IEEE 802.1 Security MACsec Privacy Frame Stats Review

Don Fedyk – don.Fedyk@labn.net

Disclaimer

- This is a work in progress. The material here is for discussion purposes and may contain errors.

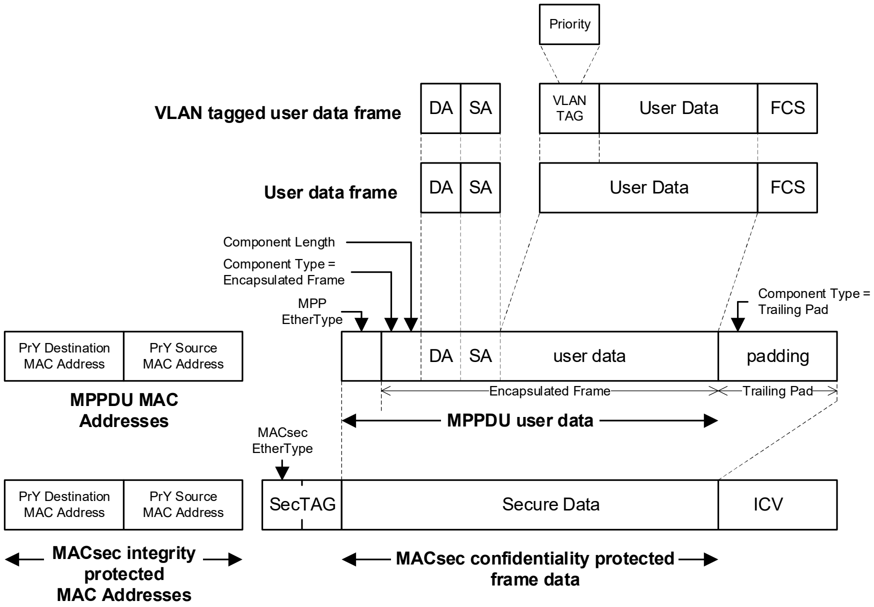
Configuration parameters for MAC Privacy 802.1AEdk

- How big should an MPPDU be?
  - Examples Showing Encapsulation Arithmetic

Determining Frames Size and Rates for MAC Privacy Channels and Frames

- PrY Channel Frames:
    - Fragmentation
    - Fragmentation is default enabled. Default is on.
        - Allows Higher efficiency, (allows late addition) – not in the standard.
    - Setting an MPPDU too small can force fragmentation when Max size user frames are encountered.
    - Determine the maximum user frame size “User Data Frame size”.
- PRY Frames
    - No Fragmentation
    - Determine the maximum user frame size “User Data Frame size”.
    - This must be greater that or equal to the User Data Frame size or larger frames could be dropped.

MAC Privacy – Which Length?



This is the source of the defining length

MAC Privacy

MAC Privacy & MACsec

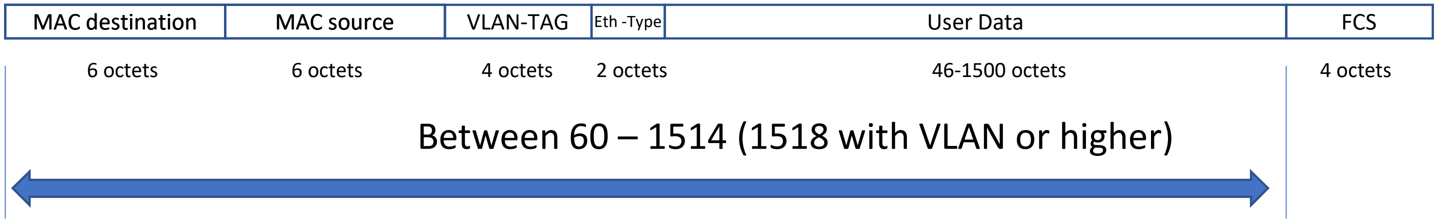
Clause 17 Draft figure

Standard Ethernet Frames

Standard Ethernet encapsulation:

