Bachelor Thesis

MAC Layer Implementation of the IEEE 802.15.3c Standard

Alberto Arco Miras

September 2010

Prof. Dr.-Ing. Tomas Kürner

Tutor: Dipl.-Ing. Marcos Liso

Department of Mobile Radio Systems

Institute for Communications Technology

Braunschweig Technical University

Acknowledgment

First of all, I wish to thank Prof. Thomas Kürner for giving me the opportunity to conduct this work. My special thanks go to Marcos Liso, who provided the topic for this bachelor thesis. His ongoing assistance and many fruitful discussions concerning various aspects of this work had a great influence on the style of this document and certain design decisions of the practical part of this work. So, the successful completion of this bachelor thesis is to great extend due to her.

Moreover, I want to thank my partner, Pablo Olivas, for the help he offered me on my new entrance on Simulink.

Additionally, I thank Amanda Eyer, that has helped me to correct the English language of the document..

Last but not least, I thank my friends, my family and Maria for their support, without which I would not have been able to complete this work.

Abstract

The transfer of data without wires is one of the main handicaps on this time. Not only for commercial reasons, also for research and manufacturing.

The wireless technology on multimedia streaming has to evolve as fast as the multimedia technology does. Multimedia technology is evolving in order to reach better quality for the user's screen, therefore it needs more capabilities requirements; as better screens, better encoders, better devices to reproduce them. If we add the multimedia and the wireless technology, the result is that we need a wireless channel which could support the bandwidth that the video encoder needs.

The uncompressed video or High Definition multimedia has entered so fast on the commercial issues. The main problem with that is the large amount of disk space that it needs, therefore the high rate transmission that require for streaming. To stand up to this with the wireless technology, the IEEE has developed the standard 802.15.3c in order to reach the data rate needed to transport the multimedia data without need wires.

This standard is based on personal area networks (WPAN), which are computer networks used for communications between computer devices; including telephones, smartphones and tablets. Typically, a wireless personal area network uses some technology that permits communication within about 10 meters - in other words, a very short range. One such technology is Bluetooth. The WPANs could serve to interconnect all the ordinary computing and communicating devices that many people have on their desk or carry with them today. It is in its infancy and is undergoing rapid development. Proposed operating frequencies are around 2.4 GHz in digital modes.

This standard uses the extremely high frequency of 60 GHz, it is considered between the highest radio frequency bands, from 30 to 300 GHz above which electromagnetic radiation is considered to be low (or far) infrared light. On this frequency band the rates could be greater than 5 Gbps, against the 2.5 Gbps that the uncompressed 1080p video, it is a good solution.

The 60 GHz band usually requires line of sight between transmitter and receiver, and the 802.15.3c standard specification ameliorates this limitation through the use of "beam forming" at the receiver and transmitter antennas to increase the signal's effective radiated power.

The aim of this bachelor thesis is to build a simulator of the 802.15.3c MAC layer protocol, in order to study the ways of the standard to construct a piconet, the structure of the channel time and the different modes to transmit data that is implemented on it.

First of all, there is a summary of the main process of the standard to reach communication between two devices and an explanation of the MAC frames used for it. After that, I am going to talk about how is working the simulator and the results of it. Finally, the document is finished with conclusions and future works on the simulator.

Contents

1 802.15.3c network

The 802.15.3c protocol was built with the motivation of the increasing demand of wireless communications with the ubiquitous network connectivity, low-cost and low-power consumption, high data rate, quality of service (QoS) support, security. The wireless personal area networks (WPAN) can achieve all these demands. However there was not a protocol with all this demands that reaches the data rates proposed for High Definition data.

This protocol uses a type of network called piconet, this type of network is also used on other protocols like Bluetooth or ZigBee. The devices are connected through an ad hoc model and uses wireless technology of no more than 10 meters, it is called piconet.

A piconet is formed when at least two devices, such as a portable PC and a cellular phone, connect. When a piconet is formed, one device acts as the PNC (piconet Coordinator) while the others act as normal devices (DEV) for the duration of the piconet connection.

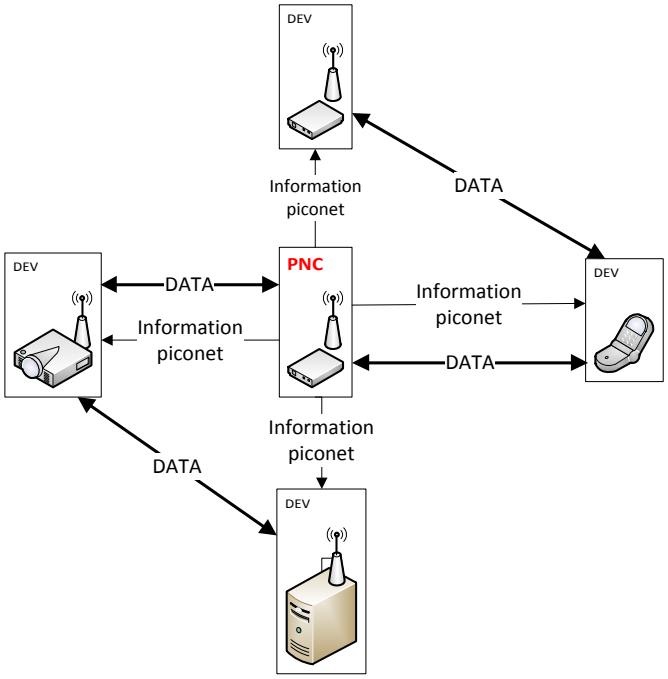


Figure 1 - piconet structure

In Figure 1, it is shown a basic established piconet. There are in total 5 devices. The device that is on the middle is acting as the PNC and the other as the DEVs. The PNC should send information of the piconet (Beacons) periodically, and on the same moment, it can be sharing information with one or more DEV and also the DEVs between them.

The main characteristics of a piconet:

- High Rate WPAN:
    - Short range (at least 10m, up to 70m possible).
    - High data rates.
- Dynamic Topology:
    - Mobile devices often join and leave piconet.
    - Short time to connect (<1s).
    - Support for multimedia quality of service(QoS).
- Ad-hoc network with Multimedia QoS provisions:
    - Time Division Multiple Access (TDMA) for streams with time based allocations.
    - Peer to peer connectivity.
- Multiple Power Management Modes:
    - Designed to support low power portable devices.
- Low price point, low complexity and small size.
- Secure Network
- Ease-of-use:
    - Dynamic coordinator selection and handover.
    - Does not rely on a backbone network

In the next parts, it is explained all the basic actions in order to reach to transmit data. However it is also explained some concepts that are considered necessary to understand the factors.

Firstly there is an explanation about how a piconet is coordinated on the perspective of a PNC, how can it start and end the piconet inclusively the way to pass away the managing, also showing the different type of topologies.

After that, it shows how the time is structure and which are the application and the access method for each period of time

Finally, it is explained how a DEV can join the piconet, so, when is associated, the way that it has to ask for an available channel time to transmit and the basic data transmit modes that the standard can implement as its acknowledgment modes.

1.1 Piconet coordination

As it's explained before, the PNC is the first device of the piconet or, when there is already a piconet created, the device with more capabilities of the piconet.
Here are the ways to begin and end a piconet, how can be changed the PNC and the different types of piconet that could be created.
There are some figures of the process during the time with different geometric figures,

1.1.1 Beginning a piconet

When any DEV turns on, it tries to find an available piconet. If it doesn't receive any beacons (information frames that are sent by the PNC periodically, explained in *1.2.1*), it scans for an available channel, after a specified caution time period it becomes the PNC and starts to send beacons

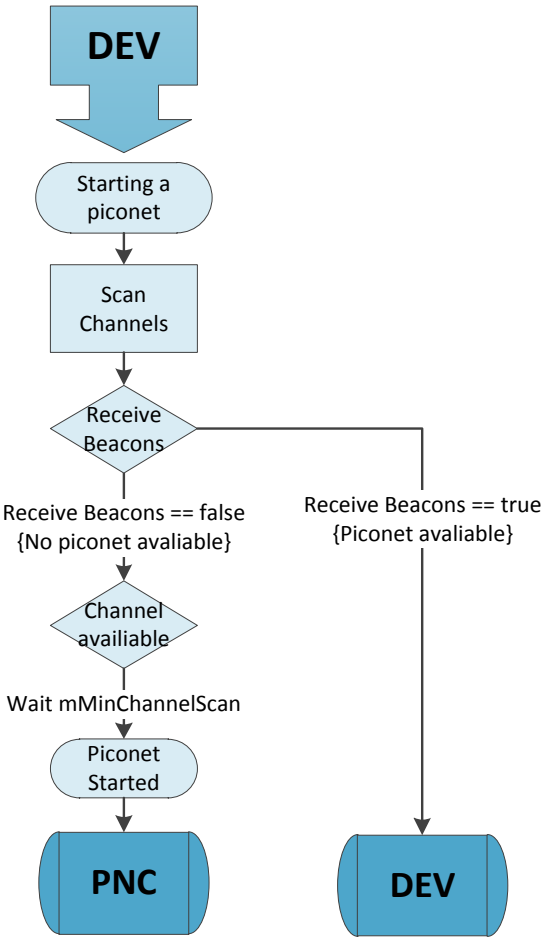


Figure 2 - Starting piconet

In *Figure 2* there is the schema of a device when it turns on. At first it scans the channels in order to look for beacons, if it receives a beacon it means that there is piconet already

available and it can try to ask for an association, then it becomes a DEV. If no beacons are received it means that there is no piconet on the coverage of the device, then it can start to find a channel available that is not used for other protocol, after waiting a *mMinChannelScan* (it is a period of time defined by the device to be sure that nobody is using the channel), it can begin to start a piconet, the device becomes the PNC of the new piconet and can start to send beacons to notify the new DEVs that are coming.
It is possible that a piconet exists but it doesn't accept more DEV; in that case the new DEV can start its own piconet (becoming the PNC of its piconet). The new piconet can work as a "Child piconet" or a "Neighbour piconet", depending on the necessities of the devices that are coming.

1.1.2 Child piconet

When a DEV wants to start a new piconet under the existence of another, it founds a Child piconet, established under the main piconet. There can be many Child piconet under the main piconet; all of them can support more dependent piconet.
This type of piconet is useful to extend the coverage of the network, or to share computational or memory requirements with other capable DEVs. The Child PNC is also a member of the parent piconet; it can act like a normal DEV of the parent piconet: exchange data, request information, services, etc…

1.1.3 Neighbour piconet

If a DEV wants to start a new piconet under the existence of another, but there are no vacant PHY channels, it can start a Neighbour piconet. In that case, it shares the frequency spectrum with the piconets that are already established.
Any piconet can have one or more Child or Neighbour piconet on the same time.
Unlike the Child PNC, the Neighbour PNC is not a member of the main piconet, and will not exchange data, request information, services, etc…

1.1.4 Change PNC

The PNC can change in different situations, if:

- The new DEV is more capable
- The PNC leaves

If either of these cases happens, the PNC should begin a process called Handover and will give control to another DEV in the piconet.

On the figure below, there is a schema of the handover process:

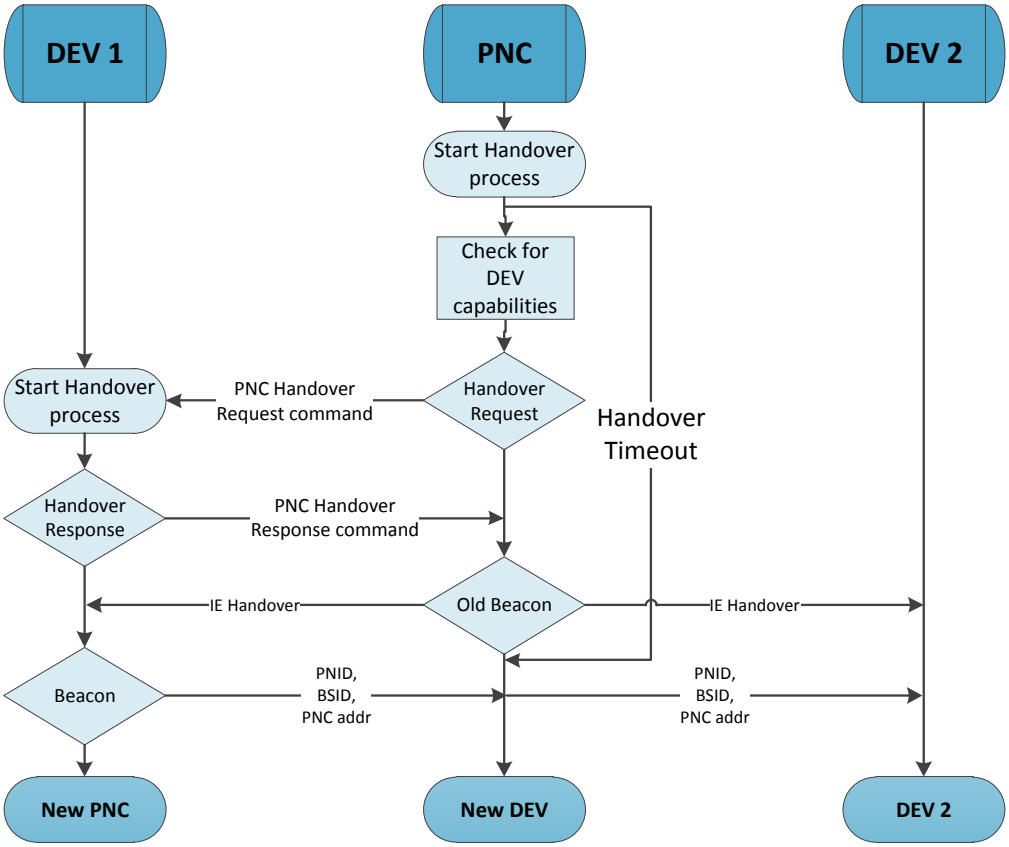


Figure 3 – Handover Process

Could be the possibility that a new DEV that is associated has better capabilities than the actual PNC, when a new DEV has entered the PNC should check its capabilities and compare them with his own. If it is more capable, the PNC passes on the control to the new user by using the PNC Handover Request command. The DEV should answer it with a PNC Handover Response command. Finally the old PNC has to notify all the DEVs of the piconet that the handover process was done, and the new PNC has to start to send its new beacons.

This process is also used when a PNC wants to leave the piconet, it should find the DEV with more capabilities and handover the control to it, if no DEV is available, it has just to finish the piconet.

1.1.5 Ending a piconet

When the PNC wants to leave the network, without there being another DEV capable to be the PNC, it should notify the other DEVs with a Beacon frame that the piconet is going to shut down. If the PNC could not give notice of his leaving, then the piconet is over. The other DEVs have to start a new piconet. When the PNC is a member of other parent piconet, then it shall disconnect as a normal DEV, with the Disassociation Process.
In the case that the parent piconet ends the operation, the parent PNC would select one of its dependent PNCs to receive its control. The other dependent piconet would shut down in order to start the association with this new parent piconet.

1.2 Channel time structure

The time should be divided in order to access without collisions and priorities.
In this protocol, the time is divided in consecutive periods called Superframes. Each Superframe consists of different time periods with different access methods, because each field is used by different users and different kind of information. The PNC, as the master DEV, must coordinate each Superframe, giving access to other DEVs and allowing them to use the network.

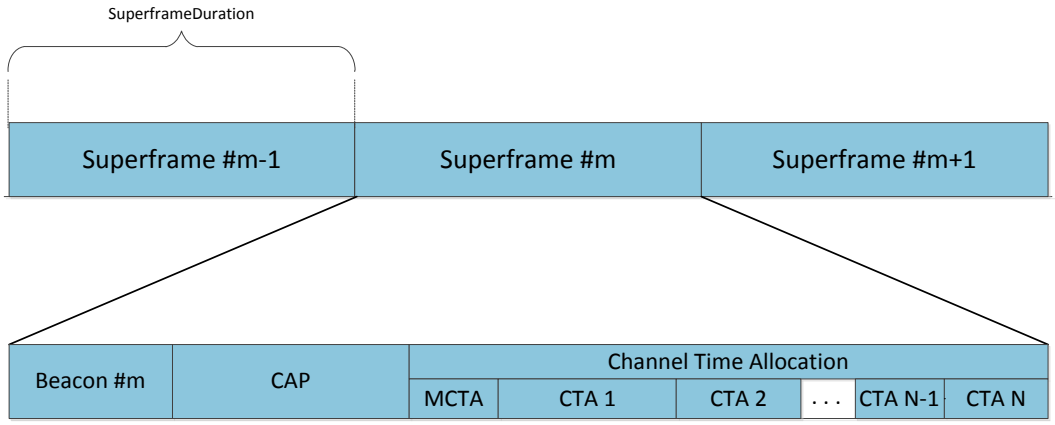


Figure 4 - Superframe

The PNC coordinates the channel access by sending a Beacon that marks the start of a Superframe. The Beacon is used in order to announce piconet-related information such as Superframe length, CAP and CTAP duration, network identity and timing synchronization stamp. The minimal length of a Superframe is 512 µs and the maximal length is 65535 µs.
The Superframe is composed of two periods, namely the Channel Access Period (CAP) and the Channel Time Allocation Period (CTAP).
Devices in the piconet synchronize with the PNC after receiving the Beacon and then transmit in either the CAP or CTAP.

