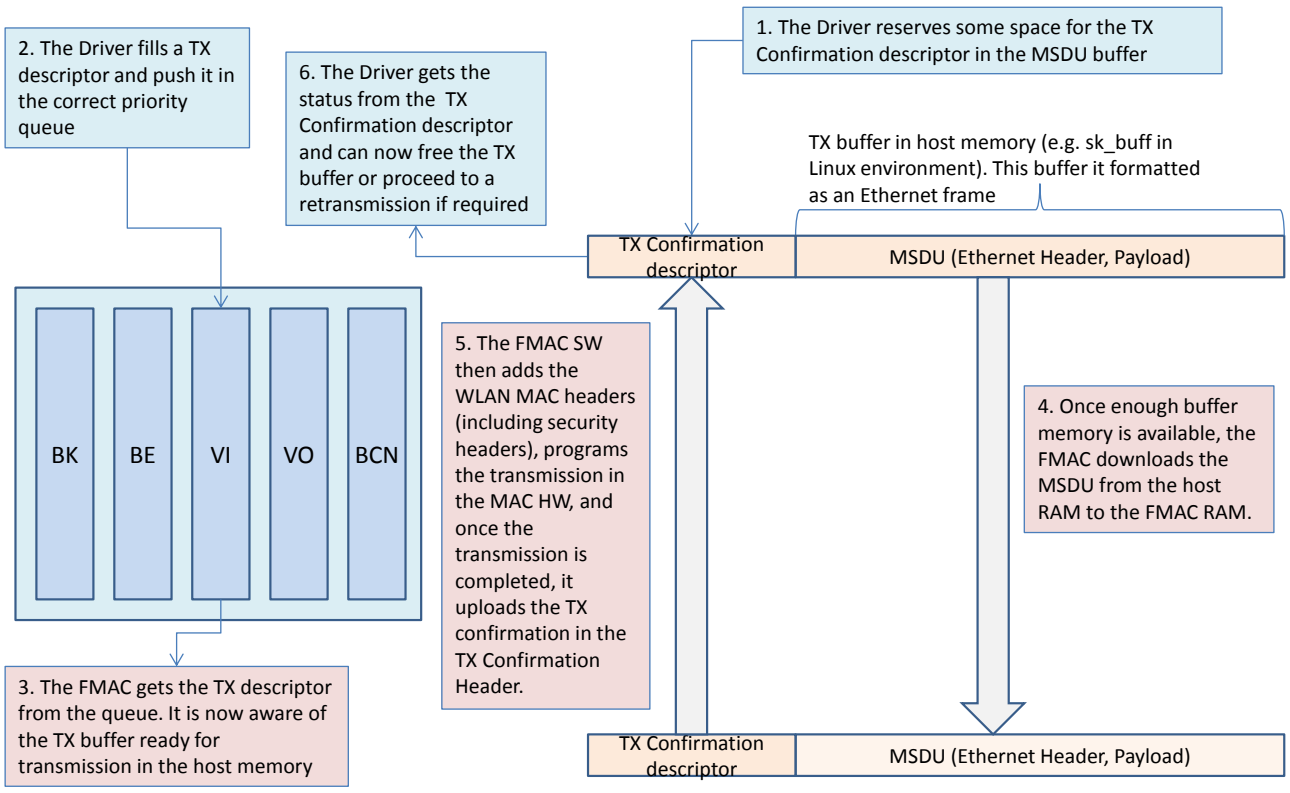

Figure 30: Overview of a transmission

2.4.4 Receive Path

This module processes the frames forwarded by the LMAC SW. The receive process is mainly composed of the following actions:

- ✓ Perform duplicate filtering in order to filter out the frames already received.
- ✓ Perform the reordering of the frames received under a BlockAck agreement
- ✓ Perform the reassembly of the fragmented frames
- ✓ Perform the security checks (replayed frames, TKIP MIC)
- ✓ For management frames that are received, route them to the UMAC tasks using the RXU_MGT_IND message
- ✓ For data frames, convert their format from WiFi to Ethernet

These actions are performed from a function (*rxu_cntrl_frame_handle()*) called by the LMAC SW in the RX kernel event. This function returns a status indicating to the LMAC if it shall proceed to the upload of the packet to host memory or not.

A typical reception is handled in two steps. The first step consists in uploading the frame to a host buffer. This upload is done after the frame conversion to Ethernet format.