- Frame sizes are dependent on media
- Ethernet Standard are 1500 octets of user data
- Ethernet Jumbo is 9000 octets of user data
- Uses the Media overhead bytes

Goal is to determine the MTU for the situation



MAC destination	MAC source	VLAN-TAG	Eth -Type	User Data	FCS
6 octets	6 octets	4 octets	2 octets	46-1500 octets	4 octets

Between 60 – 1514 (1518 with VLAN or higher)

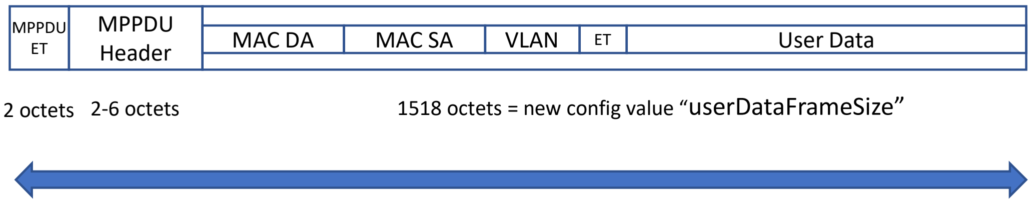
User data no FCS
a.k.a. L2 MTU

MAC Privacy allows for an unfragmented Max User data frame
This means encapsulating nominally up to 1518 octets but possibly higher.
Other formats (e.g. LLC, SNAP) are supported as well but are less than or equal to 1500 octets. IEEE 802.3 allows up to 2000 for envelope frames 802.3as-2006

MAC Privacy MPPDU

- MAC Privacy Encapsulation adds
- 2 octets for Fixed Full frames
- +4 octets for a fragment – but that can be absorbed since fragments are variable
- Therefore 2 octets plus user data frame 1518 + 2.

userDataFrameSize



2 octets   2-6 octets   1518 octets = new config value “userDataFrameSize”

1520 octets (2 octets overhead counts as padding)

This is not a frame. A PrY Frame in non collocated PrY adds 16 octets more.

User data octet count

- 1500 bytes of Ethernet Frame User data is:
- 1518 User data octets for MAC Privacy – but-
- 1520 bytes are needed for an unfragmented 1518 bytes.
- From a statistics collection perspective these 2 bytes are padding and they belong to the MAC PrY padding count. There are no stats for MPPCI header overhead – they are collected as padding. The rational is that empty frames are all padding.
- However, when comparing MAC PrY to MACsec for the same frames if multiple frames are carried in an MPPDU there is a next savings of overhead.

Summary

- Only need to configure the PrY L2 MTU e.g. 1518 or similar (e.g. 9018)
    - This number is whatever the base traffic user traffic format is for example PBB frames would use a larger number.
    - MPPDU length will be fixed at 1518 +2 octets. (1520)
    - The Frame is assumed to carry 1500 bytes – often this value may never be met with traffic for example 1492 might be the IP MTU.
- As far as MAC privacy this value 1518 is the configurable encapsulation payload.
    - userDataFrameSize, user-data-frame-size
- If this number is configured smaller than source user traffic, some large frames may be fragmented – for channels.
- This number must be supported for privacy frames or the 1500 (9000) Frame Payload of the user frame is impacted.
- If a smaller payload is required for other reasons for PrY channels this number can be adjusted downwards this guideline is merely to prevent fragmentation of whole frames, but implementations may fragment anyway in the interest of reducing delay or increasing efficiency.
- With a 64 octet Minimum Fragment the maximum wasted data is 63 octets.