1.2.1 Beacon

This frame is sent by the PNC to set the timing allocations and to broadcast management information for the piconet, including information for QoS management, power management and the basic timing. It is used to control the synchronization of the channel. It is transmitted one time in each Superframe and contains necessary information for the piconet operations.

1.2.2 Contention Access Period (CAP)

Similar to IEEE 802.11 (Wi-Fi), the 802.15.3c MAC protocol also employs CSMA/CA in CAP with the exception that the transmission may not exceed the duration of the CAP.
A Backoff algorithm, random time used to prevent collisions on the same access of different DEVs, must be applied before sending any frame other than the ImmACK (Immediate Acknowledgment), and the channel has to be idled for a Backoff Interframe Space (BIFS) before any backoff may be carried out.

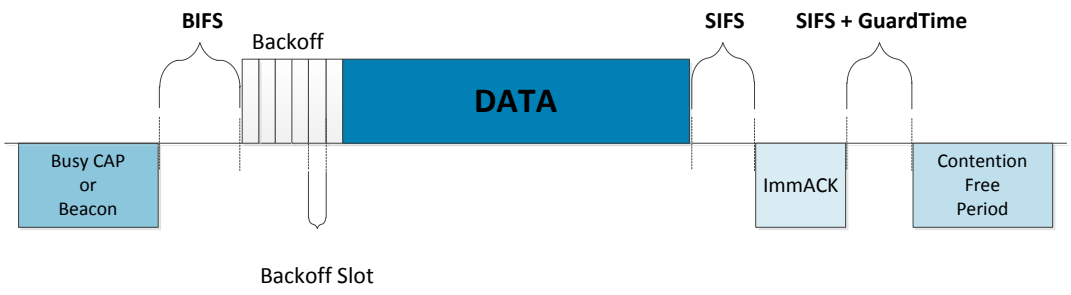


Figure 5 – Contention Access Period

On *Figure 5*, there is an example of a normal data transmission. After sending the beacon is the Channel Access Period, when a DEV wants to use it, it should wait a BIFS and some

Backoff slots (the Backoff is divided in slots in order to increment or decrement it), after this if no DEV has access before it can transmit data, the receiver should what a Small Interframe Space (smaller than BIFS) and then it can transmit the acknowledgment. Finally, the rest of the time of this period can be used by other DEV after a SIFS and GuardTime (time specified by the PNC to separate the different transmissions).

1.2.3 Channel Time Allocation Period (CTAP)

The CTAP is composed of Management Channel Time Allocations (MCTAs) and Channel Time Allocations (CTA).

A CTA stands for a guaranteed start time and reserved transmissions duration, when no other DEVs will compete for the cannel time resource. CTAs may be allocated by the PNC for both isochronous flows and asynchronous data packets.

Requests for channel time are generated by DEVs that have data transmit during the CTAP, and the requests are sent during the CAP. The duration of the channel time requested must cover the data transmission time itself, the (SIFS) and the ACK frame so that the DEV or PNC is able to transmit an ACK immediately, after the packet was received successfully.

Upon receiving a request, the PNC evaluates the current channel usage and all pending requests for CTAs. The PNC allocates CTAs to DEVs by announcing the CTA allocation in the Beacon of the next Superframe.

The assigned CTAs are used by the owned DEV like a TDM channel, since no competition by other DEVs is possible. However, a DEV must not extend its transmission beyond the end of frame duration required to finish a complete cycle DATA-SIFS-ACK. Between adjacent CTAs, a guard time plus and SIFs duration are always required in order to avoid overlap of use of adjacent CTAs.

The MCTA is used by the PNC, in order to use it for managing.

1.3 Join the piconet

The way to enter a piconet is by requesting an association. A DEV that wants to join an established piconet should first of all be on the coverage of the PNC and receive its Beacons. Then, the DEV should request an empty association place, using the Association Process.

Once it is associated, it can start using the services of the piconet and interact with all the DEVs of the established network.

1.3.1 Association Process

When a new DEV turns on and receives the Beacons of one PNC, it means that it is covered by a piconet. The Association Process provides the DEV with an associated ID and the information about all the DEVs that are on the network, which are necessary for the DEV to use the services of the piconet. The Association Process is an exchange of messages filled with information about the PNC and the DEV in question. This information concerns the addresses, capabilities, IDs, etc.

The messages are shown in the figure bellow:

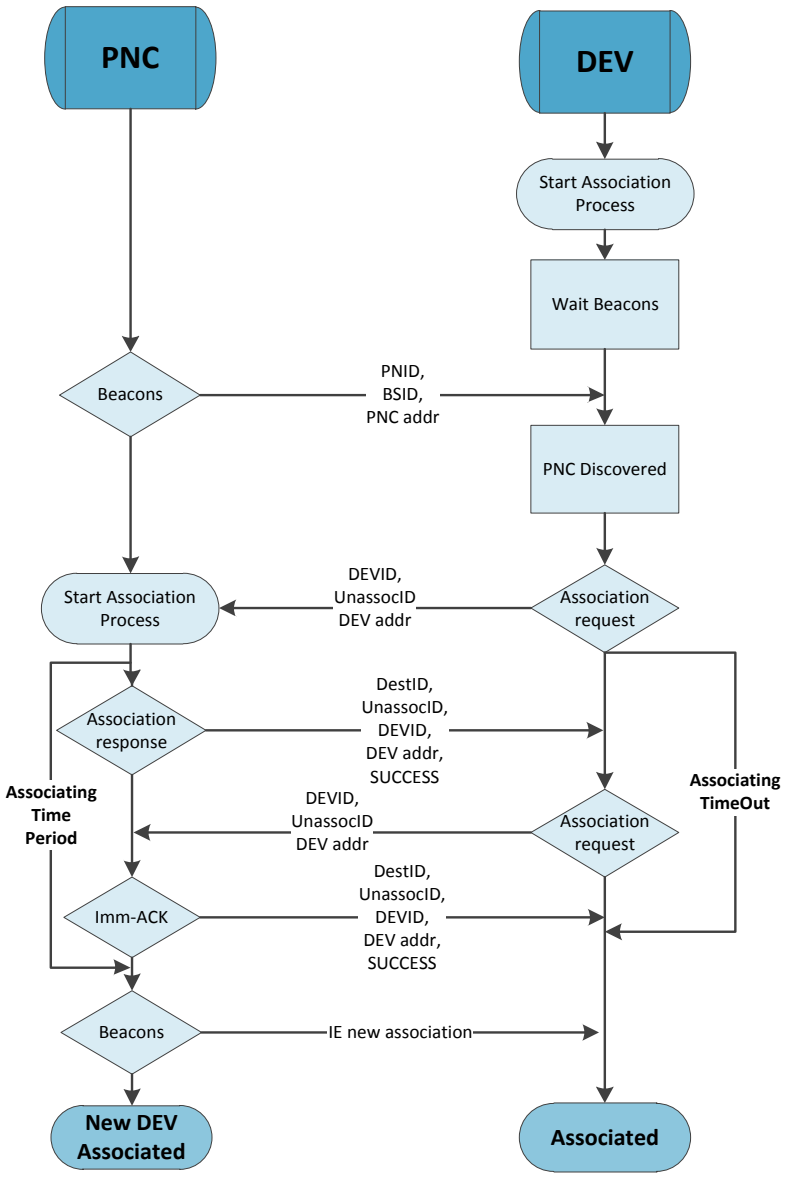


Figure 6 – Association process

The Command frames, Association Request and Association Response, and the ACK, ImmACK, are explained in *2.4*. The Associating Timeout is a period of time specified by the user. If this period expires, the PNC will discard the Association Process, and the DEV should start again. The Association Time Period is used by the PNC to know if a DEV is disconnected, and is used by a DEV if the PNC is disconnected. If the ATP of PNC expires, the PNC should dissociate the DEV, using the Disassociation Process. If the ATP of a DEV expires, it doesn't receive any Beacon), the DEV should consider itself disconnected from the piconet and may try to associate itself again.

1.3.2 Disassociation Process

When a DEV wants to leave the piconet, it should use the Disassociation Process. Afterwards the DEVID and all the other options specific to this piconet have no utility.
The PNC should inform the other DEVs with a Beacon that the specific ID has left.

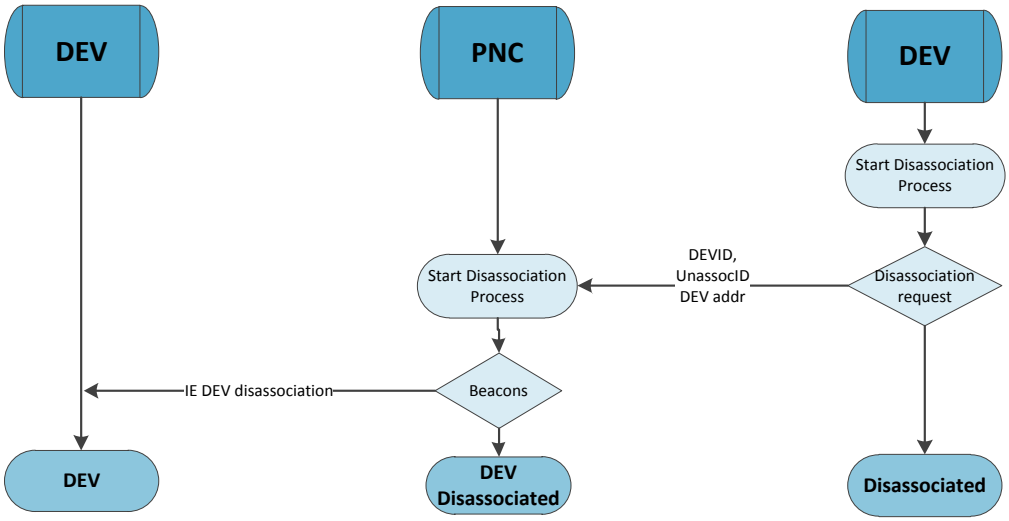


Figure 7 – Disassociation Process

The Disassociation Process is also used by the PNC when it wants to remove a DEV from the PNC or when the ATP expires. The PNC should also inform the other DEVs about the disassociation.

1.4 Assign channel time

Like it is explained in 1.2, a DEV wanting to transmit data, it should request for a CTA in order to own a period of the channel time. This period can be variable, depending on the data to transmit and the capabilities of the Channel Time of the PNC. If a DEV doesn't have any more data to transmit, it leaves the CTA, as to achieve the same requesting process without any Time Units (TU are slots of time of Channel Time Allocation that are defined by the PNC device when it starts the piconet). Then the PNC should remove the CTAs assigned and they are free for another petition.

Due to request CTAs, a DEV should start the CTA request process.

1.4.1 Request CTA process

The CTA request process is a series of messages that are exchanged between the DEV and the PNC. The DEV should include the number of TU (Time Units) needed. The PNC should check for the capabilities of its piconet. If the petition of the DEV can be satisfied, the PNC answers with whichever CTAs that have been assigned.

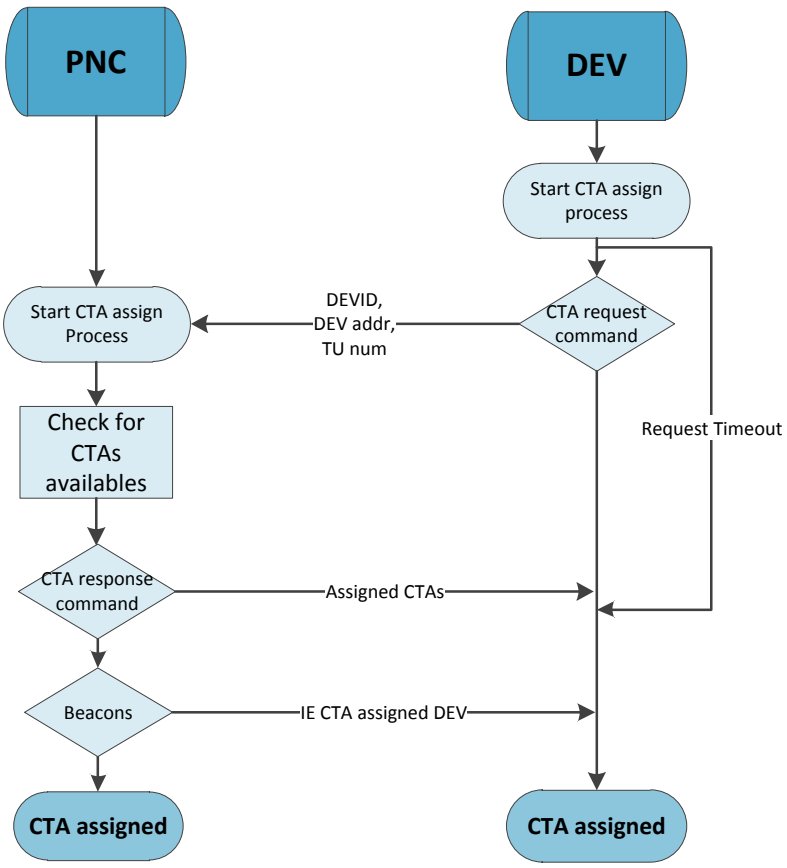


Figure 8 – Request CTAs

1.4.2 Leave CTA process

There are three possibilities for leaving CTAs:

- A DEV has no more data to transmit
- A PNC can no longer offer the CTA assigned for a DEV
- A DEV1 needs the CTA that is using another DEV2

The first case is illustrated in the figure bellow:

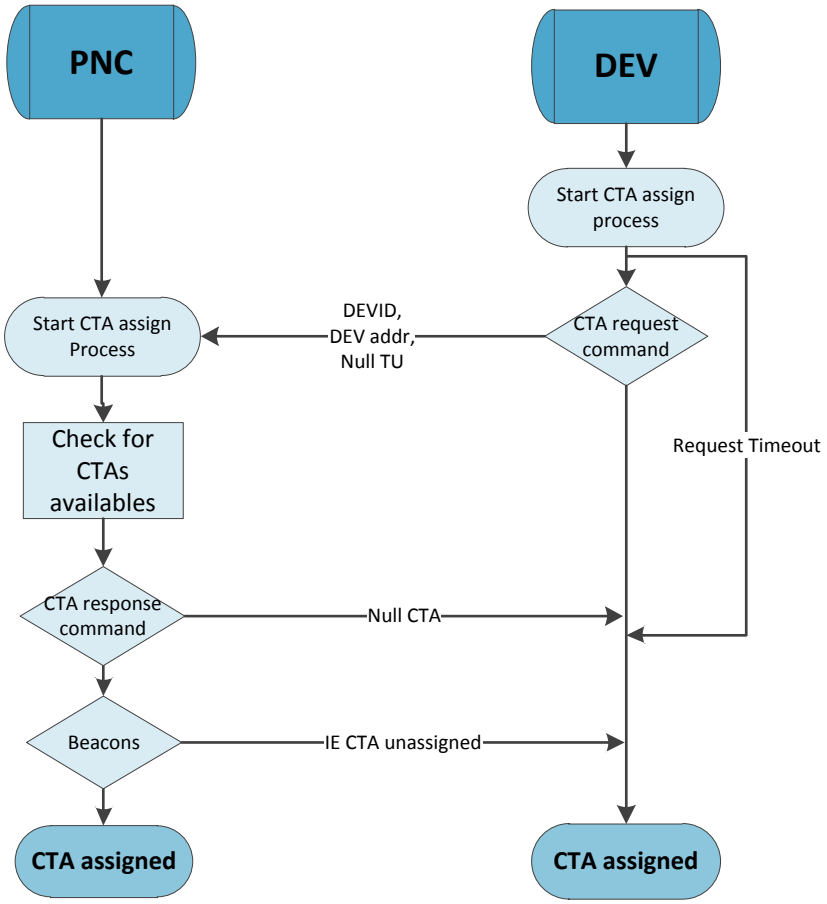


Figure 9 Removing CTAs

If there are enough capabilities available, the PNC should provide the ordered CTAs.