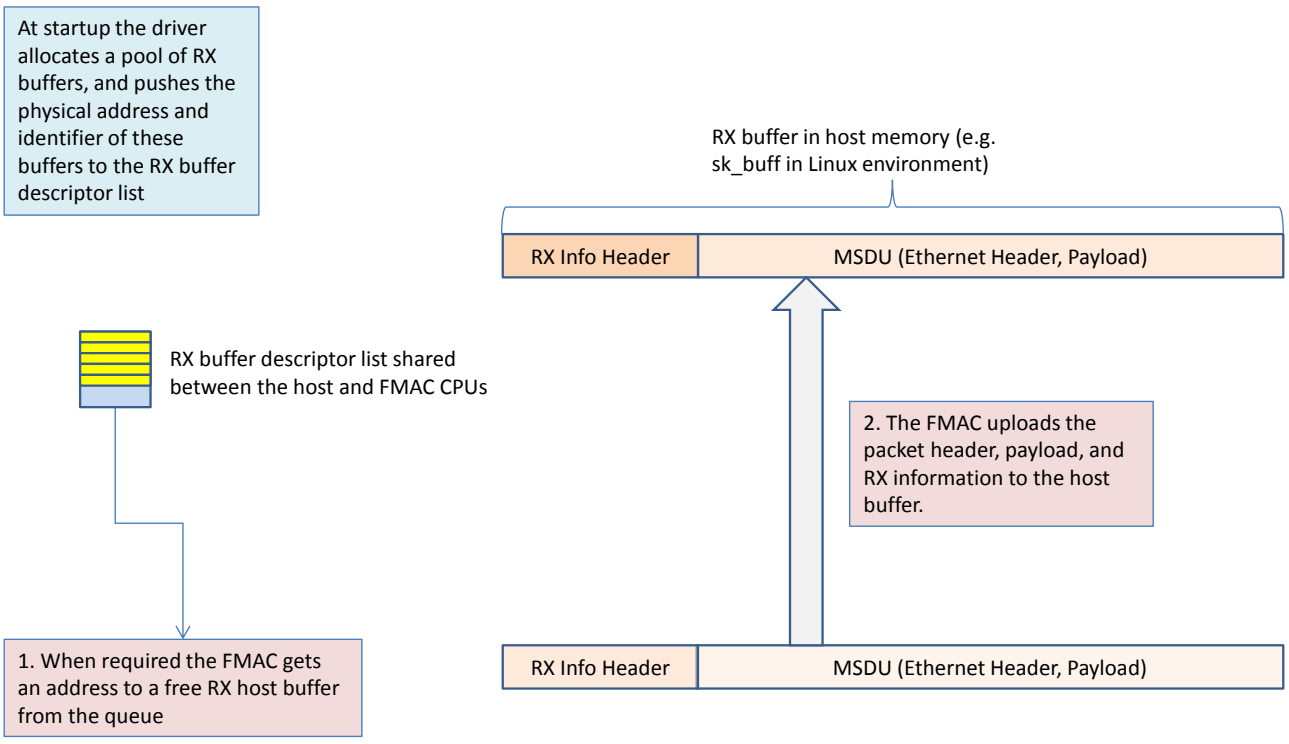

Figure 31: RX flow – Payload upload

A second step consists in uploading a RX descriptor to the host to indicate that a RX buffer has been updated, and which actions shall be performed on this buffer (e.g. forward it immediately, keep it for some time, etc.).