Data Frame Fitting

- User Data Configuration = 1518 (1500 Frame data bytes)
- MPPDU = 2 bytes (Real size 1520)
- Frame Size 1572 Fits a 1500 byte frame with no padding.
- MAC_PRY + MAC_SEC Headers = 52

One MPPDU

```
"2": {
   "1": {
      "length": 1572,
       "frame_data": 1500,
      "mppdu_ovrhd": 2,
      "mac_pry_hdr_icv": 52
   }
}
```

User Data Config 1518
User data octets = 1518
Pad Octets = 2 octets
IETF port stats 1572 (includes 52 bytes encapsulation)

"Perfect Fit"

When User Data config is longer - it spills into extra padding 1520

One MPPDU

```
"2": {
   "1": {
      "length": 1572,
      "serial_num": 0,
      "frame_data": 1500,
      "mppdu_ovrhd": 2,
      "mac_pry_hdr_icv": 52,
   },
   "2": {
      "length": 2,
      "pad_data": 2,
       "padtype": "trailing"
   }
```

User Data Config 1520
User data = 1518
Pad Octets = 4 octets
IETF port stats 1574
(includes 52 bytes encapsulation)

Config larger by 2 octets causes 2 octets additional

At < 1518
Fragmentation occurs for 1500 bytes of frame data

This increased header for fragments pushes up the padding count.

Worst Case Padding is 53 bytes for frame that is long by one octet.

MinFRAG = 64
58 + 6 MPPCI header

User Data Config 1517
User data = 1518
Pad Octets = 53 + 6 + 6 = 65
1455 leftover for next frame
IETF port stats 1571 + 1571
(includes 52 bytes encapsulation/ PrY frame)

More than One MPPDU

```
"1": {
   "1": {
      "length": 1518,
      "frame_data": 1442,
      "mppdu_ovrhd": 6,
      "mac_pry_hdr_icv": 52,
      "express": false,
      "seq": 0,
      "initial": true,
      "final": false
   },
   "2": {
      "length": 53,
      "pad_data": 53,
      "padtype": "trailing"
   }
},
"2": {
   "1": {
      "length": 116,
      "frame_data": 58,
      "mppdu_ovrhd": 6,
      "mac_pry_hdr_icv": 52,
      "express": false,
      "seq": 1,
      "initial": false,
      "final": true
   }
```

Under by 1 octet causes 63 (65 – 2) octets additional. Under by 2 - 62, 3-61, 4-60 etc.

Notes

- Minimum frame fragment of 64 octets is frame data + MPPCI header (6 Octets) .
- 58 octets of frame data + 6 bytes of MPPCI header = 64. This can be larger – 64 octets of frame data yields a minimum fragment of 70 octets. But large is less efficient. (Originally, I had coded 64 octets of user data which gave a 70 octets fragment with 59 octets of padding, but this meant a one octet over causing a fragment would introduce 59+6+6-2 = 69 octets extra.)
- Constants use for computation

```
SMAC = 6
DMAC = 6
VLANTAG = 4
ETHTYPE  = 2
MPPCI = 2
MPPCI_LONG = 6
SECTAG = 8
SCI = 8
ICV = 16
FCS = 4
USER_FRAME_OVERHEAD = SMAC + DMAC + VLANTAG + ETHTYPE  #  = 18
MACSEC_PRY_OVERHEAD = SMAC + DMAC + VLANTAG + SECTAG + SCI + ICV + FCS  # = 52
MACSEC_OVERHEAD = SECTAG + SCI + ICV + FCS # = 38
```