In the case that the PNC can no longer offer all of the CTAs that are assigned, it removes them with a CTA Response Command and informs the DEVs that it is going to finish the stream.

1.5 Acknowledgment data modes

There are different ways to transmit the data, therefore different ways to acknowledge them. If we join this standard with the last one, there are five different acknowledgement modes available. There are acknowledgement modes for basic transmit of data, another which allows sending a few frames without acknowledging, other one that don't need acknowledgment.

1.5.1 No-ACK mode

A transmitted frame with the ACK Policy field set to indicate no-ACK shall not be acknowledged by the intended recipient. The transmitting DEV assumes that the frame is successful for all its local management entities and proceeds to the next frame scheduled for transmission. The errors are not contemplated in this kind of transmissions, because these frames have not feedback and are sent periodically.

1.5.2 Immediate ACK mode

This is the most simple acknowledgement mode, it is just used to answer if a frame was received correctly, indicating, if is needed, the fragment number and the MSDU number. It is not only used for data, it can be used also for other processes that needs acknowledgment
Here is a figure that shows how it works for a data transmission:

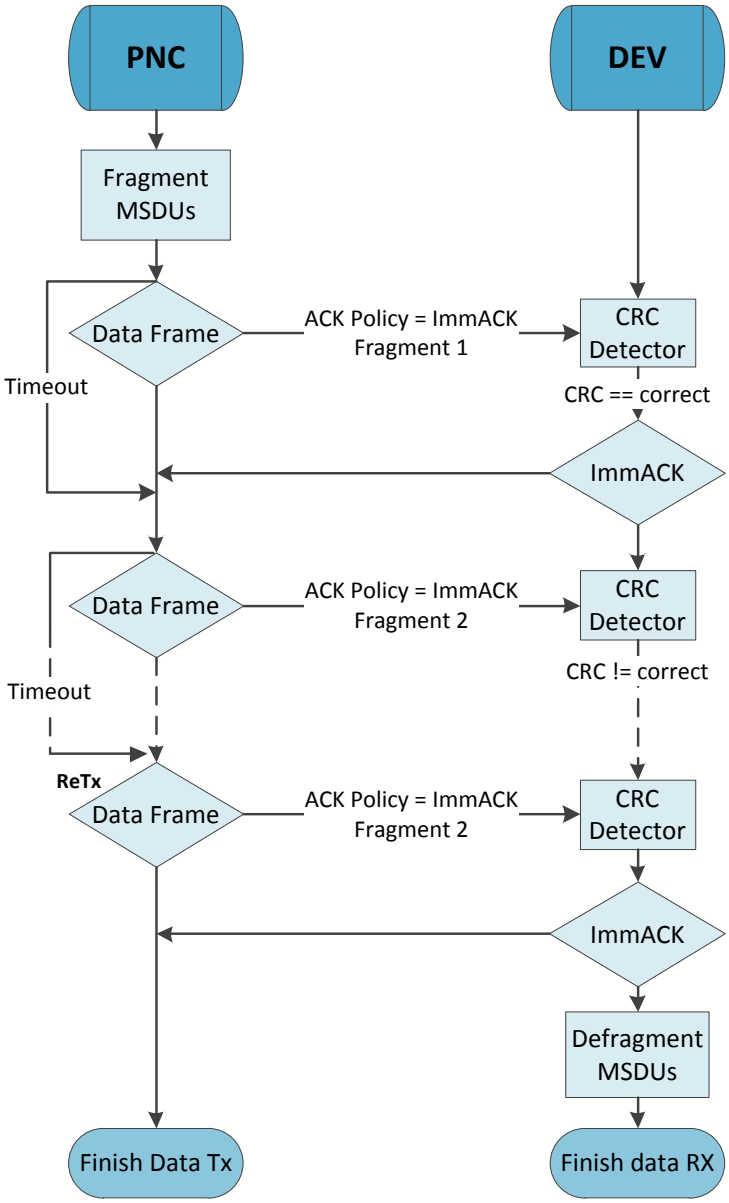


Figure 10 – Immediate ACK acknowledgment mode

At first the sender should fragment the MSDUs. This mode acknowledge each fragment received but it is not used for negative acknowledgment then when a Data frame is transmitted, it switch on a clock that is used to know if the frame was lost or was transmitted with errors, if this time expires the same fragment is retransmitted.

1.5.3 Delayed ACK mode

The Delayed-ACK protocol is composed of two phases: an initialization phase and a data exchange phase. Prior to data transmissions, the communicating pair negotiates the type of acknowledgement policy to be used. The initialization procedure involves the source node

sending a single data frame with the ACK policy field set to Dly-ACK Request. If the destination accepts the use of Dly-ACK, it responds with a Dly-ACK. As defined in the standard, this acknowledgement frame is used to acknowledge the received data frame and indicate the burst size. By burst size it is meant the maximum number of frames the sender may send before receiving the ACK. The value of the burst size may be determined according to the receiver buffer requirement and the delay tolerance of the underlying traffic. Once the initialization is performed the data exchange phase begins.

If the Dly-ACK negotiation is not accepted, the receptor should send an ImmACK, and the sender should start again or change the mode transmission

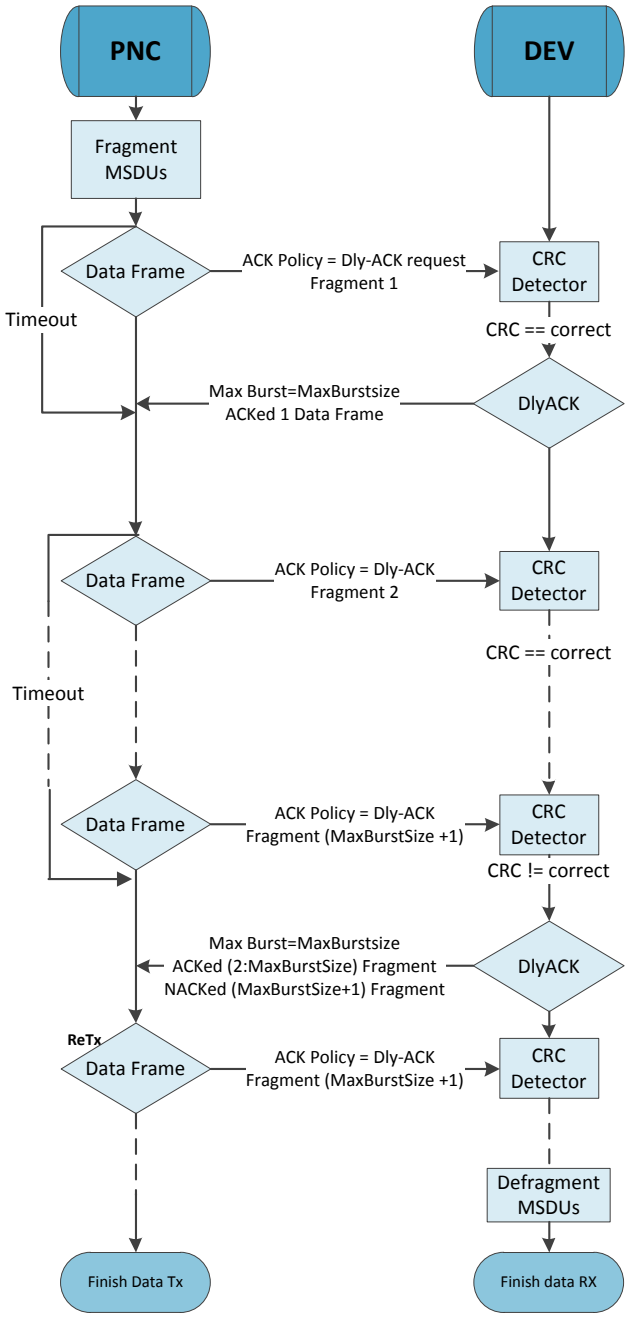


Figure 11 – Delayed ACK acknowledgment mode

In the Figure 11, there is a transmission using the Dly-ACK, on the first data transmission it is just a request, in order to know how many frames could send on the same time. And after be acknowledged it starts to use the bursts, it send a number specific of data frames before receiving the ACK. Then when the burst is finish the receptor would answer it, telling which frames were wrong received.

1.5.4 Implied ACK mode

Imp-ACK mode is used for bidirectional data transmission, it means that in the same CTA stream, both DEVs can transmit data, command frames, etc…

The acknowledgment answer of a Data frame doesn't need to be inside of an ACK frame, that's why, it is implied. It is using the ACK Policy Imp-ACK and the field Imp-ACK NACK for wrong acknowledgment, explained in 2.1.1.1.

If no data or command is needed to send, it can also be acknowledged with an ImmACK.

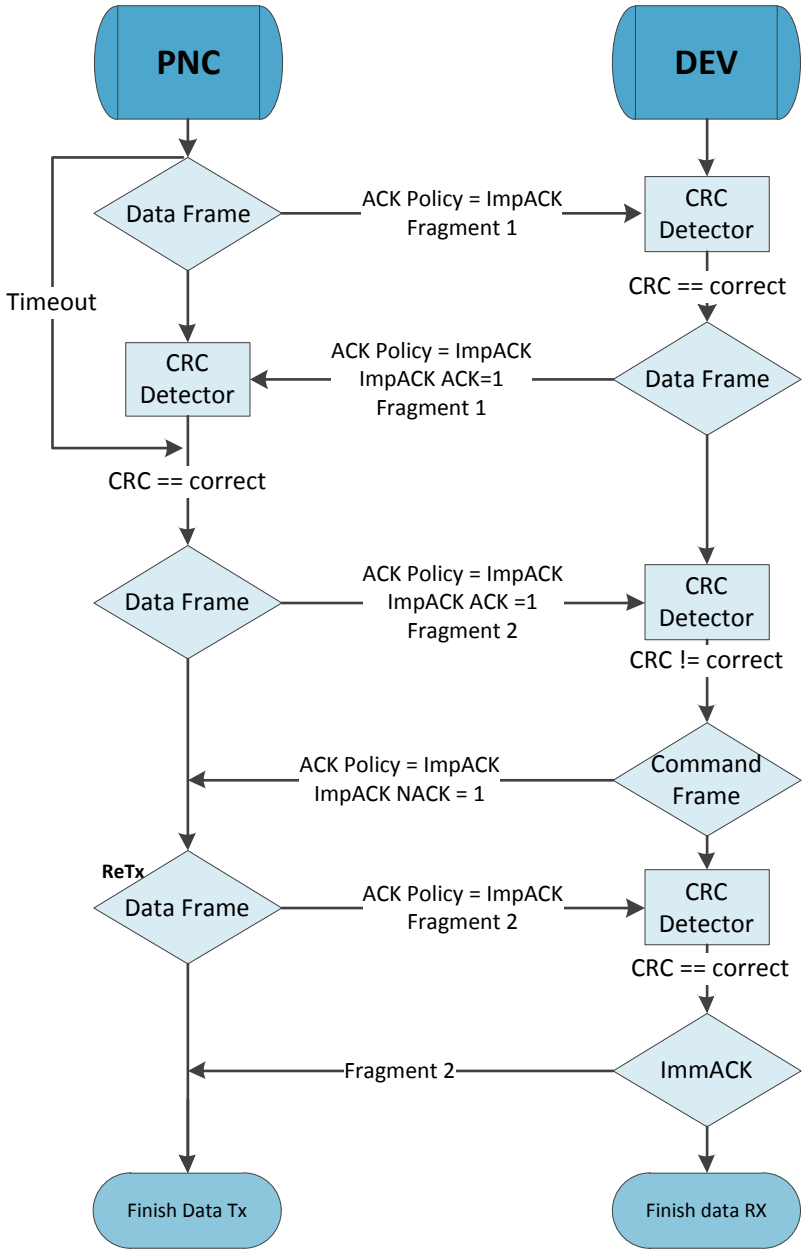


Figure 12 – Implied ACK acknowledgment mode

On the Figure 12, it is an Imp-ACK transmission. It shows how can be possible the bi-directional transmission using the implied acknowledgment. And how can the receptor answer if a data frame was wrong receiver, it can use a negative acknowledgment (NACK).

1.5.5 Block ACK mode

This mode is used to acknowledge the aggregated frames (frames with more than one payload field, explained in *2.6*). It is not a specific type of MAC frame; it is just a Data frame with the ACK policy of BlkACK.

On the aggregation modes, there is one field that is used by the BlkACK. It is a vector of 8 bits, that each of this indicates if the Subframe was well received or if it had errors.
The figure below shows how it works:

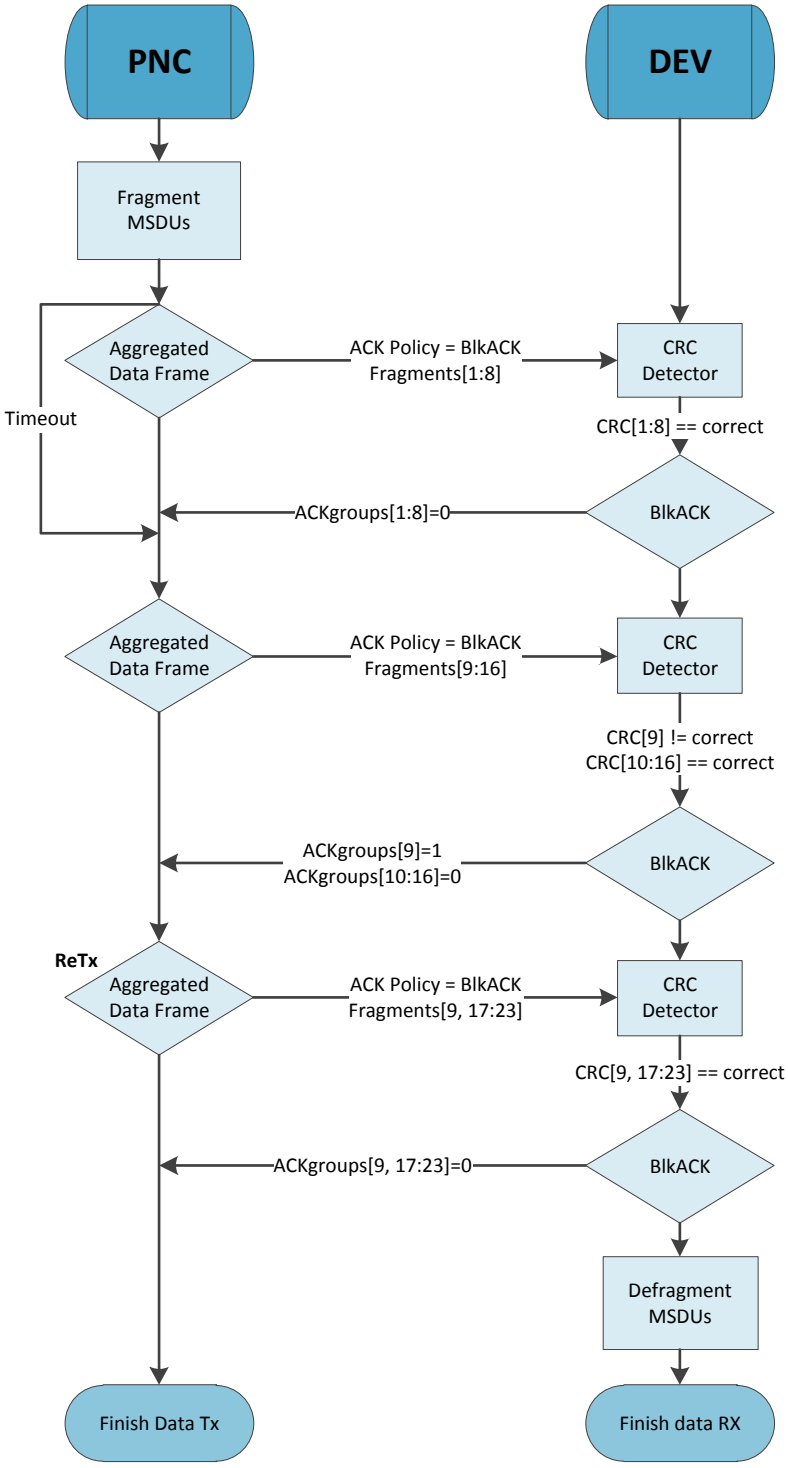


Figure 13 – Blk ACK acknowledgment mode

There is 8 Subframes inside of the aggregated frame, on the first case all the Subframes are received correctly. But on the second transmission there is a Subframe that have any error and it should be retransmitted, then the receptor notify the sender, and it sends again this wrong fragment but which the next fragments that would have to send.

