IEEE P802.11
Wireless LANs

|  |
| --- |
| Resolution for CID 7087, 7088 (and 7541) for D5.0 |
| Date: 2016-02-02 |
| Author(s): |
| Name | Affiliation | Address | Phone | email |
| Graham SMITH | SRT Wireless | Davie, FL, USA. | 916 799 9563 | gsmith@srtrl.com |

Abstract

This submission proposes resolution for CID 7087, 7088

Also consider CID 7541

Green indicates material agreed to in the group,

yellow material to be discussed, red material rejected by the group and

cyan material not to be overlooked.

The “Final” view should be selected in Word.

|  |  |  |
| --- | --- | --- |
| Identifiers | Comment | Proposed change |
| CID 7087Graham Smith10.22.21351.36 | EDCA should be same or very similar to DCF but with AIFS[AC] in place of DIFS. The way it is presented by using the 'EDCA Slot Boundaries' is confusing and very unclear as to what bouindary applies at what decision point. | The commenter will bring a contribution to 'clean up' and clarify EDCF operation. It has too many mistakes to list here. |

|  |  |  |
| --- | --- | --- |
| Identifiers | Comment | Proposed change |
| CID 7088Graham Smith10.22.2.41351.39 | A BIG PROBLEM exists with the "aRxTxTurnaroundTime" inclusion in every slot boundary. If it is in every slot determination, then each STA may/will have a different wait time. For example assume the STA is at a randomly selected backoff slot and counting down, then every time the medium becomes busy then it should wait AIFS but if it waits "AIFS aRxTxTurnaroundTime" then a STA with a higher aRxTxTurnaroundTime waits less. In fact no STA will actually wait AIFS which is the real need. The aRxTxTurnaroundTime is only used at the very end when the STA can switch on its TX early. One does wonder why this obsession with this turnaround, it was not used in DCF, and there is a case as to why it is not required here. As long as the STA has waited for the prescribed time before it actually transmits, it does not matter if it goes earlier to make up for a turnaround time. It should not need telling. By the way, I recall that a practical value for this is 1-2us (RIFS for example). However, if considered necessary, it has to be clear that this can only be used only once in the countdown, it cannot be used every time the medium goes busy and the countdown is halted otherwise the wait time is shortened and a STA with a larger aRxTxTurnaroundTime actually has an advantage. We need to fix this | The commenter will bring a contribution to explain and correct this. |



Discussion:

EDCA is derived from DCF according to 10.22.2.1

10.22.2.1 P1348.36

*“****The EDCA channel access protocol is derived from the DCF procedures*** *described in 10.3 (DCF) by adding four independent enhanced distributed channel access functions (EDCAFs) to provide differentiated priorities to transmitted traffic, through the use of four different access categories (ACs).”*

So one would expect that the EDCA procedure would be similar to DCF but **with AIFS[AC] replacing DIFS**. Keep that in mind as we plough through this.

NOTE: To remind oneself on the IFS times, see P1270.

10.22.2.4 P1351.36 is where “EDCA slot boundaries” are defined, then one, and only one of 4 actions is allowed, without relating them to each other. What a strange (stupid) way of doing it? It does not, anywhere that I can find, lay down the basic rules clearly.

There are 6 boundaries. In fact they are variations on the equivalent of AIFS and EIFS.

In DCF we have simply DIFS and EIFS. Why we have 6 for EDCA is a stretch.

Note the following relationships:

AIFS[AC] = SIFS + AIFSN[AC] x aSlotTime )

EIFS = SIFS + AckTxTime + DIFS

**First 10.22.2.4 starts confusingly**

P1351.18

*“Each EDCAF shall maintain a backoff timer, which has a value measured in backoff slots as described below.*

*When the backoff procedure is invoked, the backoff timer is set to an integer value chosen randomly with a uniform distribution taking values in the range [0,CW[AC]] inclusive.”*

What is “*backoffslots as described below*” referring to? it seems to refer to the “slot boundaries”, and this is plain wrong. Now CW[AC] is in aSlotTimes, so this is OK. Let’s clean this up.