MAC Pry Statistics What get counted where

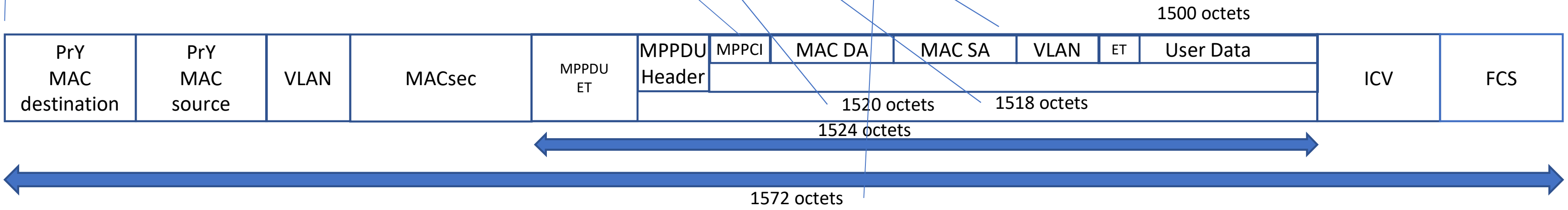
MAC Pry Stats

counter64 outMppdus = 1
counter64 outUserFrames = 1
counter64 outUserOctets = 1518
counter64 outPadOctets = 2
counter64 outUserFragments = 0
counter64 inMppdus = 1
counter64 inErroredMppdus
counter64 inUserFrames = 1
counter64 inUserOctets = 1518
counter64 inPadOctets = 2
counter64 inUserCompleteFragments = NA
counter64 inUserDroppedFragments = NA
counter64 inUserErroredFragments = NA

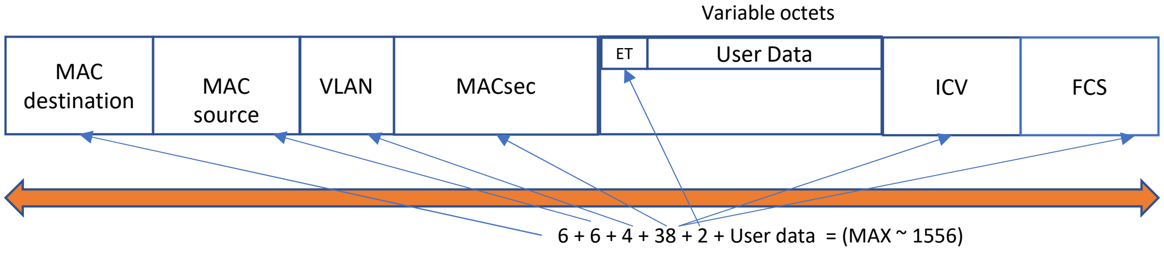
1520

IETF Interface Stats

counter64 inOctets = 1572
counter64 inUnicastPkts = 0
counter64 inBroadcastPkts = 0
counter64 inMulticastPkts = 1
counter32 inDiscards =NA
counter32 inErrors = NA
counter32 inUnknownProtos = NA
counter64 outOctets = 1572
counter64 outUnicastPkts = 0
counter64 outBroadcastPkts = 0
counter64 outMulticastPkts = 1
counter32 outDiscards = NA
counter32 outErrors = NA



Efficiency MACsec & MAC PrY & MACsec

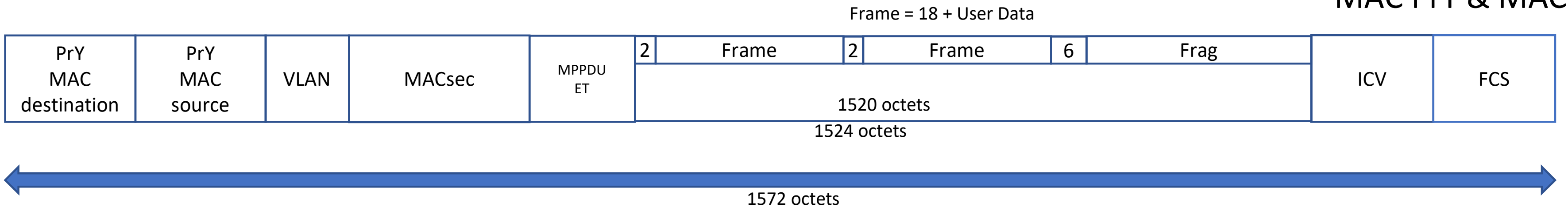


6 + 6 + 4 + 38 + 2 + User data = (MAX ~ 1556)