2 MAC frames

There are different kinds of frames specified on the standard, each type of frame have a common part (MAC header) but almost all of them has a specific part for each one (MAC Frame Body).

2.1 MAC general frame

This is the general standard frame, the MAC Header is the same for all the type of frames, but the frame body varies depending on which type of frame is. This frame could have different sizes.

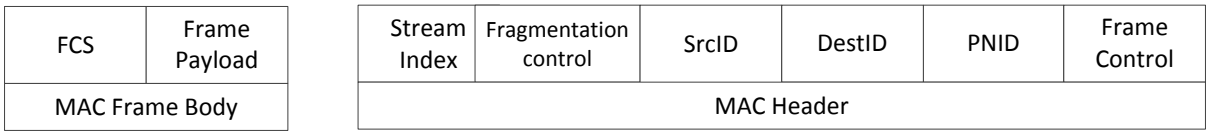


Figure 14 MAC General Format

The MAC Header is always 80 bytes, and is filled with control fields.
The MAC frame body could be changed depending on the frame type.

2.1.1 MAC Header

- PNID: PNC ID
- DestID and SrcID: Destination and source DEV ID of the frame.
- Stream index: The PNC allocates a unique stream index value for each isochronous stream in the piconet.

2.1.1.1Frame Control

This field defines the function of the frame.

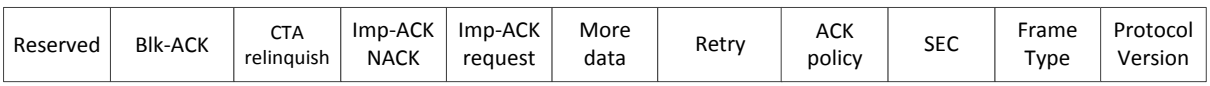


Figure 15 - Frame Control

- Frame Type: Defines the type of frame that is being transmitted, here are the main of the 802.15.3c:
    - Beacon:

  - Immediate ACK
  - Delayed ACK
  - Command
  - Data
- SEC: When is 1 the frame is secured
- ACK Policy, Imp-ACK request and Blk-ACK: These are used to indicate the type of acknowledgment procedure that the addressed recipient or allowed perform. The ACK policy of a frame is determined by the combination of them. There are six types:
  - No ACK
  - ImmACK
  - Delayed ACK
  - Delayed ACK request
  - Imp ACK
  - Block ACK
- Retry: Shall be one if the frame is a retransmission of an earlier frame.
- More Data: Shall be set to zero if the DEV will not use the rest of the channel time in that CTA.
- Imp-ACK (NACK): When the FCS for the frame body failed, it shall be set to zero.
- CTA relinquish: Shall be set to one when the DEV wants to stop using his CTA and give it to another DEV.

2.1.1.2 Fragmentation control

It is used to fragment and reassembly the frames. The table below shows its fields:

Reserved	Last Fragment Number	Fragment Number	MSDU Number

Figure 16 – Fragmentation Control

- MSDU Number: Indicates the sequence number of the current frame. For data frames, each DEV shall maintain one modulo512 counter for each of its isochronous streams and one for asynchronous data traffic.
- Fragment Number: Indicates the order of the fragmented frames. If the frame is not fragmented, it shall be set to zero.

- Last fragment number: Indicates the total number of fragments in the frame. The value should be one less than the number of fragments. It shall be set to zero for all unfragmented MSDUs.

2.2 Beacon frame

The beacon frame is always sent by the Piconet Coordinator.
It can have different included information, depends what the PNC wants to notify, here is the frame:

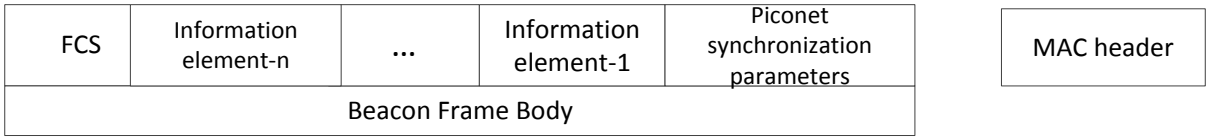


Figure 17 – Beacon Frame

2.2.1 Piconet synchronization parameters

These are the essential parameters for synchronizing the frames:

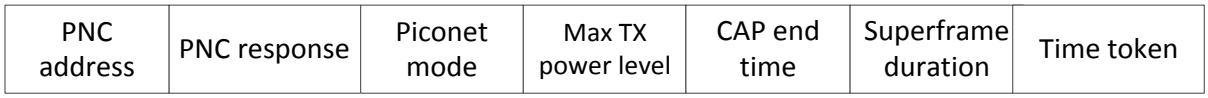


Figure 18 piconet synchronization parameters

- Time token: Counts the number of beacons
- CAP end time: Defines the length of the CAP
- piconet mode: Defines certain characteristics of the piconet and the Superframe
- PNC response and PNC address: Provides information about the PNC.
- Information elements: These are the fields that the PNC uses to inform the DEVs about what is happening on the piconet
- The body of the information elements is the following:

Figure 19 Information Elements

There are different information elements; some of them always have to on a beacon, others aren't always necessary.
These are the main information elements that we are going to work with:

- Channel time allocation IE: This information element is used to define which CTAs use the DEVs and how long they are.
- BSID: The BSID IE is used to provide information on the piconet, like the ID.

2.3 Acknowledgement frames

2.3.1 Immediate ACK (ImmACK) frame

An ImmACK frame is the normal MAC header with the correct frame type (ImmACK), SrcID and DestID, and without MAC body, it is not necessary.

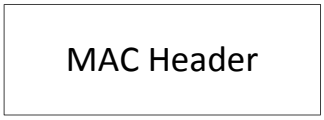


Figure 20 - ImmACK Frame

2.3.2 Delayed ACK (Dly-ACK) frame

The Dly-ACK frame is used like explained in *1.5.3.*

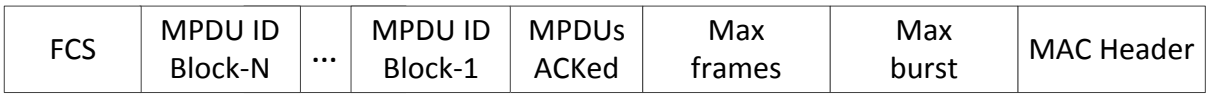


Figure 21 – Dly-ACK Frame

- Max Burst: Indicates the number of frames that may be sent in one burst.
- Max frames: Indicates the maximum number of frames that may be sent before requesting a Dly-ACK from the DEV receiving the frames.
- MPDUs ACKed: shall contain the number of the MPDUs that are being acknowledged.

2.3.2.1 MPDU ID Block

It is used to control the sequence of the fragments and MSDUs

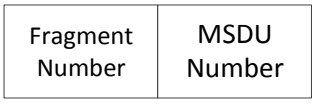


Figure 22 – MPDU ID Block Dly-ACK

2.4 Command frame

When sending command frames the ACK Policy field shall be set to ImmACK or No-ACK. Dly-ACK Request and Imp-ACK Request shall not be permitted.
Here is the structure of the body:

Figure 23 – Command Block

The inside of the Command Block:

Figure 24 – Command Block

The MAC command types are listed below. These frames are used to transmit the commands between a DEV and the PNC: association commands, channel time commands, security commands, etc...

- Association request.
- Association response
- Disassociation request
- PNC information request
- PNC information
- Channel time request
- Channel time response

2.4.1 Association request

This command is used to associate with the PNC. In the MAC Header the SEC field shall be set to zero, the DestID shall be set to the PNCID and the SrcID shall be set to the UnassocID. The Association request command has these fields:

Figure 25 – Association request

- Overall capabilities: Information about the PHY capabilities of the DEV.
- ATP: Association Timeout Period is a maximum amount of time in milliseconds that the association relationship will be maintained in the whole association communication between the PNC and DEV.
- DEV Utility: Some extra parameters asked to the PNC.

2.4.2 Association response

This command is used to respond to the association request between the DEV and the PNC. Only the PNC can use this command.
In the MAC Header, the ACK Policy field shall be set to no-ACK, the SEC field shall be set to zero, the DestID shall be set to the UnassocID and the SrcID as the PNCID.
This frame has these fields:

Reason code	ATP	DEVID	DEV address	Length	Command Type

Figure 26 – Association response

- ATP: This value contains the finalized value for the ATP of the request in milliseconds. It may be different from the one requested by the DEV in its Association Request command, if the PNC is not able to support the value requested.
- Reason code: If the answer is successfully, it has to be 1.

2.4.3 Disassociation request

The Disassociation Request command shall be formatted as illustrated:

Reason Code	Length	Command Type

Figure 27 – Disassociation request

- Reason code: If the answer is successfully, it has to be 1

2.4.4 Channel time request

This frame command may be used to request, modify, or terminate CTAs corresponding with isochronous or asynchronous stream/data traffic. These are the fields:

CTRqB-n	…	CTRqB-2	CTRqB-1	Length	Command Type

Figure 28 – Channel time request

Each Channel Time Request block (CTRqB) corresponds to a channel time request.
If the DEV is making a request for asynchronous channel time, then there shall be only one block.
Each CTRqB is composed by the number of the targets that are addressed the messages of the source DEV, their IDs; it is also composed by the information of the stream that it is transporting. The DEV source requests the number of TUs (Time Unit) that it needs in order to stream. These TUs are units of time that depend on the Rate Factor of the network. In each piconet there are a specific number of TUs in every Superframe

2.4.5 Channel time response

This command frame is used to response on the assigning channel time process, and indicates how many CTAs were assigned.

Reason code	Aviable TUs	Stream Index	Stream request ID	Length	Command Type

Figure 29 - Channel time response

- Available number of TUs: The number of Time Units the CTA has assigned. The Time Units depend on the Rate Factor of the network and the Superframe. In the case of a super-rate allocation, the TUs of the Superframe will be maximal.

For isochronous CTRqs, if the Available TUs are less than the DEV's request for a channel time, the Available TUs will be set by the PNC to the number of TUs that the PNC would be able to set.
For asynchronous stream requests, the response frame is sent only if the PNC is unable to fulfil the request, in which case the available number of TUs fields is set to zero.

- Reason code: If the answer is successfully, it has to be 1.

2.5 Standard data frame

This frame s used on the simplest mode of data transmit, it is just the header and the payload follow with the error checker (FCS).

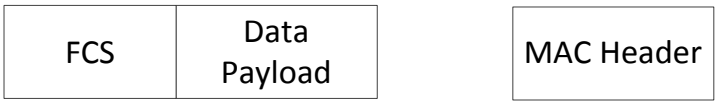


Figure 30 – Data frame

The length of the Data Payload field is limited by the maximum size allowed for the MAC frame body. Null data frames are allowed. For example, a null data frame may be used in a Dly-ACK negotiation.
The FCS is not sent on zero length frames.

2.6 SC and HSI aggregated frame

In this case it is using the aggregation mode, which is an improvement of this standard, it achieves to reduce the overhead of the frame, transmitting 8 payloads on just one frame. It is possible using the Block ACK acknowledgment mode, explained in *1.5.5*.
On mode Single Carrier or High Speed Interface the frame fields are the same. Between these modes, it changes the configuration options (field values) and therefore the speed rate, error control, etc.

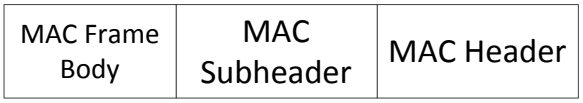


Figure 31 – Aggregated frame SC/HSI

In that case, it has one more Subheader, It is because it has to control more different fields (aggregated) and there are not enough field values on the standard MAC header.

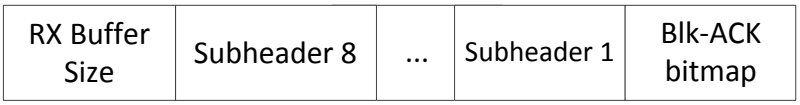


Figure 32 – Subheader SC/HSI

- Blk-ACK Bitmap is a vector of 8 bits that is used for acknowledging the aggregated frames. If the Subframe “n” is correctly received then it transmits a zero on the position “n” of the vector if not, it transmit a 1.

- RX Buffer Size: Indicates the free buffer space at the target DEV.

2.6.1 Subheader field

It shall be formatted like that:

Reserved	Last fragment	Fragment Number	MSDU Number	Skewed Constellation	Subframe Length	Resolution Indication	Retry	FCS Present	MCS information

Figure 33 Subheader Format

- MCS information: Indicates the transmission mode.
- FCS present: It field indicates if there is a CRC present on the Subframe.
- Retry: If the Subframe is a retransmission of a previous frame it field is set to true.
- Resolution Indication: If this field is true, the Subframe Length has a resolution of 512 octets, if not it has a resolution of 1 octet.
- Subframe Length: Indicate the resolution in according with the Resolution Indication.
- Skewed constellation: It shall set to one if a skewed constellation is used.
- MSDU Number: The number of the MSDU contained in the Subframe.
- Fragment Number: The number of the Fragment contained in the Subframe.
- Last Fragment: Indicates if the Subframe is the last of the whole MSDU.

2.6.2 Frame body SC/HSI Aggregated

In this field contains the data of the Frame, Could be a maximum of 8 Subframes and each Subframe is followed with its CRC

FCS n	Subframe payload n	...	FCS 1	Subframe payload 1
MAC SC/HSI Aggregated Frame Body				

Figure 34 MAC Aggregated Frame Body

- Subframe Payload: Contains the data.
- FCS: CRC-bits of the previous Subframe data.

2.7 AV aggregated frame

This type of frame is optimized to carry uncompressed audio and video data; it contains some specific fields that there are information of the video and the audio.

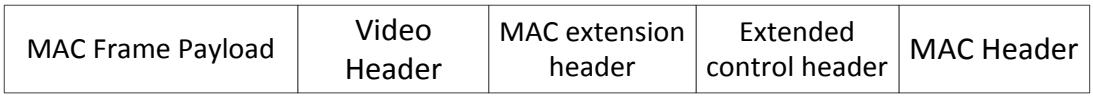


Figure 35 – AV Aggregated frame

2.7.1 Extended control header

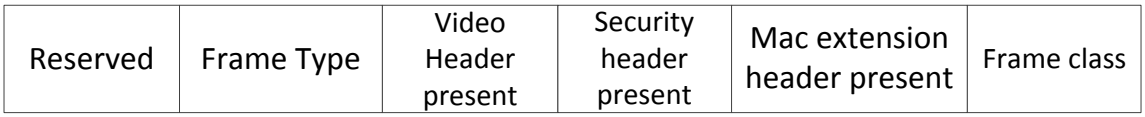


Figure 36 – Extended Control header

- Frame class: Indicate which the class of the frame is.
- MAC extension header present: Shall be set to 1 if there is the MAC extension header.
- Security header present: Shall be set to 1 if there is the Security header.
- Video header present: Shall be set to 1 if there is the Video header.
- Frame Type: Indicates which type of data is transported in the frame, could be mixed.

2.7.2 MAC extension header

This extension is used to control the Subframes that are not good received.

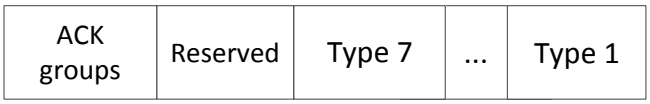


Figure 37 – MAC extension header

- Type 1-7: Indicate the type of data is transported in each Subframe.
- ACK groups: As the field defined in *2.6*, it indicates which Subframes needs retransmission

2.7.3 Video header

This extension is specific to transport video. It supports to control of the video and audio that is transmitted.

Reserved	Video control 4	Video control 3	Video control 2	Video control 1

Figure 38 – Video Header

2.7.3.1 Video control

Reserved	Video frame number	Intelaced field indication	H-position	V-position

Figure 39 – Video control

- V-position: Indicates in which vertical position of the frame pictured is positioned the Subframe that is transported
- H-position: : Indicates in which horizontal position of the frame pictured is positioned the Subframe that is transported
- Interlaced field indication: This bit indicates if the frame pictured that is carried is on the top field or the bottom field.
- Video frame number: Indicates the number of the frame picture that is carried.

2.7.4 Frame body AV aggregated

These fields are equal to the SC/HSI modes. But in each Subframe payload there is one more header, just to control the fragmentation. And the field of the MAC general header called Fragmentation control is used for other time values.