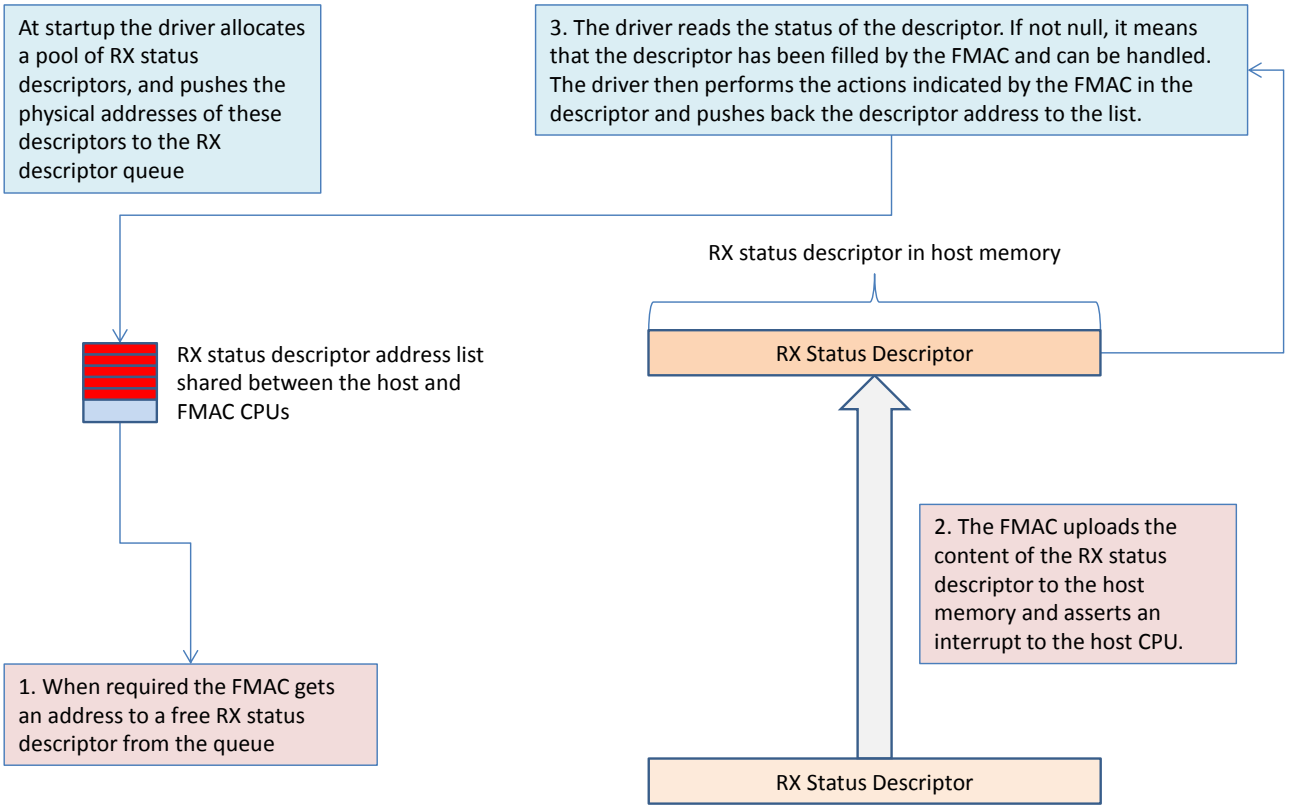

Figure 32: RX flow - RX descriptor upload

This 2-steps approach allows uploading the payload data as soon as possible, in order to reduce the amount of memory required on the FMAC SW. The uploaded frames are then forwarded to the upper layers by the host driver only upon the reception of RX descriptor indicating it has to do so.

The different procedures performed by the UMAC RX path make use of this 2-step approach. As an example, for the reordering, the payloads are uploaded to the host memory as soon as they are received, i.e. in the order of arrival, but the forwarding to the upper layers is done via the sending of the RX descriptors in the correct order.

2.4.5 Rate Control

The rate control algorithm is designed in the aim to be small in code and memory size. It is based on per-STA statistics retrieved from the A-MPDU and singleton MPDU transmissions performed by the MAC.

2.4.5.1 Data structures

2.4.5.1.1 Buffer control structure

This structure includes the information that will be used by the MAC HW to know the parameters of the transmission (i.e rates to be used, protection, bandwidth, etc.). It is composed of the following information:

- ✓ The policy table used for the transmission (see [5] for more information)
- ✓ The MAC control information (to be written in the MAC Control Information 1 field of the TX Header Descriptor, see [5])
- ✓ The PHY control information (to be written in the PHY Control Information field of the TX Header Descriptor, see [5])

This structure is located in the shared memory (because the MAC HW needs access on it), and two of them are allocated per STA, in order to be able to modify the transmission parameters on one structure while some transmissions are ongoing using the other structure.

2.4.5.1.2 Statistics and control

The RC keeps in SW RAM a per-STA structure containing the TX statistics and control information about the Rate Control. This structure is composed of the following information:

- ✓ Two pointers to the buffer control structures used for the transmissions (see above)
- ✓ A pointer to the STA statistics used by the rate control algorithm
- ✓ The frame protection information currently in use with this STA
- ✓ The preamble type currently in use with this STA
- ✓ A bitfield indicating if the policy table needs to be updated, and which part of it needs to be updated
  - ➢ This bitfield indicates for example if some protection has to be enabled/disabled, if the rate has to be changed, if the transmission bandwidth has to be changed, etc.
- ✓ The index of the buffer control structure currently in use for the transmissions to this STA