Overhead is 56 Octets Per User data (38 MACsec 18 Ethernet)

MACsec

MAC PrY & MACsec



Overhead = 1 Frame 1572 – User data
52 + 18 + (2 MPPCI Padding + Other Padding)

Overhead = 2 Frames 1572 – (User data1 + User data2)
52 + 18 + 18 + (4 MPPCI counted as Padding + Other Padding)

Overhead = 3 Frames 1572 – (User data1 + User data2 + User data3)
52 + 18+ 18+18 + (10 MPPCI counted as Padding + Other Padding)

Padding Statistics

- For most traffic mixes MAC PrY has no more overhead/per user data than MACsec alone, but it has padding.
    - It can have less overhead for small frames if MPPDUs are filled
- Padding counts as sent/octets received.
- Currently Pad is composed of:
    - “Trailing PAD” Zero Octets added to an MPPDU
    - “Explicit PAD” Zero Octets
    - MPPCI octets of any Component frame of fragment (Including PAD)
- A similar project for IPSec counts all pad packets and all pad octets separate from padding added to a frame.
- To do something similar, need to consider padOctets into padOctets (mppdus with some user data) and allPadMppdusOctets pure Padded MPPDUs.
- Explicit pad and Trailing pad would not be differentiated. An all pad MPPDU could have either or both.

Received Stats & Padding

```
outMppdus =
outUserFrames =
outUserOctets =
outPadOctets =
outUserFragments =
inMppdus = 250
inErroredMppdus = 0
inUserFrames = 215
inUserOctets = 170516
inPadOctets = 209484
inUserCompleteFragments = 131
inUserDroppedFragments = 0
inUserErroredFragments = 0

Interface stats
inOctets = 393000
```

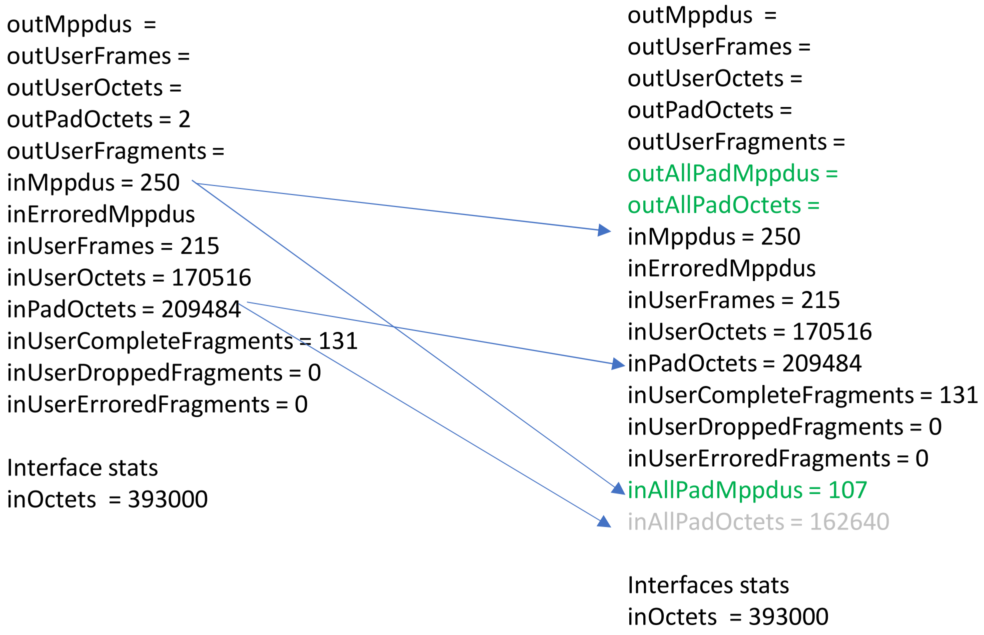
What do we know?