*“Each EDCAF shall maintain a backoff timer, which has a value measured in integers of aSlotTime.*

*When the backoff procedure is invoked, the backoff timer is set to an integer value chosen randomly with a uniform distribution taking values in the range [0,CW[AC]] inclusive.”*

Let’s take one boundary at a time:

1. *Following AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily idle medium during the SIFS) after the last busy medium on the antenna that was the result of a reception of a frame with a correct FCS.”*

Compare to ‘DIFS’ for DCF, this is indeed a time of AIFS[AC], but it adds that the medium may be **busy during SIFS and free for the bit afterwards** – that is different, I wonder if anyone does it? Last part makes it clear that this boundary is only after a good packet reception.

Why do we have aRxTxTurnaroundTime? It is not in the DCF? Also it is implementation dependent. The question is, does the STA ALWAYS transmit after this boundary - NO? We will look at this term later.

The next bullet is the same but for the EIFS case.

1. *Following EIFS – DIFS + AIFSN[AC] × aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium after the last indicated busy medium as determined by the physical CS mechanism that was the result of a frame reception that has resulted in FCS error, or PHY-RXEND.indication (RXERROR) primitive where the value of RXERROR is not NoError.*

*“EIFS – DIFS + AIFSN[AC] × aSlotTime + aSIFSTime”*

Now EIFS = **SIFS + AckTxTime + DIFS**,

Hence substitute for EIFS

= (SIFS + AckTxTime + DIFS) – DIFS + AIFSN[AC] × aSlotTime + aSIFSTime

= SIFS + AckTxTime + (AIFSN[AC] × aSlotTime + aSIFSTime)

= **SIFS + AckTxTime + AIFS**

Yes, replaced DIFS with AIFS. Is this expression better?

Next boundary:

1. *When any other EDCAF at this STA transmitted a frame requiring acknowledgment, the earlier of*
	1. *The end of the AckTimeout interval timed from the PHY-TXEND.confirm primitive, followed by AIFSN[AC]× aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium, and*
	2. *The end of the first AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily medium idle during the SIFS, the start of the SIFS implied by the length in the PHY header of the previous frame) when a PHY-RXEND.indication primitive occurs as specified in 10.3.2.9 ((#1198)Ack procedure).*

This is simply the STA transmitting using a different EDCAF, and waiting until that transmission is complete. This is the same as in DCF when waiting after having transmitted one packet and then transmitting another.

Anyhow what is it saying?

1. This is case of no ACK returned. Simply wait for the ACKTimeOut, then start AIFS, which is the same as a packet that was received correctly. Does “aRxTxTurnaroundTime” have a place here? Do we ALWAYS transmit after this boundary - NO?
2. This is case of an ACK returned. If so, then this is always the determination over 1).

Anyway, again this boils down to the same time AIFS[AC] as in a) and b).

Let’s look at the next one

1. *Following AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily medium idle during the SIFS) after the last busy medium on the antenna that was the result of a transmission of a frame for any EDCAF and which did not require an acknowledgment.*

Same as a) but after a packet not needing an ACK. Again same time AIFS[AC]

Next one:

1. *Following AIFSN[AC] × aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium after the last indicated idle medium as indicated by the CS mechanism that is not covered by a) to d).*

This I really struggle with; “idle medium after last indicated idle medium”. Is it ‘idle’ or ‘busy then idle’, what’s the difference? It is either idle or busy. What is an example of what this critieria is supposed to cover? Nope, can’t think of one, maybe someone could elucidate here? I think this should be “**after the last indicated BUSY medium”. Anyway, it is simply wait AIFS[AC], same as before**.

1. *Following aSlotTime of idle medium, which occurs immediately after any of these conditions, a) to f), is met for the EDCAF*

It is simply saying that the SlotTime of idle medium is a boundary. This applies to when in backoff. If the medium is idle for a slot time, then couint down a slot. This is the odd one out. The others are all AIFS, this is aSlotTime. The rules are clear that the backofftime is in aSlotTimes, so we do not need this “boundary”.