The STA statistics structure is composed of the following information:

- ✓ The last time the Rate Control has run.
- ✓ An array of 10 statistics structures with the following information:
  - ➢ The number of frames transmitted in the rate control period.
  - ➢ The number of successful frames in the rate control period.
  - ➢ The estimated probability of success.
  - ➢ The rate configuration, including format and modulation, guard interval, preamble type, bandwidth, number of spatial streams, MCS / rate index.
  - ➢ A counter indicating how many times the sample has been skipped in the probability of success calculation because of lack of attempts.
  - ➢ A flag indicating if the probability of the previous rate control period is available.
  - ➢ A counter indicating the number of times the AMPDU has been retried using this sample.
  - ➢ A flag indicating if the rate can be used in the retry chain.
- ✓ Four step structures containing the calculated throughput and the index of the rate in the sample table of the four steps of the retry chain.
- ✓ A step structure containing the calculated throughput and the index of the rate in the sample table of step 0 or step 1 of the retry chain, updated when these are replaced by the trial rate.
- ✓ The number of MPDU transmitted in the rate control period.
- ✓ The number of AMPDU transmitted in the rate control period (in case of aggregation disabled, this field is the number of MPDU transmitted).
- ✓ The EWMA average number of MPDU in each AMPDU frame.

- ✓ The number of transmissions to be done before the next trial transmission.
- ✓ A counter used to regulate the use of lower rates for trial transmission.
- ✓ The status of the trial transmission.
- ✓ A bitfield indicating the position of the trial rate in the retry chain (step 1 or step 2), whether next transmission could be part of a AMPDU, whether the next step of the retry chain has to be selected (SW retry with aggregation).
- ✓ The step of the retry chain that has to be set as first step.
- ✓ The format and modulation supported by the STA.
- ✓ A union representing the MCSs which can be used by the rate control:
  - ➢ Four per-spatial stream HT MCS
  - ➢ The VHT MCS map
- ✓ The legacy rate set that can be used by the rate control.
- ✓ The maximum MCS that can be used by the rate control.
- ✓ The minimum legacy rate index that can be used by the rate control.
- ✓ The maximum legacy rate index that can be used by the rate control.
- ✓ The maximum bandwidth that can be used by the rate control.
- ✓ The maximum number of spatial streams that can be used by the rate control.
- ✓ Whether the short guard interval can be used by the rate control.
- ✓ Which preamble type can be used by the rate control (0: short and long preamble, 1: only long preamble).
- ✓ The number of samples to be used in the sample table.
- ✓ The maximum AMSDU size.

2.4.5.2 Rate Control Steps

2.4.5.2.1 Algorithm overview

The Rate Control algorithm uses the statistics retrieved during a fixed period of 100 ms for calculating the throughput and the probability of success for each rate of the subset taken into account. These values are used for populating the retry chain table which is used in the next 100 ms period.

The subset of rates used in the algorithm is updated every 100 ms, after the throughput and probability calculation.

In order to test some other rates in addition to the retry chain ones, one out ten transmissions is a look around: a random rate chosen in the sample table is inserted in the retry chain in place of step 1 or 2.

2.4.5.2.2 Initialization

The per-STA rate control information is initialized upon a new STA addition to the MAC (either the AP we connect to or a STA connecting to our AP). At this stage, based on the new STA parameters, the parameters of the RC algorithm are set. The sample table is initialized as follows: sample 1 is the lowest allowed rate, last sample is the highest allowed rate with max MCS equal to 7, the other samples are randomly chosen between all the allowed rates, limiting the bandwidth to the 2 highest bandwidths.

Steps 1 to 3 of the retry chain are set starting from the last sample of the table (highest rate); step 4 is set to the lowest rate.

The statistics (number of failed frames, total number of frames, probability and throughput) are reset.

The number of samples is set based on the maximum number of allowed rates, with maximum number set to 10 samples.

Once those initializations are done, the buffer control information structures are initialized based on the retry chain set in the RC algorithm.

2.4.5.2.3 Frame transmission preparation

Each time a frame is pushed to the FMAC FW, the RC is invoked to check if the retry chain has to be modified. The rates used in the retry chain can be changed upon several events:

- ✓ The RC interval (100 ms) is expired: the new values of throughput and probability of success are calculated and the retry chain consequently updated. The less useful samples of the sample table are removed and new ones inserted.
- ✓ The trial period is passed: a random rate is selected from the sample table and inserted in the retry chain, in place of step 1 or 2.
- ✓ The maximum number of SW retries has been reached: the next step of the retry chain is selected for the following transmission.