FCS n	Subframe payload n	...	FCS 1	Subframe Payload 1
MAC SC/HSI Aggregated Frame Body				

Figure 40 - AV aggregated frame body

2.7.4.1Subframe payload

Figure 41 – Subframe payload

- Length: Shall be the length of the MSDU payload.
- Sequence number: It is the number of the Subframe transmitted, each Subframe should have a different number.
- Last fragment: Indicate if the Subframe is the last of a fragmented MSDU.
- First fragment: Indicate if the Subframe is the first of a fragmented MSDU.
- MSDU: Here is the data payload.

3 802.15.3c simulator

The simulation has been made with Simulink and Stateflow; these are commercial tools for modelling, simulating and analysing multi domain dynamic systems that run under Matlab, which is a product of MathWorks.

It is based on the representation of the 802.15.3c MAC layer protocol. How two DEVs can establish a network associate and share information.

The simulation is based in these steps:

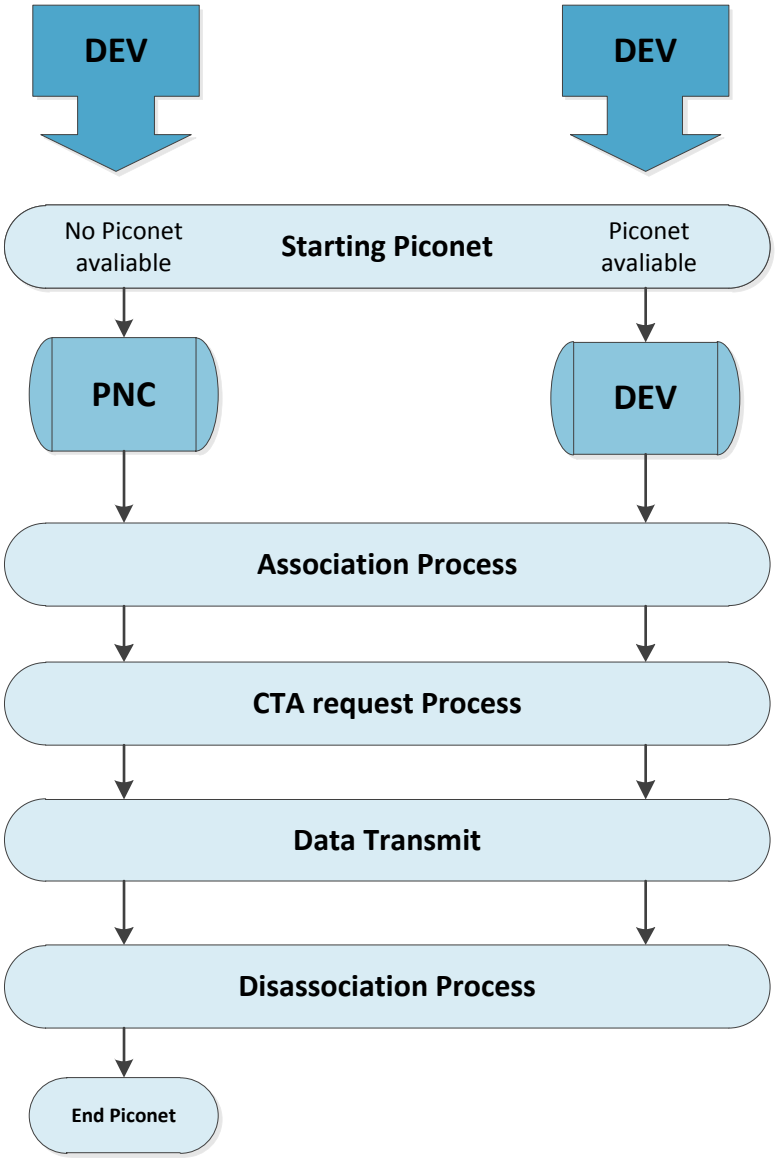


Figure 42 – General simulation

Firstly, there are two users that just turn on the DEVs. Then, there is the "Starting piconet process" where the roles will be decided of both DEVs. The first DEV that was turned on would be the PNC.

After that, there is one that acts like the PNC and other one the DEV. The PNC is the master of the piconet; it is the only one who decides the rules of the network.

If the DEV want to enter to the piconet it has to associate asking the PNC, for that it has to pass through the Association Process and the PNC would accepts it. Then, if the DEV wants to share data with the PNC, it has to ask for a CTA. When the PNC assigns it, the data transmission could start successfully. When the data transmit is over, one of them have to Disassociate and the PNC finishes the piconet.

The next processes are showed with State machines, like it is working on the Stateflow. On each state there are three kinds of variables: the output, the input and the local variables; it means that some variables comes from another blocks of the simulator (like the receptor), these are the inputs, and others go out the State machines, like the type of frame that is sent, these are the output variables. In order to jump between states, it should pass a series of conditions, corresponding on the values of the input and local variables.

3.1 Starting

All the devices that turn on should pass this process. Here will be decided the role of the device, it is explained with State machines:

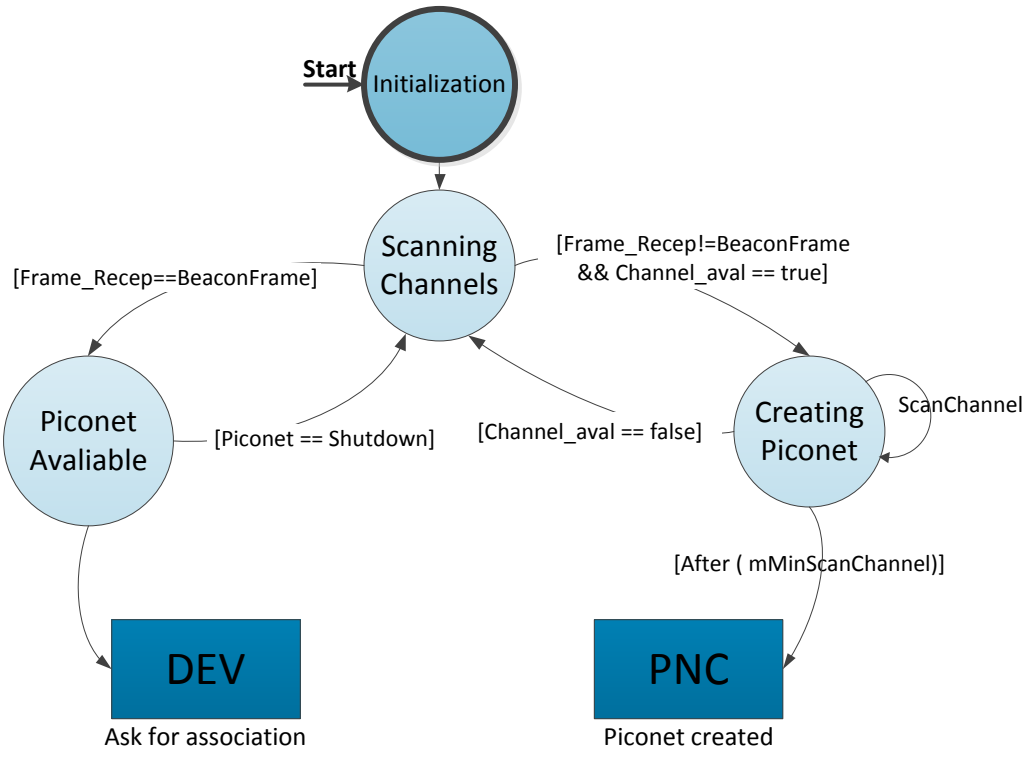


Figure 43 – Starting piconet Stateflow

This first state machine is the same for both users. It depends on who is connected before.

The first DEV that is turned on will be waiting on the state “Scanning channels”, it would not receive any Beacon but it will scan one available channel, then it would stay in the state “Creating piconet” scanning the channel for *mMinScanChannel (period of time)* millisecond. Finally, the piconet would be created and this DEV will become PNC, the master of the piconet, then it has to send Beacons to inform the other DEVs that are its piconet available.

The second DEV that is turned on will enter to the state “Scanning channel” but in this case, it will be informed that a Beacon Frame has been received and it would become the DEV, ready to start with the Association Process.

3.2 Association

3.2.1 DEV Association

This states are the actions that a DEV have to do if it wants to associate with the PNC, it is showed on the figure below:

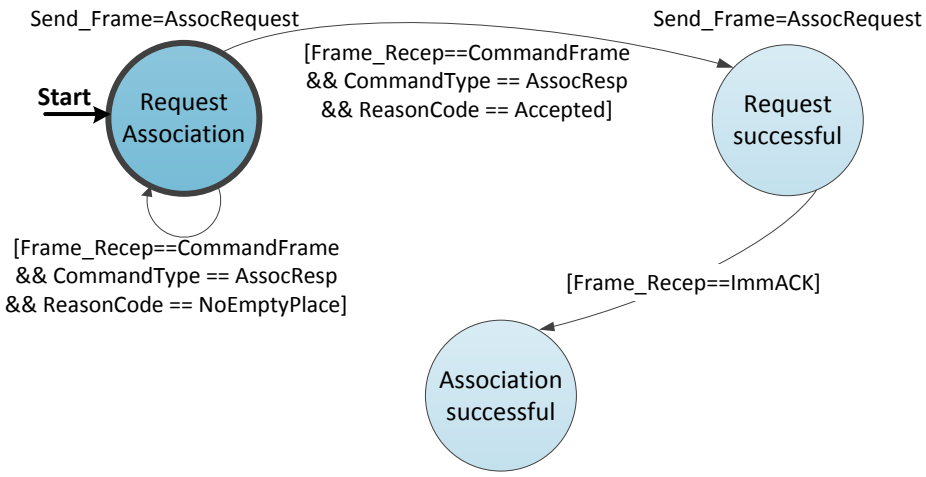


Figure 44 – Association State machine DEV

In this case, the first state of the DEV is Request Association, this state is used to send an Association Request frame to the PNC and wait for and answer of it. If it’s not successful, in this case: No empty place, the DEV enters again in this state and then sends again the Request Association until it is answered Accepted.

When the Reason Code is accepted, it means that the DEV has a place on the association list of the PNC and the three conditions are accepted, then it pass to the next state Request

Successful, here it response with another Association Request in order to provide with all the rest information that the PNC need.
Finally, it will change to the last state if it receives an ImmACK and the Association is successful.

3.2.2 PNC association

If arrives to the PNC one association's request, it should pass through the states defined on the figure below:

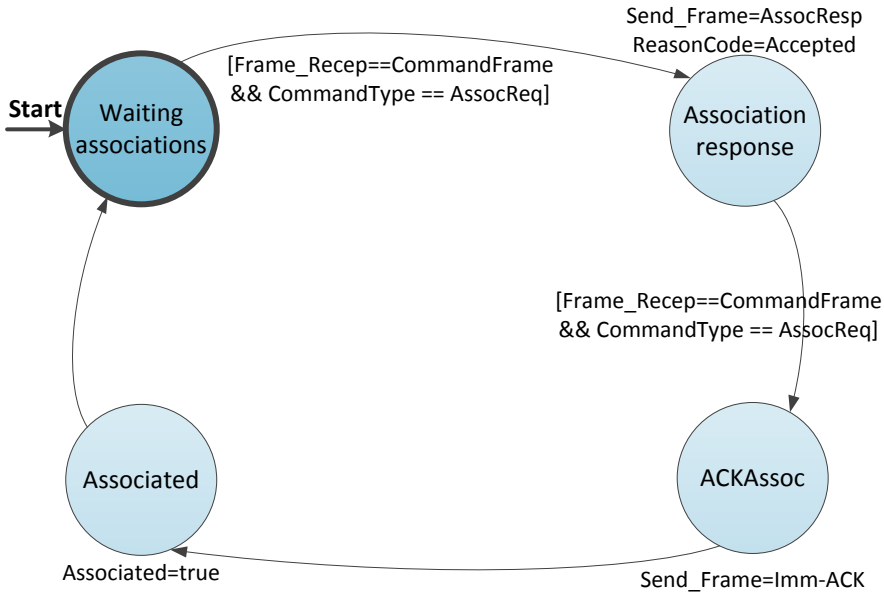


Figure 45 – Association State machine PNC

The PNC is all the moment on the state Waiting associations, in order to wait different request of the new DEVs. The PNC should answer any request with positive or negative answer.
At First, it is waiting for associations until arrive an Association Request, then it change the state and sends an Association Response with accept on the Reason Code. After, it receives one more time the Association Request, but with more information about the DEV. The PNC answer it with the ImmACK, acknowledging that the second request is received.
Finally, it save the DEV and return again to the first state: "Waiting associations".

3.3 Channel Time Allocation

3.3.1 DEV Channel Time Allocation

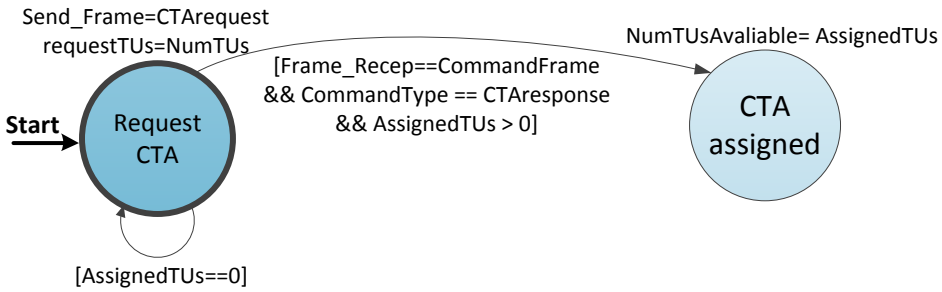


Figure 46 – CTA request State machine DEV

The DEV asks for a specific number of TUs with a CTA request to the PNC. It should wait until the PNC answer him with a Command frame of the type: CTA response. In this response, it's indicated how many TUs were assigned.

3.3.2 PNC Channel Time Allocation

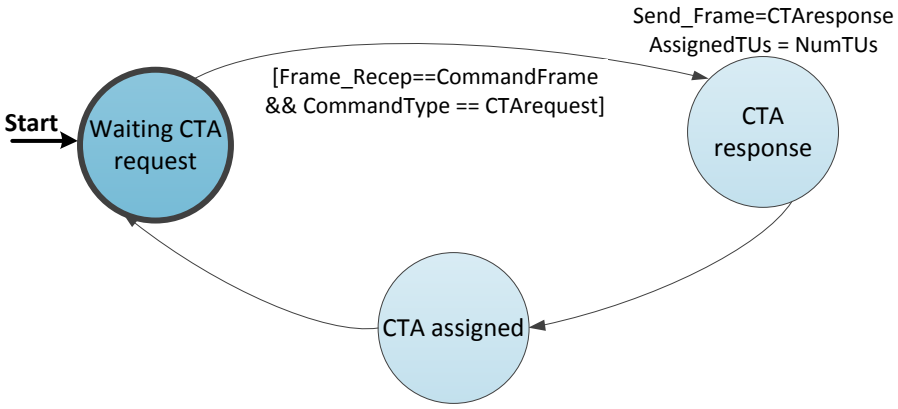


Figure 47 – CTA request State machine PNC

The PNC is all the moment waiting for new CTA requests, in order to wait different request of the associated DEVs. The PNC should answer any request with positive or negative answer.

When it is on the state Waiting CTA request and it receives a Command frame that is the type of CTA request. The PNC check if it is possible to offer the TUs that the DEV are asking for and it responses with a CTA response, indicating how many TUs are assigned.

The last state is CTA assigned, that mean that the previous DEV was served and the PNC can attend for other request on the state Waiting CTA requests.

3.4 Data transmit

In this part, it is simulated four modes of data transmission: Standard mode, SC and HSI aggregated mode and AV aggregated mode.
On data transmission, the DEV and PNC must control the number of fragment and MSDU that are received or sent, in order to know if the frame has been lost or was wrong, and retransmit if it is necessary.

3.4.1 Standard transmission

It is a standard data transfer mode simulator, it is not defined on the standard 802.15.3c but it's made in order to arrive to the aggregated modes.
It can be considered the simplest mode.