**So none of these are slot boundaries as such they are a time duration.**

The next instruction is P1352.1

*On these specific slot boundaries each EDCAF shall make a determination to perform one and only one of the following functions:*

* *Decrement the backoff timer*
* *Initiate the transmission of a frame exchange sequence*
* *Invoke the backoff procedure due to an internal collision.*
* *Do nothing*

It seems to read that the STA decides what to do, also what is “shall make a determination” mean? The following paragraphs do define each of these steps. We need to re-write this. Worse though is that everything here is actual incorrect. The first 5 slot boundaries defined are equivalent to DIFS (or EIFS) in DCF, 6th is SlotTime. They are NOT the slot boundaries as used in the backoff procedure.

Next is:

P1352.16

*“At each of the above-described specific slot boundaries, each EDCAF shall decrement the backoff timer if the backoff timer for that EDCAF has a nonzero value.”*

Now each “slot boundary” is AIFS[AC] this has a value in the order of 43us (16 + 27) EXCEPT FOR f) slot is 9us. This reads as though backoff timer decrements only occur at slot boundaries of 43us. **This is not right.** If free for 9us, then the slot is decremented. If becomes busy, timer is **suspended**, the wait AIFS then **resume**.

Then we have

P1352.20

*“At each of the above-described specific slot boundaries, each EDCAF shall initiate a transmission sequence if*

* *There is a frame available for transmission at that EDCAF, and*
* *The backoff timer for that EDCAF has a value of 0, and*
* *Initiation of a transmission sequence is not allowed to commence at this time for an EDCAF of higher UP.*

Again this has nothing to do with the slot boundaries of AIFS. If the backoff timer is 0, then transmit – it’s that simple.

This constant reference to the specific slot boundaries i.e. all of them, is confusing. 5 of these slot boundaries apply at the point when a packet is first presented and when the medium goes busy when in backoff. The big point is that when the backoff timer has reached 0, the STA can transmit.

Finally we have

P1352.28

*At each of the above-described specific slot boundaries, each EDCAF shall report an internal collision (which is handled in 10.22.2.4 (Obtaining an EDCA TXOP)) if*

* *There is a frame available for transmission at that EDCAF, and*
* *The backoff timer for that EDCAF has a value of 0, and*
* *Initiation of a transmission sequence is allowed to commence at this time for an EDCAF of higher UP*.

Why wait for the boundary, the ONLY criteria is that the backoff timer is 0.

So, having determined that the EDCA slot boundaries and descriptions are a mess, and incorrect, I wonder how anyone ever implemented this. I can only assume that they did the sensible thing, ignored all this rubbish and implemented DCF with AIFS in place of DIFS.

I will try to re-write with as few changes as possible.

**Before I do that let’s check on “backoff timer”.** First mentioned

P1288.20

*10.3.3 Random backoff time*

*After this DIFS or EIFS medium idle time, the STA shall then* ***generate a random backoff period*** *(defined by Equation 10-1) for an additional deferral time before transmitting, unless the* ***backoff timer already contains a nonzero value,*** *in which case the selection of a random number is not needed and not performed. This process minimizes collisions during contention between multiple STAs that have been deferring to the same event.*

*Backoff Time = Random() x aSlotTime (10-1)*

*P1290.54*

*10.3.4.3 Back Off Procedure for DCF*

*To begin the backoff procedure, the STA shall set its Backoff Timer to a random backoff time using the equation in 10.3.3 (Random backoff time).*

…..

*If no medium activity is indicated for the duration of a particular backoff slot, then the backoff procedure shall decrement its backoff time by aSlotTime.*

***If the medium is determined to be busy at any time during a backoff slot, then the backoff procedure is suspended****; that is, the backoff timer shall not decrement for that slot. The medium shall be determined to be idle for the duration of a DIFS or EIFS, as appropriate (see 10.3.2.3 (IFS)), before the backoff procedure* ***is allowed to resume****. Transmission shall commence when the Backoff Timer reaches 0.*

**So it is very clear, the backoff timer counts down in periods of aSlotTime. If medium goes busy during a slot, then is suspends, waits for medium to be free, waits DIFS, then continues count down. For example, decrements timer by 9us, after 5us medium goes busy, suspends, waits DIFS, resumes, then 4us later decrements timer by 9us**.