- 250 MPPDUs
- 215 User frames
- 131 User Fragments (~ 2 fragments/frame ~65 frames fragmented)
- 170516 User Octets
- 209484 PAD Octets
- 250 * 1520 = 170516 + 209484 = 380000
- 170516/379500 = 44.9 %
- 1518*250 = 379,500 = 100% (2 octets/frame overhead)
- 1572 * 250 = 393000
- No Errors.

How many MPPDUS carry no data?

Best guess between 137 to 35 = 172 /2 = 86 all PAD?

Finer Grain Padding Stats same example



outMppdus =
outUserFrames =
outUserOctets =
outPadOctets = 2
outUserFragments =
inMppdus = 250
inErroredMppdus
inUserFrames = 215
inUserOctets = 170516
inPadOctets = 209484
inUserCompleteFragments = 131
inUserDroppedFragments = 0
inUserErroredFragments = 0

Interface stats
inOctets = 393000

outMppdus =
outUserFrames =
outUserOctets =
outPadOctets =
outUserFragments =
outAllPadMppdus =
outAllPadOctets =
inMppdus = 250
inErroredMppdus
inUserFrames = 215
inUserOctets = 170516
inPadOctets = 209484
inUserCompleteFragments = 131
inUserDroppedFragments = 0
inUserErroredFragments = 0
inAllPadMppdus = 107
inAllPadOctets = 162640

Interfaces stats
inOctets = 393000

Given AllPadmppdus and UserDataSize
AllPadOctets can be computed.

170516+46844/(250-107) = 1520

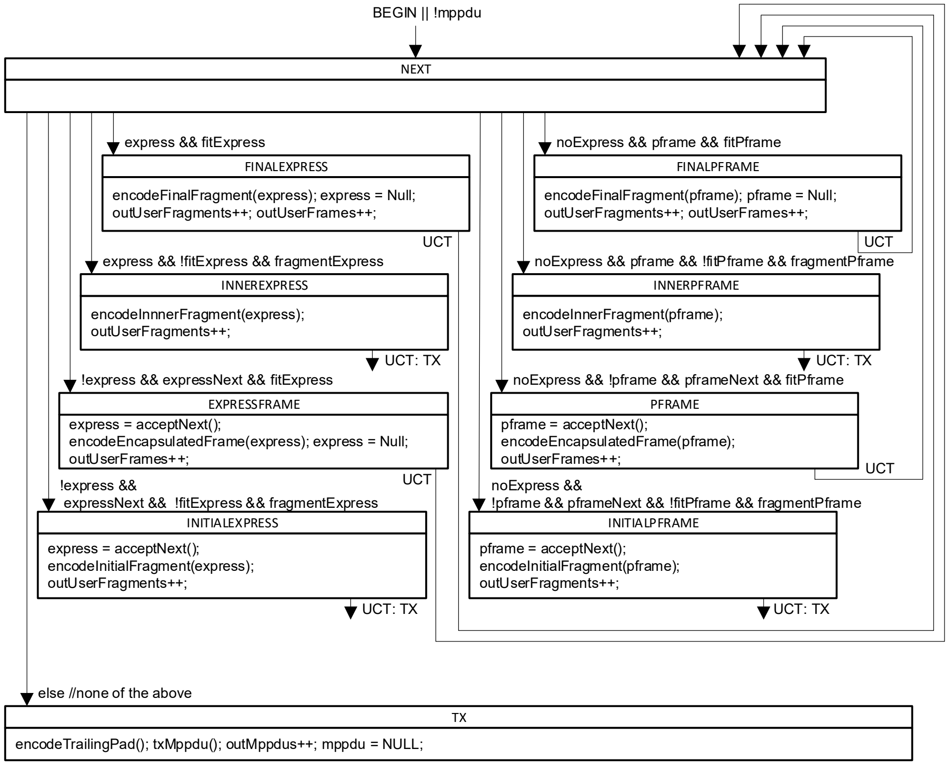
250 -107 = 143 (Mppdus containing data)

Received 107 pure padding MPPDUs
107 * 1520 = 162640

Total MPPDUs * total Frame size
1572 * 250 = 393000

There was actually 107 all Pad MPPDUs – does it matter?