3.4.1.1 DEV Standard transmission

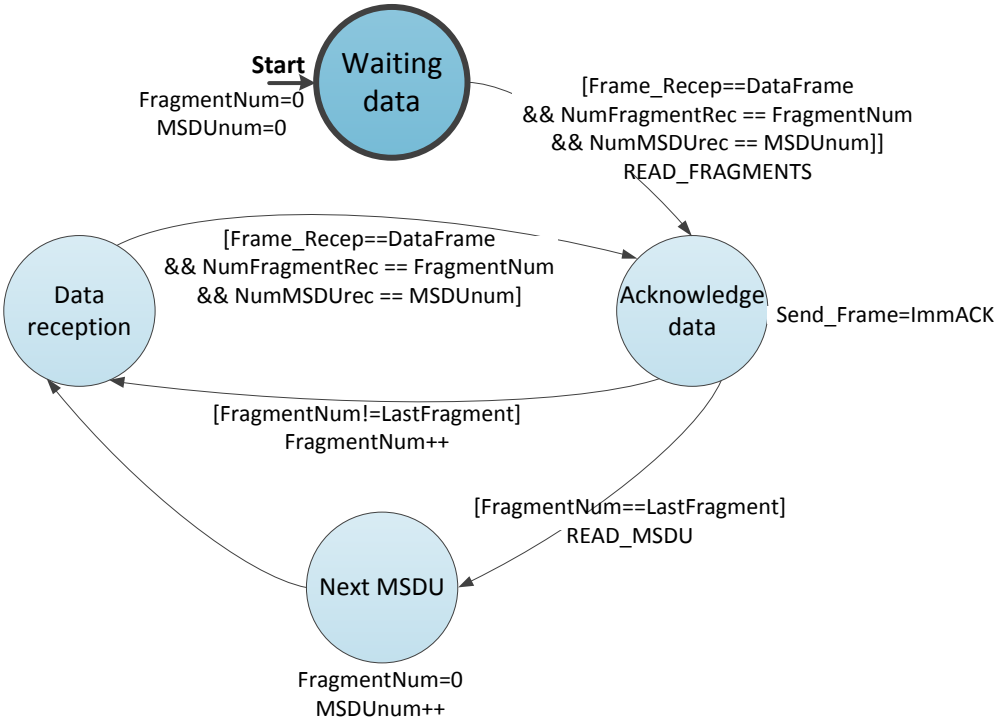


Figure 48 – Standard transmission State machine DEV

Since has been assigned CTAs to the DEV, it can start to share data with the PNC.
The DEV waits on the state Waiting data until a Data frame arrives. When a Data fragment is received and its number of fragment corresponds to the fragment that the DEV is waiting for, it has to start to save those (READ_FRAGMENTS), in order to reorder the fragments when

all of them have been received. Then it comes the state Acknowledge data, in this state its sends an ImmACK frame with the number of fragment and MSDU previous received on the last Data frame, in consideration of making known to the PNC that the Data was well received. And it goes to the state Data Reception waiting for the next fragment.

If a Data fragment is received and it corresponds with Last Fragment of the MSDU, it means that the MSDU was completely received and then it can save the MSDU completely (READ_MSDU). After that it continues receiving data but initializing the count fragment, in order to receive other completely MSDU.

3.4.1.2 PNC Standard Transmission

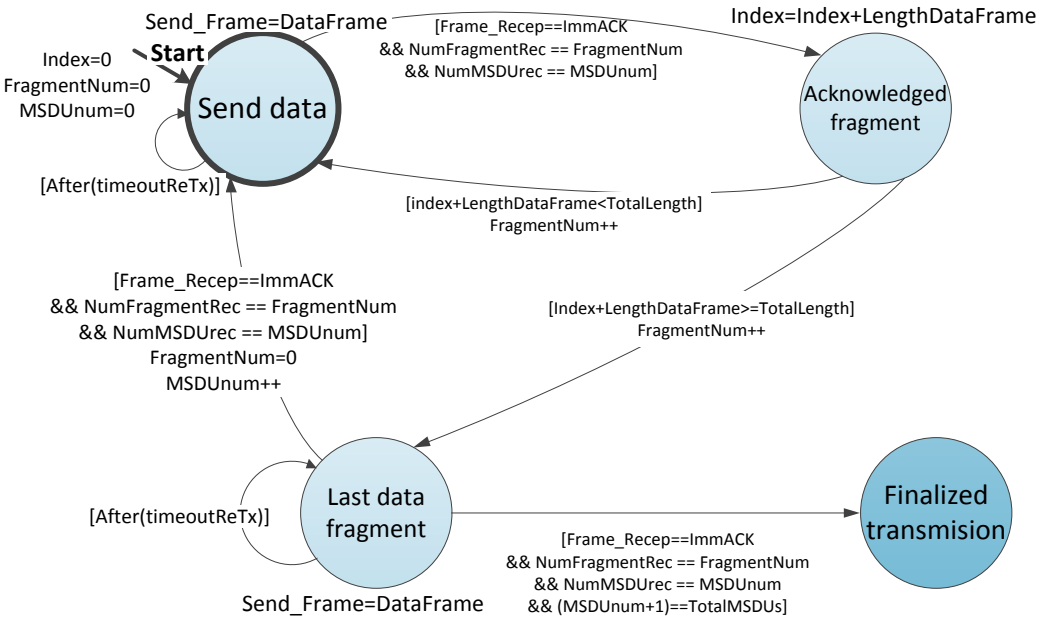


Figure 49 – Standard transmission State machine PNC

When the PNC has a new user associated and assigned CTAs, it can start to send the data.

The first state is Send data; in this state it sends a data fragment with the corresponded fragment and MSDU number, then it waits until an ImmACK is received and compare the fragment and MSDU numbers that comes with this ACK frame. If no acknowledgment is received after *timeoutReTx* (timeout until retransmission), it sends again the last data frame with the same number of fragment and MSDU.

The variable *Index* is used to fragment the source data that is going to be sent. This variable is incremented in *LengthDataFrame*, fragment's size defined by the PNC, each time a data is well acknowledged. When left just one fragment, the *Index* value is on one step to arrive to

TotalLength, total data's size, it changes the state to Last data fragment, in order to send the last frame and initialize the fragment number and increment by one the MSDU number.
If the MSDU number arrives to the limit, the transmission is finished

3.4.2 SC / HSI aggregated transmission

The Single Carrier and the High Speed Interface are aggregated data transmission; it means that it sends 2 or more Subframes in the same frame. Each of these Subframes should be acknowledged individually with the same BlkACK frame.
In this simulator is used a fixed number of Subframes (eight) and if a retransmission is needed, it's sent the whole frame again.

3.4.2.1DEV SC/HSI aggregation transmission

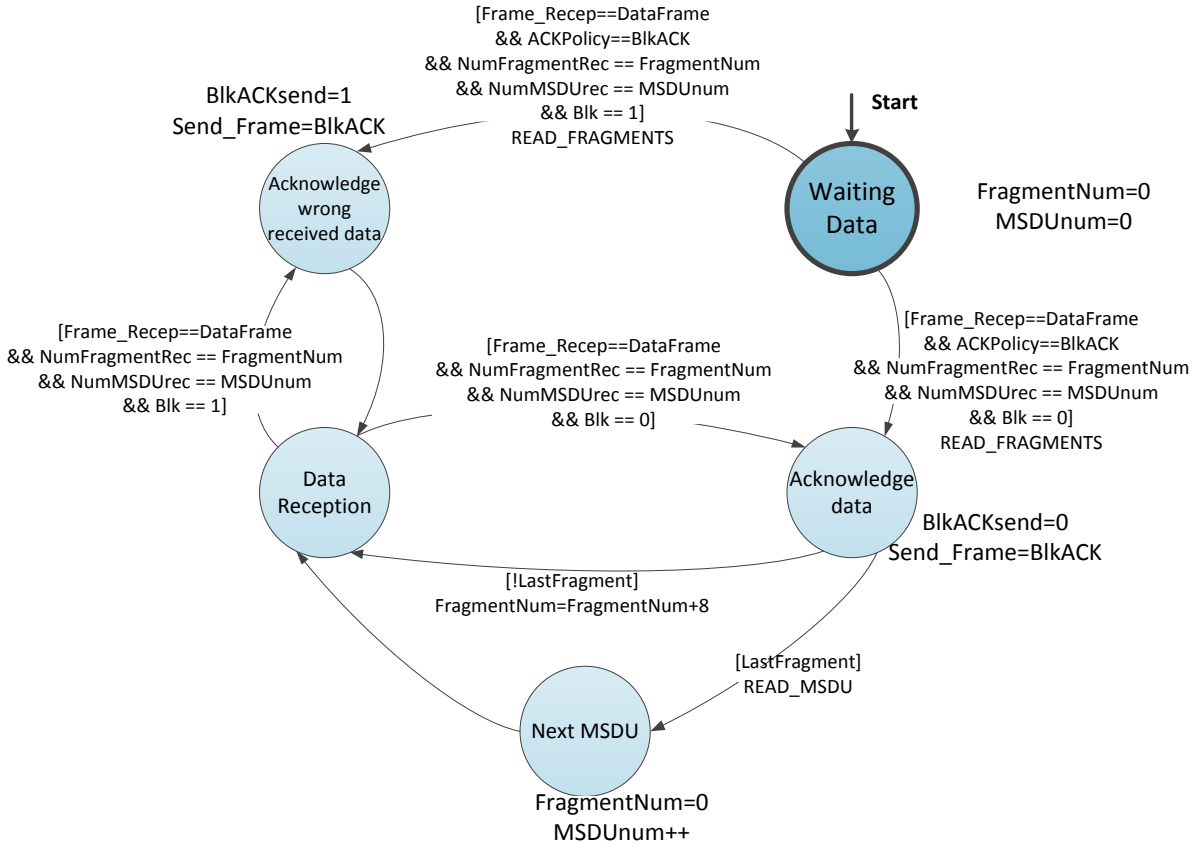


Figure 50 – SC / HSI aggregation transmission DEV

This part of the simulator is almost the same of the DEV Standard transmission mode, defined in *3.4.1*. It has a different acknowledging method, in this case use BlkACK, explained on *1.5.4*. There is also a new state defined "Acknowledge wrong received data", the simulator

arrives to this state when a Subframe of the whole frame have errors (Blk=1). Then, it acknowledges the PNC with a BlkACK indicating that the last data frame contains error, and waits until the frame is sent again.

3.4.2.2 PNC SC/HSI aggregated transmission

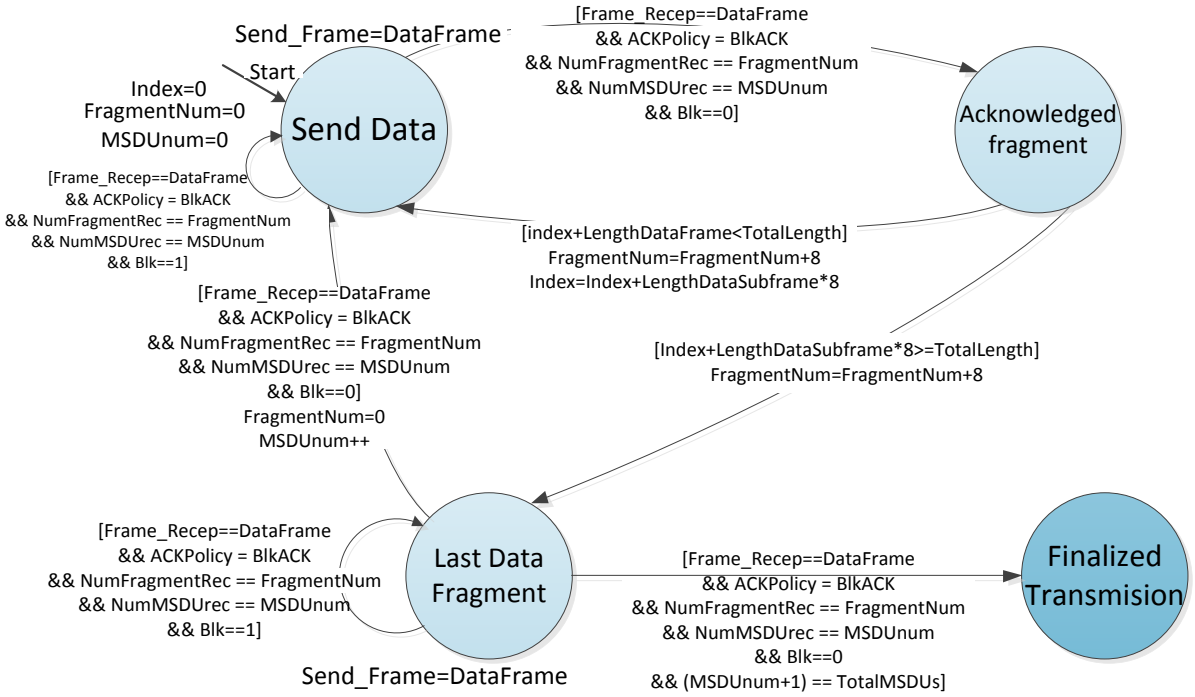


Figure 51 - SC / HSI aggregation transmission PNC

This part of the simulator is almost the same of the PNC Standard transmission mode, defined in *3.4.1*. But it is with the BlkACK acknowledgment mode applied. Then it is not necessary to wait some timeout for retransmit because it's received a data frame with the ACKPolicy set to BlkACK and it contains a variable *Blk* that indicates if the previous data frame was well received by the DEV.

3.4.3 AV aggregated transmission

This part of the simulator is the same like the SC / HSI aggregated mode defined on the previous section, but using 7 instead of 8 Subframes and with some headers that are not relevant on the states of the transmission.

4 Results

In the next pages we can see the results of our simulator. These results are successfully, we have been able to construct a basic piconet structure to test the different transmit modes.

To carry out the simulation, we have used Simulink and Stateflow. These two applications have their own limitation and restrictions. We had to adapt some specific things to make the simulator working. One main handicap was working with fixed size signals, most of the blocks of Simulink are not able to work with variable size signals, and there are some that support this option but it limits a lot the work. Working with triggered subsystems is not trivial, because the blocks that have an execution longer than one sample time cannot be inside of triggered subsystems, like the buffers or the CRC generators. Combining Stateflow and Simulink with triggered subsystems is a delicate topic, in this simulator a loop is made between the blocks of the two DEVs, in order to have one channel for each way of transmission, these blocks are controlled by Stateflow, then can be the possibility that one or more action execute on the same time and it is not valid on the model. Finally working with fixed size signals, make the work more difficult, because frames that are smaller become as large as the data frame are, therefore must be a compensations between the size of the frames and if it results too big, the computer simulates it every time slower.

That is not all negative; using Stateflow with Simulink we can control the logical of the simulator intuitively. The blocks of the Simulink can be managed with Stateflow, activating and deactivating them when is necessary. We can also control the values of the field with the Stateflow.

After pass the Association Process, the CTA request Process, starts the data transmit. We have tested it on the four data transmit modes implemented:

- Standard mode: The simplest mode of the standard 802.15.3.
- Single Carrier (SC) and High Speed Interface (HSI): Aggregated modes using Blk-ACK acknowledge.
- Audio Video (AV): Aggregated mode optimized for transmit uncompressed audio and video, using also Blk-ACK acknowledge.

4.1 Standard mode

This transmit mode is used on the previous standard 802.15.3, it has been implemented in order to compare with the other 3 aggregated modes. Because it is the simplest way, and it was good to start for a basic communication and improving it until arrive to more complex transmission.

4.2 SC/HSI mode

This mode is specific for the 802.15.3c and uses the aggregated improvement transmission. The aggregated method can make the transmission faster, because can be more payload in just one frame than in the Standard mode and it also uses a new method of acknowledgment that allows to retransmit the fragments not consecutively, all this options together makes the whole transmissions more effectively.

4.3 AV mode

The Audio Video mode is specific for the 802.15.3c also. It uses the aggregated mode transmission like the SC/HSI mode. And the difference between them is the specific extra headers that it has for the video.

4.4 Tests

In order to test the simulator, we have drawn some correlated pictures that, together, they act like an animated video. There are three different tests with the three data modes. The first test is a channel without errors, the second test is a channel with error but the receptor doesn't detect these errors and the last one is the fully simulator, the channel contains errors and the receptor detect them.

There are no so much differences to write on the results between the three data modes, the differences that have been detected are written on each test.

4.4.1 Without error

In this test, the channel is not introducing errors on the transmission and consequently there are no retransmissions and the transmission is fluently with normality.
This is a picture of the transmission:

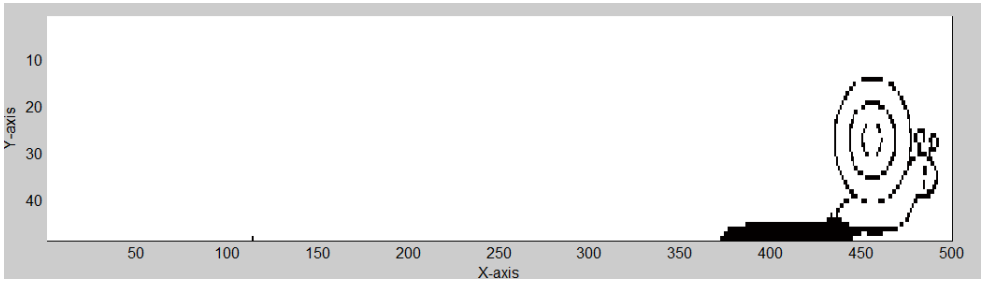


Figure 52 – Picture without errors

The picture is without any error, it is like the source one and there are not delayed movements of the snail.