NOTE: It waits DIFS, not (DIFS – aRXTXTurnaroundTime) but Fig 10-19 is the only place it is mentioned and text at P1296.58.

*The STA may employ any non-negative value for each of the parameters:*

*— aRxPHYDelay*

*— aMACProcessingDelay*

*— aRxTxTurnaroundTime*

*— aTxPHYDelay*

aRxTxTurnover Time

It is clear that the “Boundaries” are in fact the EDCA equivalent of DIFS used in DCF backoff. For simplicity we call this “AIFS”. A BIG PROBLEM exists with the “aRxTxTurnaroundTime” inclusion in every AIFS.

If it is in every AIFS determination, then each STA may/will have a different wait time. For example assume the STA is at a randomly selected backoff slot and counting down, then every time the medium becomes busy then it should wait AIFS but if it waits “AIFS – aRxTxTurnaroundTime” then a STA with a higher aRxTxTurnaroundTime waits less. In fact no STA will actually wait AIFS which is the real need.

**The aRxTxTurnaroundTime is only used at the very end, or very beginning, to allow the STA time to switch on its TX**. One does wonder why this obsession with this turnaround, it was not used in DCF, and there is a case as to why it is not required here. As long as the STA has waited for the prescribed time before it actually transmits, it does not matter if it goes earlier to make up for a turnaround time. It should not need telling. By the way, I recall that a practical value for this is 1-2us (RIFS for example).

However, if considered necessary, it has to be clear that this **can only be used only once in the countdown**, it **cannot be used every time the medium goes busy** and the countdown is halted otherwise the wait time is shortened and a STA with a larger aRxTxTurnaroundTime actually has an advantage.

It could be entirely deleted without affecting the procedures. It would be interesting to know when this term crept in.

**Let’s make the rule that this term can only be included once. A convenient place is to invoke it only at the beginning. I will see if I can add text for that. What is clear is that it cannot be invoked every time.**



Look at Figure 10-26, in this figure the aSIFSTime for some strange reason has RX/TX in it. Why? SIFS is SIFS.

The example is for AIFSN=2, CW = 0. So the STA is simply supposed to wait SIFS + (2 x aSlotTime) –Rx/Tx so making time go backwards does not really help. The action is to calculate the AIFS minus Rx/Tx then start counting down. As written this figure seems to indicate that the Rx/Tx time is part of SIFS and SlotTime.

Assume for simplicity sake, we use “boundary” a), SIFS + AISFN x Timeslot – Rx/Tx

BUT what if there is a stop, say half way during the first slot? What is the **total backoff time**?

**If we use the text** then we have:

It is SIFS + aSlotTime/2 + [SIFS + (2 x aSlotTime) – Rx/Tx] + aSlotTime/2 + aSlotTime –Rx/Tx

If not convinced that the last Rx/Tx is there or not consider if it stops twice? Halfway through the first then half way through the second.

SIFS + aSlotTime/2 + [SIFS + (2 x aSlotTime) **– Rx/Tx**] + aSlotTime + [SIFS + (2 x aSlotTime) – **Rx/Tx**] + aSlotTime/2 **– Rx/Tx**

Clearly wrong!!!

Somehow we must have words that it is **only deducted once**.

BUT if we look at the diagram it is effectively saying, don’t wait SIFS but wait SIFS – Rx/Tx then start counting slots.

The diagram is simply wrong as it shows the case where the medium is busy hence there is a random back off required. The diagram is based on an AIFSN of 1 or 2 and presumeably a random CW = 0.

Also, as pointed out in CID 7085, the implemetation specific, pick any value list of tiny times do not add up to SIFS or TimeSlot. So the diagram needs work.

Here is what Figure 10-26 should look like, nice and clean.



Notes:

* D1, D2, M1, M2 etc. times do not add up aSlotTime, because they don’t. In fact I don’t really know why they are shown at all, it detracts from the basic timing information that is the basis of the EDCA backoff. I would delete them but …I won’t.
* Rx/Tx only appears at the decision boundary, in this case for AIFSN = 2. It should not be shown for SIFS. It is used at the point where the calculation for the number of backoff slots is carried out.
* It needs to be stated that this example is for random [0,CW] as the medium is busy AND for the case of AIFSN = 2

*I tried to make as few changes as possible rather than try to re-write the entire section.*

***I also re-use the text from the DCF backoff procedure.***

**RESOLUTION for CIDs 7087, 7088, 7541**

REVISED

At P1351.18 (10.22.2.4)

Edit as follows:

“Each EDCAF shall maintain a backoff timer, which has a value determined using equation 10-1.