Current State Machines



BEGIN || !mppdu

NEXT

express && fitExpress

FINALEXPRESS

encodeFinalFragment(express); express = Null;
outUserFragments++; outUserFrames++;

UCT

express && !fitExpress && fragmentExpress

INNEREXPRESS

encodeInnnerFragment(express);
outUserFragments++;

UCT: TX

!express && expressNext && fitExpress

EXPRESSFRAME

express = acceptNext();
encodeEncapsulatedFrame(express); express = Null;
outUserFrames++;

UCT

!express &&
expressNext && !fitExpress && fragmentExpress

INITIALEXPRESS

express = acceptNext();
encodeInitialFragment(express);
outUserFragments++;

UCT: TX

noExpress && pframe && fitPframe

FINALPFRAME

encodeFinalFragment(pframe); pframe = Null;
outUserFragments++; outUserFrames++;

UCT

noExpress && pframe && !fitPframe && fragmentPframe

INNERPFRAME

encodeInnerFragment(pframe);
outUserFragments++;

UCT: TX

noExpress && !pframe && pframeNext && fitPframe

PFRAME

pframe = acceptNext();
encodeEncapsulatedFrame(pframe);
outUserFrames++;

UCT

noExpress &&
!pframe && pframeNext && !fitPframe && fragmentPframe

INITIALPFRAME

pframe = acceptNext();
encodeInitialFragment(pframe);
outUserFragments++;

UCT: TX

else //none of the above

TX

encodeTrailingPad(); txMppdu(); outMppdus++; mppdu = NULL;

State machine conditions:

mppdu : True iff (if and only if) an MPPDU has been generated and not yet transmitted.

express : True (not Null) iff the PrY is holding the remainder or all of an Express user data frame.

expressNext : True iff the PrY's user has selected the next user data frame for transmission frame, that frame is available for transmission but has not yet been accepted by the PrY), and is an Express frame.

noExpress: True iff express and expressNext are both False.

pframe : True (not Null) iff the PrY is holding the remainder or all of a Preemptible user data frame.

pframeNext : True iff the PrY's user has selected the next user data frame for transmission frame, that frame is available for transmission but has not yet been accepted by the PrY), and it is a Preemptible frame.

fitExpress : True iff the Express frame (or the whole of the remainder of the fragmented Express frame) can be encoded in the remaining MPPDU octets.

fragmentExpress : True iff the Express frame or its remainder can be fragmented, and the next fragment encoded in the remaining MPPDU octets.

fitPframe : True iff the Preemptable frame (or the whole of the remainder of the fragmented Preemptable frame) can be encoded in the remaining MPPDU octets.

fragmentPframe : True iff the Preemptable frame remainder can be fragmented, and the next fragment encoded in the remaining MPPDU octets.

State machine procedures:

express = acceptNext() : Accept the next user data frame (an Express frame) for transmission, similarly frame = acceptNext() for a Preemptible frame.

encodeEncapsulatedFrame(express), encodeEncapsulatedFrame(pframe) : Encode the user data frame in the MPPDU, and add the number of user data octets encoded (not including the MPPCI) to outUserOctets.