4.4.2 Without detecting errors

This channel has an error generator on a random frame of the whole transmission. It is just changing some bits of the random frame.

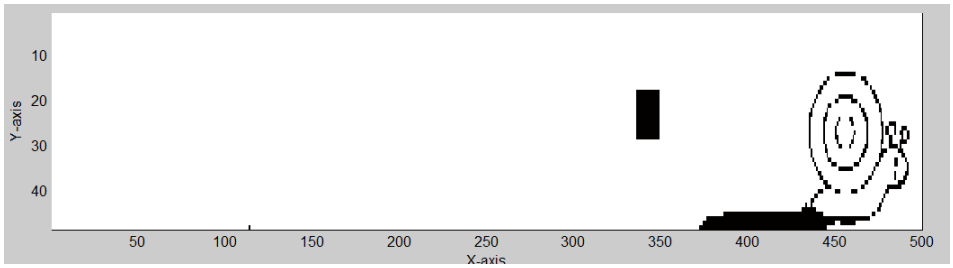


Figure 53 – Picture without error detection

In the previous picture the errors are introduced but the CRC checker is not running, the transmission works with normality, and then there is an error that the receptor doesn't watch out, because it is not taking care of the error, for this reason there is a black square on one picture.

4.4.3 Detecting errors

In this case the simulator is fully working. On the channel there is a block that introduces errors but just before entering to the receptor there a CRC checker that watch out if the frame has any error. Here we have a picture of the example:

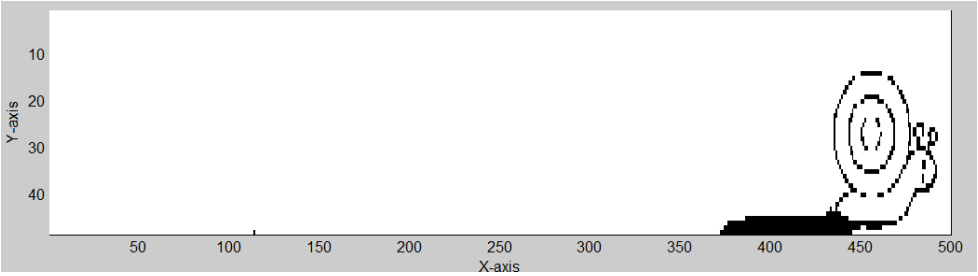


Figure 54 – Picture with error detection

The transmission is working properly but there is a moment with a delay on the procedure of the pictures, because there have been an error and the sender had to retransmit.

On the standard mode, there is a delay on the procedure of the pictures, because there is not a negative acknowledgment and the sender should wait until the timeout expires.

In the SC, HSI and AV aggregated mode, there is also a delay, but this is lower than the standard mode one, because these modes use the BlkACK acknowledgment that is able to notice the sender that a fragment was received wrong, and then the sender can retransmit it immediately without waiting the expiring of the timeout.

5 Conclusions

In this thesis, we have constructed a simulator of the 802.15.3c protocol with its basic applications. We have applied all the processes needed to begin the exchange of information and also four transmit data modes. We saw that it improves the MAC Layer performance of the last standard 802.15.3, it implement more efficient data transmit modes in order to achieve high data rates.

It provides useful support for the high definition video, it has some specific fields only for that that could make the video streaming more easily.

These data modes are implemented for different specific cases. The first one that we have tried is a Standard mode, it is not specified for any connection method but it was useful to understand how a basic data transmit works. The second and the third were the Single Carrier and the High Speed Interface aggregated methods, the SC mode is optimized for low power and low complexity and the HSI one for low-latency bidirectional transfers; between these methods we have seen that there is no different on the MAC Layer part, there are only changes on the PHY layers, but we could compare with the first method and we have seen that the aggregated method is working properly, we can transmit 8 data payloads on the same data frame and answered with only one ACK frame, that highly increment the rate of the data transmit. Finally we have implements the Audio/Video, the specific method for High Definition multimedia data, aggregated method; as we have seen, it has a larger MAC fields that makes it heavier but if the full implementation of video data is done, it would support the encoder and the fragmentation methods, it also using the aggregation method but just with 7 Subframes.

If the desired rate speed is reached, it could be possible to stream High Definition multimedia. The piconet network of this standard offers such good possibilities to maintain an appropriated connection in order to achieve the multimedia streaming.

Using Stateflow was a great tool, it is very easy to implement logical operations with State machine, and working at the same time with Simulink, make it easier, and it opens a big window of alternatives.

Finally the implementation was successfully, the new methods were well implemented and working.

6 Future work

Once the basic simulator is ready, we can also improve some additional extras of the standard, could be interesting to implement more than two DEV in order to construct a bigger piconet with the different types of topologies (Child piconet and Neighbor piconet).

The security has not been implemented in this thesis, and it could be very interesting to know if it affects so much on the overhead, it is working properly and the security code is not breakable

Further could be good to implement or study other features of the MAC, including PNC handover, channel probing and switch, coexistence and power saving modes.

One of the most interesting works that can be done is the implementation of the PHY layer with the MAC layer, joining this two layers, the result could be more interesting because the PHY transmit data modes could be tested and compared.

I think that implementing this protocol in other programing language could be good to compare the characteristics of the network, as well as the data rate, the timestamps and a random number of devices that with the Simulink is impossible to do.

If the PHY Layer is implemented could be interesting study the throughput of the different data modes, and compare it with other standard.

List of Figures

List of Acronyms

DEV	*Device*
PNC	*Piconet Coordinator*
CTA	*Channel Time Allocation*
CTAP	*Channel Time Allocation Period*
CAP	*Contention Access Period*
MCTA	*Management Channel Time Allocation*
MSDU	*MAC Service Data Unit*
TU	*Time Unit*
ACK	*Acknowledgment*
ImmACK	*Immediate acknowledgment*
Dly-ACK	*Delayed acknowledgment*
BlkACK	*Block acknowledgment*
Imp-ACK	*Implied acknowledgment*
SC	*Single Carrier*
HSI	*High Speed Interface*
AV	*Audio Video*
CRC	*Cyclic redundancy checker*
WPAN	*Wireless Personal Area Network*
QoS	*Quality of Service*
MAC	*Media Access Control*
NACK	*Negative acknowledgment*

Bibliography

[1] IEEE Std 802.15.3 – 2003 "Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)"

[2] IEEE Std 802.15.3b – 2005 "Part 15.3b: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)"

[3] IEEE Std 802.15.3c – 2009 "Part 15.3b: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)"

[4] IEEE, "802.15 WPAN task group 3 (TG3)." http://www.ieee802.org/15/pub/TG3.html.

[5] L. X. Cai, *et al*., "Performance Analysis of Hybrid Medium Access Protocol in IEEE 802.15.3 WPAN," in *Proc. 4th IEEE Consumer Communications and Networking Conference*, Las Vegas, USA, Jan. 2007,

[6] "A Simulation Study of the IEEE 802.15.3 MAC", Kwan-Wu Chin and Darryn Lowe. Telecommunications Information Technology Research Institute. University of Wollongong

[7] "Performance Analysis of Uncompressed Video Streaming over IEEE 802.15.3c MAC Protocol", Xizhi An†, Seokho Kim†, Sangkyoon Nam†, Yongsun Kim‡, Wooyong Lee‡, and KyungSup Kwak.

Declaration

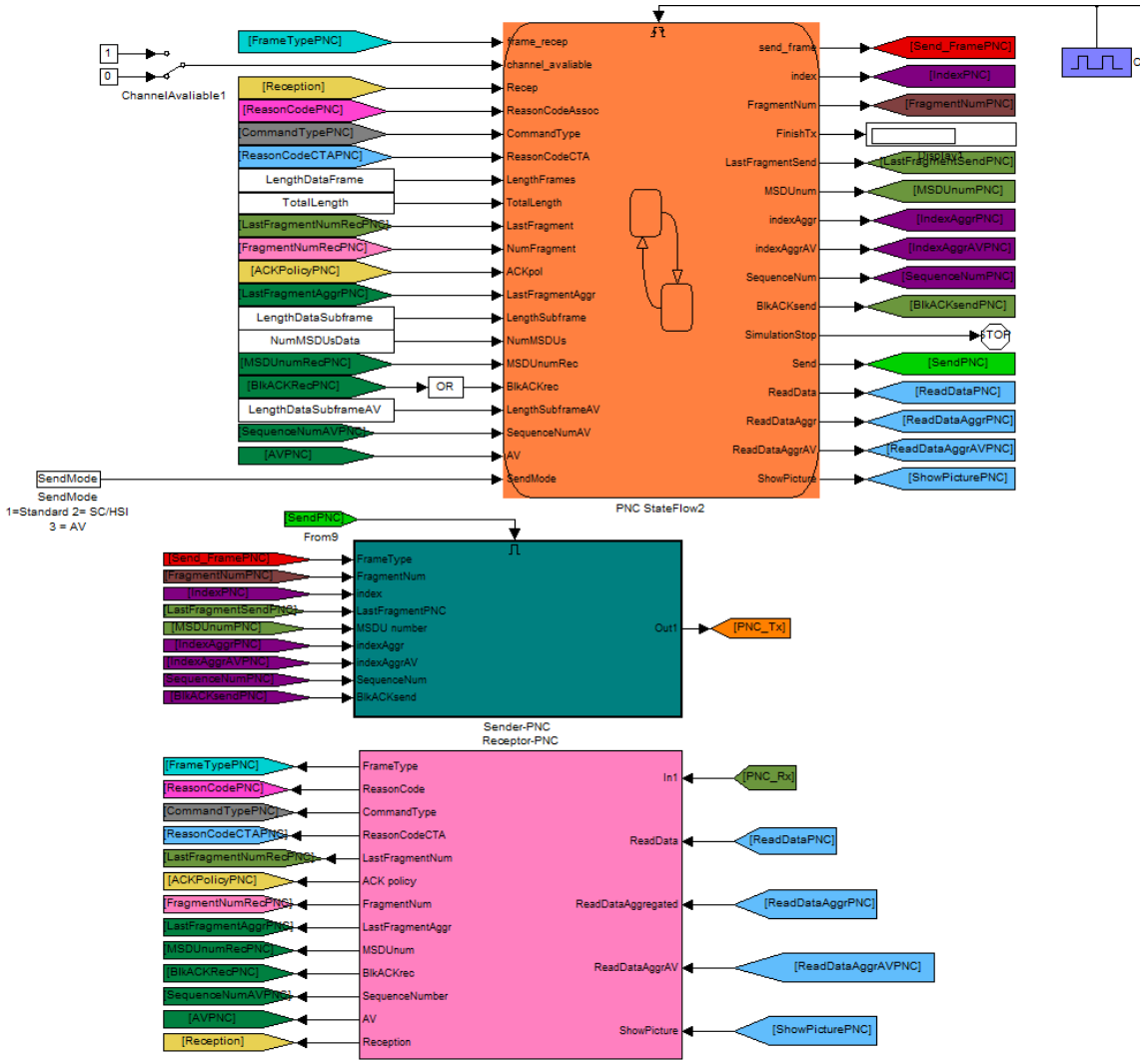
I certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged. Any uses made within it of the works of any other author are properly acknowledged at their point of use.

Alberto Arco Miras

Annexes

General model Simulink

These are the blocks that construct one DEV:



The first block is the Stateflow, this is the block that control all the logical part of the simulator, it decides what the DEV have to do and when. It is directly connected with the other two blocks: Transmitter and Receiver. It has input and output variables. The input variables come directly from the Receptor and the Output variables go directly to the Sender.

The Sender is the second block, which generates the MAC frames; it receives the orders from the Stateflow. It has an enable button that is enabled every time that the Stateflow order to send a frame. Few of the fields of the MAC frames should be update in each frame that is sent, the values of this fields come directly from the Stateflow, that depending the state will be different values.

The Receptor is the third block, it is the responsible of receive the frames and interpret them. It has a selector to decide the type of frame that has arrived, in order to know how to read it. All this read information goes directly to the Stateflow that decides which is the next step. This block is the third important block of the model, the device receive all the frames on this block and it has to which type of frame is received and according with the type, which block use to read it. Depending the frame, the information is different and change the positions, the header is always the same, this is the field where can see which type of frame it is, because it is always read on the same way. The variables on the right are used to carry the information to the Stateflow, in order to decide which the next step is. The last variable, variable "Reception" Is the variable used to know if a frame has arrived, it makes a sum of all the bits of the frame, and if it is positive, it means that a frame has arrives, if it is not a frame, it will give zero.

The blue variables on the left are coming from the Stateflow, they are used to reconstruct the MSDU and to show the images that are coming.

The button on the left of the Stateflow is used to know which device is going to be the Piconet Coordinator, one should found one channel available and the other no, who found the channel available, would create the piconet, and who not found the channel, will wait on the first state until a beacon of the PNC arrives.

Now are shown the variables used on the first block (Stateflow).

First the tags on the input, that are variable that come from the Receptor Block:

- FrameType: This variable is used to know which type of frame is received. If it is a command frame, a beacon, or a data frame, etc… This is a field of the frame control on the header
- Reception: It is Boolean that notify when a frame arrives
- ReasonCode: It is the reason code of the response frames, if the association was successful or not, this is a field of the association command frame.
- CommandType: This is used for knows which type of command is a command frame. Association request, association response, cta request,etc…, this is a field of the command body.
- ReasonCodeCTA: It is the answer on the response of the CTA process, if it was successfully assigned or not, this is a field of the CTA command frame.

- LastFragmentNumRec: This variable is used by for the Standard mode data frames, to know which the last fragment number of the MSDU is, it is a field of the fragmentation control on the header.
- FragmentNumRec: this a variable used to know which fragment number is receiving. It is a of the fragmentation control on the main header, or also can be the field of the fragmentation control of the Subheader of the aggregated frames.
- ACKPolicy: This variable is used to know which is the ACK policy of the fragment that has just arrived. On the Stateflow is useful to know if it is a standard or an aggregated frame. It is a field of the frame control on the main header.
- LastFragmentAggr: This is a Boolean that is used to know if the last fragment received if the last fragment of a MSDU, It is a field of the fragmentation control of the Subheader of the aggregated frames.
- MSDUnumRec: This is used to know which number of MSDU has just arrived. It is a field of the fragmentation control of the main common header, it is also a field of the fragmentation control of the Subheader of the aggregated frames.
- BlkACKRec: This variable is used to know the answer on the BlkACK frame, that this acknowledgment frame is sent by the receptor when an aggregated frame has just arrived. It is a field of the BlkACK frame.
- SequenceNum: This variable is used to know in which number of frame we are. It is a specific field of the specific header of the AV aggregated frames.
- AV: This field is used to know if the aggregated frame received is an AV frame or an SC/HSI frame.

Now, these are the constant values on the input of the Stateflow:

- LengthDataFrame: This constant is used to know which is the length of the standard data frame payload defined on the script “General.m”.
- TotalLength: This constant defines the total length of the MSDUs that is going to send, in order to know how many fragments will be, each MSDU is one picture in this case.
- LengthDataSubframe: This is used to know which is the length of the Subframe payload of the SC/HSI mode defined on the script “General.m.
- NumMSDUsData: This is used to know how many MSDUs is going to send, or, like on this case, the number of pictured.

- LengthDataSubframeAV: This is used to know which is the length of the Subframe payload of the AV mode defined on the script “General.m.
- SendMode: Indicate wich mode of transmission is going to be used

On the Stateflow there are also output variables, this output variables goes directly to the Sender block, in order to control the frames that are going to be sent:

- Send_Frame: This goes directly to the Sender block, in order to select the frame that is going to send.
- Index: This variable indicates the part of the data that is going to be sent. It is used to fragment the data, and select the correct fragment, and if the same last frame should be sent again (retransmissions)
- FragmentNum: It is used to assign a fragment number to the fragmet that is going to be sent.
- FinishTx: It is only and indicator, now it indicated how many frames are sent.
- LastFragmentSend: It is for the aggregated data frames, it is used to indicate that the frame that is going to send is the last of the MSDU
- MSDUnum: Indicate the number of the MSDU of the data fragment that is going to be sent.
- indexAggr: The same of the previous *Index*, but for the data of the SC/HSI aggregated frames.
- indexAggrAV: The same of the previous Index, but for AV aggregated frames
- SequenceNum: Indicated the number of the frames that is going to be sent, only for AV frames.
- BlkACKsend: It indicates if a Data frame was received with errors or not, it is used for BlkACK frames.
- SimulationStop: It stop the simulation when the disassociation process is completed.
- Send: It is a trigger value used by the Stateflow in order to enable the block Sender to transmit the selected frame.