In case the retry chain has to be updated, the following values are updated accordingly:

- ✓ The aggregation flag, which disables the aggregation and consequently enables the use of legacy rates: the aggregation is disabled when low MCSs are reached (<= MCS2) and probability of success is low (<10%), or when a legacy rate has been inserted in the retry chain.
- ✓ The maximum AMSDU size, which is set depending on the number of spatial streams (1 spatial stream: max size is 2; 2 spatial streams: max size is 4; otherwise max size is 6).

2.4.5.2.3.1 RC statistics update

Every time the RC runs, it calculates for each of the samples of the sample table the following values:

- ✓ The exponentially weighted moving average of the probability of success. In case no transmission attempts have been done in the previous RC interval, the sample is skipped and the corresponding counter is updated.
- ✓ The estimated throughput calculated for an AMPDU composed by 16 packets of 1200 bytes and multiplied by the probability of success. If the probability of success is less than 10%, the throughput is set to 0.

The rates with maximum throughput, second maximum throughput and maximum probability of success are used as steps 1-3 of the retry chain.

The second phase of the RC algorithm is the selection of new rates to be inserted in the sample table. The new rates are inserted in the following order:

- ✓ Random rate, chosen between all the allowed rates and different from the ones already in use, with bandwidth limited to the two highest.
- ✓ Higher throughput rate (first step) with inverted guard interval, if supported.
- ✓ MCS index / rate index above the higher throughput rate (first step), with short guard interval if supported.
- ✓ MCS index / rate index below the higher throughput rate (first step), with short guard interval if supported.
- ✓ MCS index / rate index above the second higher throughput rate (second step), with short guard interval if supported.
- ✓ MCS index / rate index below the second higher throughput rate (second step), with short guard interval if supported.

The rates removed from the sample table are the ones with probability of success less than 50% or which have been already skipped several times (10), excluded the rates already in use in the retry chain. The rates are removed starting from the one with lower throughput.

2.4.5.2.3.2 Trial transmission

Every trial period, a trial rate is randomly picked from the sample table and, if it is expected to provide a greater throughput, is inserted in the retry chain and the policy table updated.

In order to determine if the random rate could provide a better throughput, its transmission duration, calculated for a packet of 1200 bytes, is compared to the transmission duration of the first step and second step of the retry chain. The rates frequently skipped are used even if their throughput is not expected to be greater.

Once the trial rate is selected, this is inserted in the retry chain in place of the first step if its SW retry requests limit has not been exceeded; otherwise it is inserted in place of the second step.

The values of the step overwritten are copied in the max_tp_2_trial structure, the info field updated with the number of the step overwritten and the status of the trial transmission updated. The trial period value is updated: for HT STAs the value is directly proportional to the average AMPDU size, for NON-HT STAs the value is set to 10 or, in case the probability of success is less than 10% or greater than 95%, is set to 5.

In the next transmission the retry chain table is reverted, the info field and the trial status are updated.

2.4.5.2.3.3 SW retries

When the packet is part of an AMPDU only the first step of the retry chain is used by the HW for the transmissions. In case one or more frames of an AMPDU should be retried, the counter of the number of requests of retry is incremented, and if it exceeds the maximum number set, the RC selects the next step of the retry chain and updates the policy table accordingly. The maximum number of requests of retry for a rate is equal to its MCS index.

2.4.5.2.4 Frame confirmation

Each time a packet has been transmitted, the statistics (total count, failed count, SW retries count) are updated according to the transmission status.

In case of confirmation of a singleton, the total number of attempts and failures is distributed among the four steps of the retry chain, according with the maximum number of retries set in the policy table.

In case of confirmation of an AMPDU, the total number of attempts and failures is added only to the current first step of the retry chain.

Moreover, the trial countdown, the trial status and the SW retry request flag are updated.

References

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE P802.11-2012

[2] RW-WLAN-nX-FMAC-SW User Manual Document (RW-WLAN-nX-FMAC-SW-UM/0.03)

[3] RW-WLAN-nX-LMAC-SW Functional Specification (RW-WLAN-nX-LMAC-SW-FS/1.01)

[4] RW-KERNEL-SW Functional Specification Document (RW-KERNEL-SW-FS/1.1)

[5] RW-WLAN-nX-MAC-HW User Manual Document (RW-WLAN-nX-MAC-HW- UM/1.01)