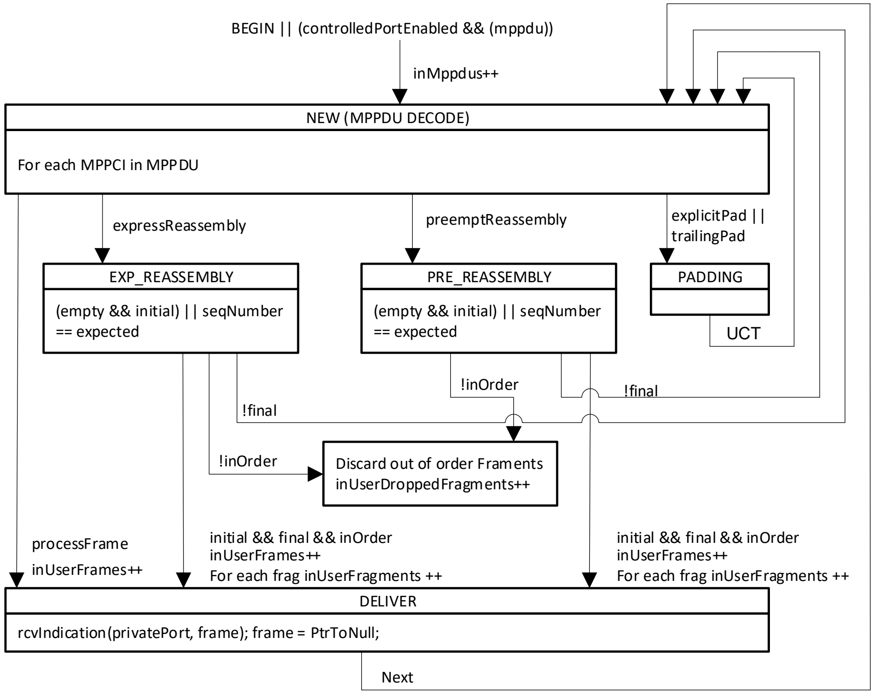
encodeInitialFragment(express), encodeInitialFragment(pframe) : Encode an Initial Fragment, encapsulating the greatest multiple of 64 octets from the user data frame that will fit in the MPPDU leaving at least 64 octets of the user data frame as a remainder, and add the number of user data frame octets encoded (not including the MPPCI) to outUserOctets.

encodeInnerFragment(express), encodeInnerFragment(pframe) : Encode a Frame Fragment (with Initial and Final bits clear), encapsulating the greatest multiple of 64 octets that will fit in the MPPDU leaving at least 64 octets of the frame as a remainder, and add the number of user data frame octets encoded (not including the MPPCI) to outUserOctets.

encodeFinalFragment(express),encodeFinalFragment(pframe) : Encode the remainder of the user data frame in a Final Fragment.

encodeTrailing Pad() : Encode the value 0 in all the remaining octets (if any) of the MPPDU, add the number of pad octets to outPadOctets.

txMppdu() : Transmit the MPPDU through the PrY's Controlled Port.



BEGIN || (controlledPortEnabled && (mppdu))

inMppdus++

NEW (MPPDU DECODE)

For each MPPCI in MPPDU

expressReassembly

preemptReassembly

explicitPad ||
trailingPad

EXP_REASSEMBLY

(empty && initial) || seqNumber
== expected

PRE_REASSEMBLY

(empty && initial) || seqNumber
== expected

PADDING

UCT

!inOrder

!final

!final

!inOrder

Discard out of order Framents
inUserDroppedFragments++

processFrame
inUserFrames++

initial && final && inOrder
inUserFrames++
For each frag inUserFragments ++

initial && final && inOrder
inUserFrames++
For each frag inUserFragments ++

DELIVER

rcvIndication(privatePort, frame); frame = PtrToNull;

Next

State machine conditions:

State machine conditions:

controlledPortEnabled : Enabling contion.
empty : True iff the assemby has no pending fragments.
expressFragment : True iff the fragment has a fragment header and an express indication
preeemptFragment : True iff the fragment has an express indication false and a fragment header
initial : True iff the fragment is an initial fragment
final:True iff the fragment is a final fragment
inOrder:True iff all the current fragments are in order
seqNumber : the sequence number of the current fragment.
expected: True iff the sequence number received is the next expected sequence number
frame: True iff the MPPCI indicates a frame
mppdu: True iff the frame is an MAC Privacy PDU
Statistic update points are illustrated with inXxx where appropriate counters are adjusted

Comments?
Thank You