Edit at P1351.36:

The medium shall be determined to be idle for a duration of a time period defined as follows on the primary channel, for each EDCAF, before initiating the transmission of a frame exchange sequence, or, if the backoff timer is suspended, before resuming to decrement the backoff timer:

1. Following AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily idle medium during the SIFS) after the last busy medium on the antenna that was the result of a reception of a frame with a correct FCS.
2. Following EIFS – DIFS + AIFSN[AC] × aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium after the last indicated busy medium as determined by the physical CS mechanism that was the result of a frame reception that has resulted in FCS error, or PHY-RXEND.indication (RXERROR) primitive where the value of RXERROR is not NoError.
3. When any other EDCAF at this STA transmitted a frame requiring acknowledgment, the earlier of
	1. The end of the AckTimeout interval timed from the PHY-TXEND.confirm primitive, followed by AIFSN[AC] × aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium, and
	2. The end of the first AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily medium idle during the SIFS, the start of the SIFS implied by the length in the PHY header of the previous frame) when a PHY-RXEND.indication primitive occurs as specified in 10.3.2.9 (Ack procedure).
4. Following AIFSN[AC] × aSlotTime – aRxTxTurnaroundTime of idle medium after SIFS (not necessarily medium idle during the SIFS) after the last busy medium on the antenna that was the result of a transmission of a frame for any EDCAF and which did not require an acknowledgment.
5. Following AIFSN[AC] × aSlotTime + aSIFSTime – aRxTxTurnaroundTime of idle medium after the last indicated busy medium as indicated by the CS mechanism that is not covered by a) to d).

Immediately after any of these conditions, a) to e), is met for the EDCAF, each EDCAF shall decrement the backoff timer if the backoff timer for that EDCAF has a nonzero value. If the backoff timer is suspended, the medium shall be determined to be idle for a duration of a time period a) to e), ignoring the term aRXTXTurnaroundTime, before resuming to decrement the backoff timer.

Each EDCAF shall initiate a transmission sequence if

* There is a frame available for transmission at that EDCAF, and
* The backoff timer for that EDCAF has a value of 0, and
* Initiation of a transmission sequence is not allowed to commence at this time for an EDCAF of higher UP.

NOTE—In the case that an EDCAF gains access to the channel and transmits MSDUs, A-MSDUs, or MMPDUs from a secondary AC, the EDCAF of the secondary AC is not affected by this operation. If the EDCAF of a secondary AC experiences an internal collision with the EDCAF that gained access to the channel, it performs the backoff procedure regardless of the transmission of any of its MSDUs, A-MSDUs, or MMPDUs (See 10.22.2.6 (Sharing an EDCA TXOP)).

Each EDCAF shall report an internal collision (which is handled in 10.22.2.4 (Obtaining an EDCA TXOP)) if

* There is a frame available for transmission at that EDCAF, and
* The backoff timer for that EDCAF has a value of 0, and
* Initiation of a transmission sequence is allowed to commence at this time for an EDCAF of higher UP.

An example showing the relationship between AIFS, AIFSN, DIFS, and slot times immediately following a medium busy condition (and assuming that medium busy condition was not caused by a frame in error) is shown in Figure 10-26 (EDCA mechanism timing relationships). In this case, with AIFSN = 2, the EDCAF may decrement the backoff counter for the first time at aSIFSTime + 2 × aSlotTime – aRxTxTurnaroundTime following the end of the medium busy condition. If, in this example, the backoff counter contained a value of 1 at the time the medium became idle, transmission would start as a result of an EDCA TXOP on-air at a time

aSIFSTime + 3 × aSlotTime - aRxTxTurnaroundTime

following the end of the medium busy condition.

At P1353.1 Replace Figure 10-26 with