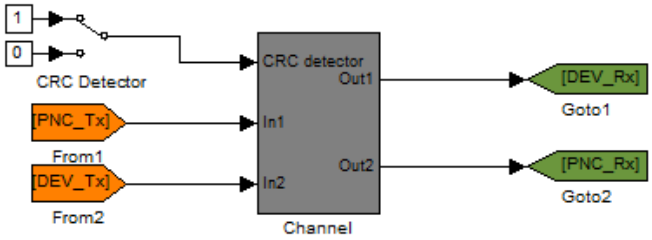
The last four tags are used by the Receptor block, in order to read the data that has arrived an reconstruct it:

- ReadData: It is used when a standard data fragment has been just received, in order to read the payload and save it.
- ReadDataAggr: This is like the previous one, but for SC/HSI frames.
- ReadDataAggrAV: This is like the previous one, but for AV frames.

- ShowPicture: This event is used when a MSDU is completely received, then it constructs it and is showed on a matrix viewer.

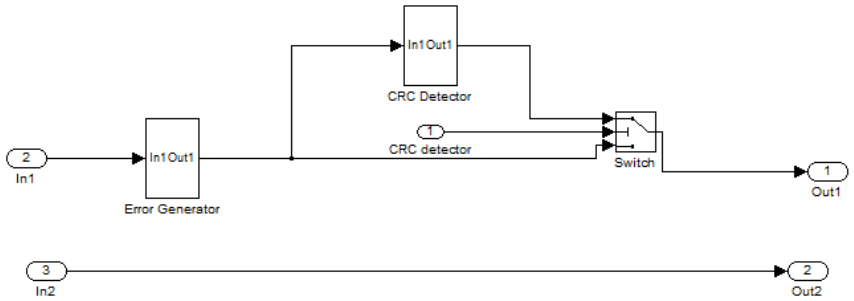
Channel block

Here is the channel block on the General mode:



This block has two different connextion, one from the Sender of the PNC (PNC_Tx) to the Receptor of the DEV (DEV_Rx), the other from the Sender of the DEV (DEV_Tx) to the Receptor of the PNC (PNC_Rx). It should be two ways in order to transmit in both directions. The button that is on the channel block, it is used to switch on the CRC checker, it is useful to see the errors that the channel is generating on the video.

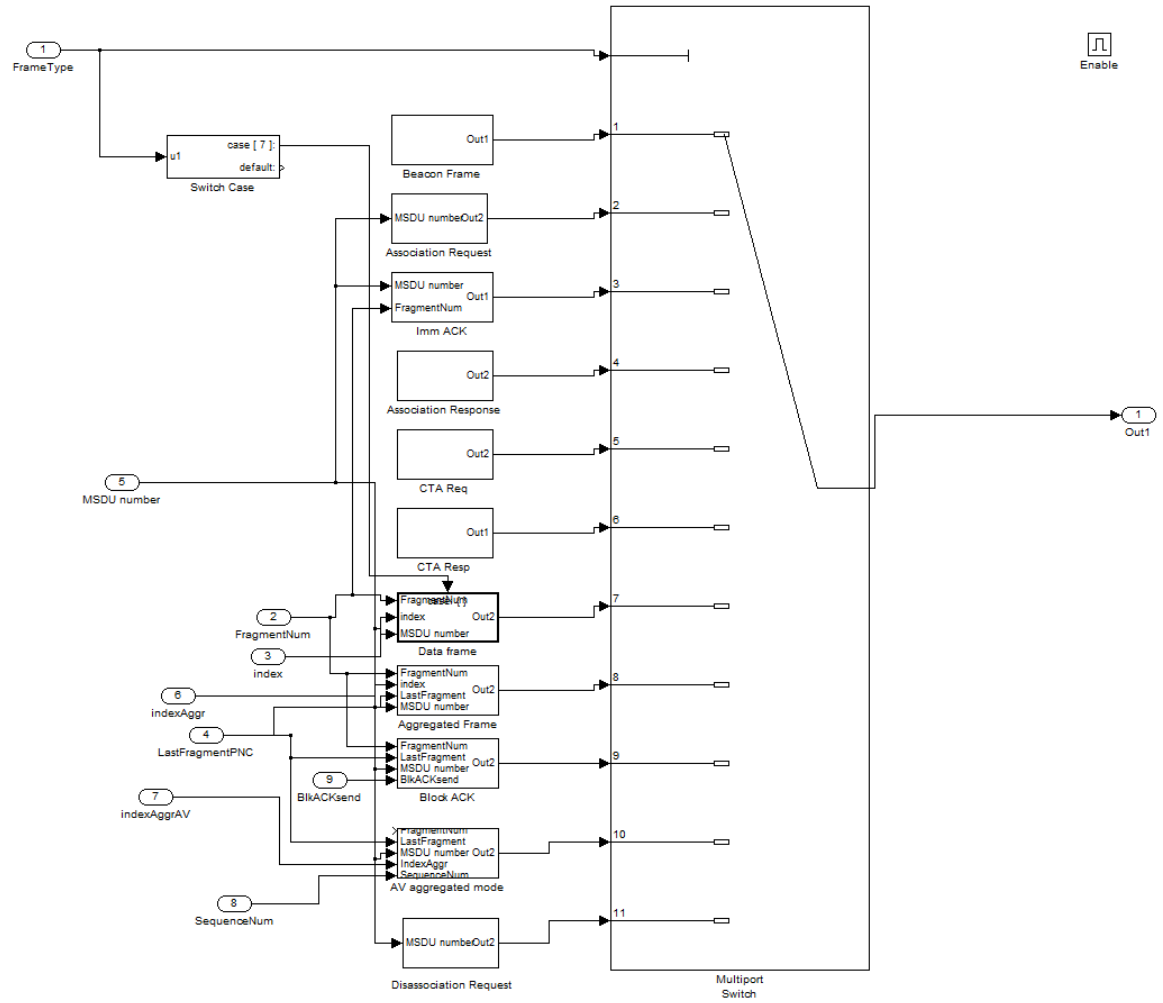
Inside the channel block:



The first way is the way from the PNC_Tx to the DEV_Rx. The first block, Error Generator, is used to change the payload of the data frames, in order to generate the errors, this block has a counter of the frames that pass through it, when it arrives to 31 (can be changed) in this case a part of the payload is overwritten.

The second block is the CRC detector, that the frame pass through depending on the input variable CRC detector, if it is positive, the CRC detector is working, if not, the frame on the other way without checking the CRC. The CRC detector is working with a block of Simulink that is able to check if the CRC is correct.

Emitter block



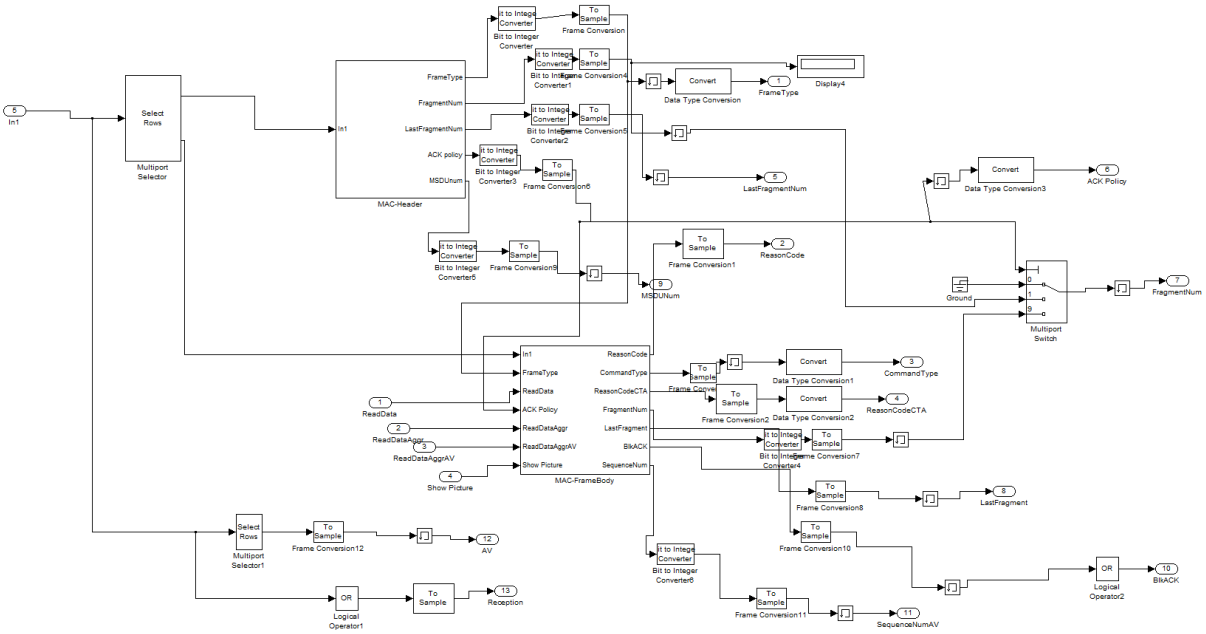
This is the emitter block, it is coordinated by a “Multiport Switch”, that decides which frame allow to pass using the input variable “FrameType”. Each frame has a port number on the switch. The frames should have the size, in order to not working with variable size signals, as we said on the document. The variable “FrameType” is coordinated by the Stateflow of the specific device that we are looking.

The input variables that are on the left are variables that the state machine is controlling, like the number of fragment and MSDU. The frames structure can be modified but taking care not changing the size of the whole frame, it should be the same like the other, each frame has a field of padding where there is no useful information and it can be adapted depending what the user want to add on the frame.

To turn on this block, the Stateflow should switch on an enable then, each time that it wants to send a frame, it send a signal to the enable, in order to active it.

Receiver block

Here is a screenshot of the blocks inside the Receiver block:



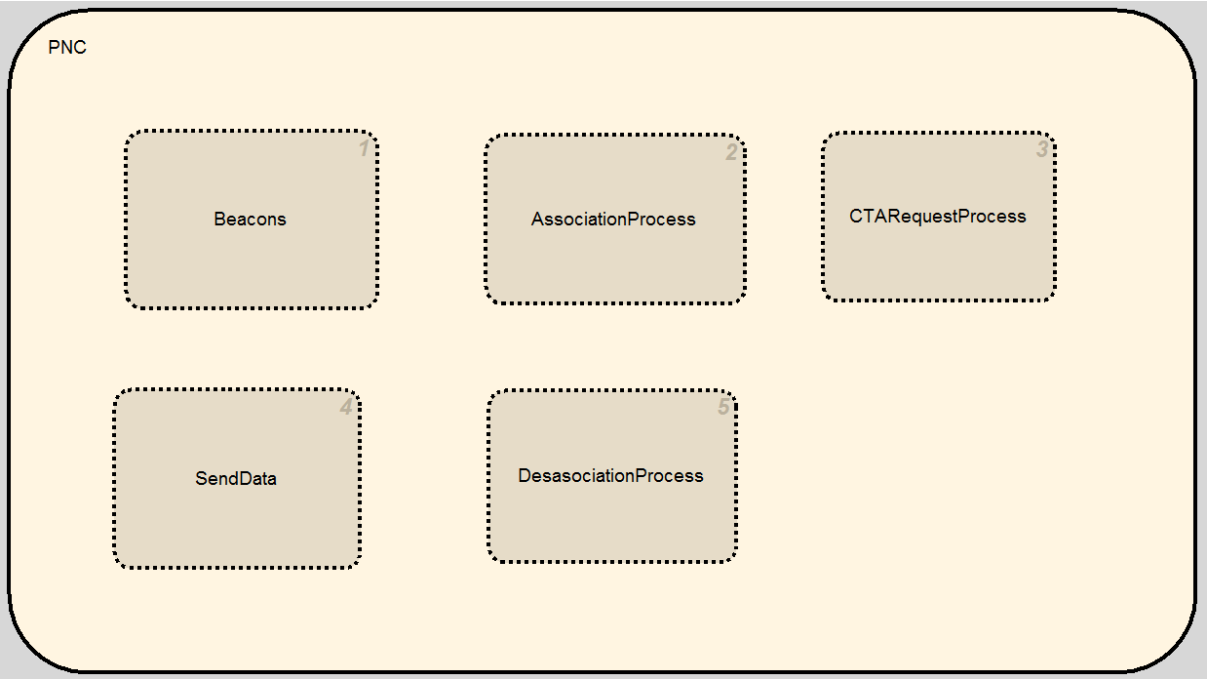
This block is the manager to read all the frames that arrive, as is explained on previous sections of the annexes. The most important variable input of this block is the In1 that is where all the frames are coming.

The two bigger blocks are the most important:

The first one is called MAC-Header, this block reads the headers of all the frames that are arriving, giving information like the type of the frame (FrameType), the mode of acknowledgment (ACKPolicy) and all the information about the fragmentation control: the number of fragment and MSDU (FragmentNum, MSDUnum), and the number of the last fragment of the MSDU (LastFragmentNum).

The second block is the MAC-FrameBody, it is used to read the part of the body of the frame. It is divided in some blocks that are enabled depending the frame type (FrameType), depending on if is a data frame, a beacon, a command… it has to read the frame body in different ways. There are also some blocks that are used to save the data payload that are arriving depending in which modes are working. And one more block that is used to reshape and show the images.

PNC State machines



This is inside the Stateflow on the part of the Piconet coordinator, this 5 states are parallel between them, and they have more state inside them. They are parallel because the PNC should always be listening for new request, then at the same time that it is sending beacons, it has to be able to associate a DEV and to send data frames. They are ordered by execution if a new DEV wants to join the piconet.

Each of this process is explained on the part *3* of the document. How it is explained on the document and before, the state machines have output, input and local variables, these are used to control the other blocks.

The first state is just used to send Beacons each period of time.

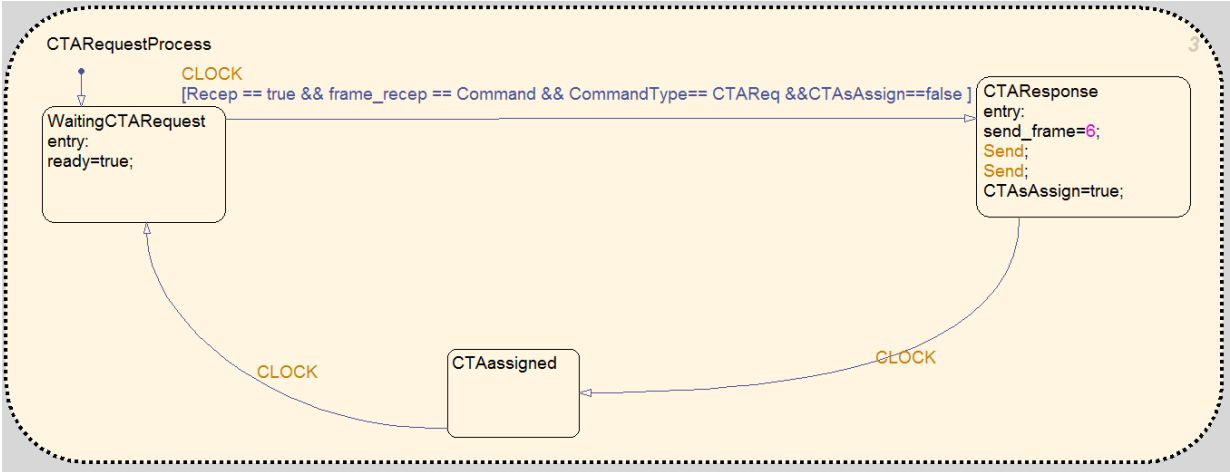
The second State, the Association Process, has some sub states, which are showed on the figure below:

The PNC is all the moment on the state Waiting associations, in order to wait different request of the new DEVs. The PNC should answer any request with positive or negative answer.

At First, it is waiting for associations until arrive a frame that the frame type (frame_recep) is a Command and the command type (CommandType) is an association request frame (AssocReq), then it change the state and sends an Association Response with accept on the Reason Code, in order to send this frame, the state should specify which number of frame is (send_frame) in order to select it on the Sender block and enable the Sender block with the variable Send. After, it receives one more time the Association Request, but with more information about the DEV. The PNC answer it with the ImmACK, acknowledging that the second request is received.

Finally, it save the DEV and return again to the first state: “Waiting associations”.

The third state is the CTA process:

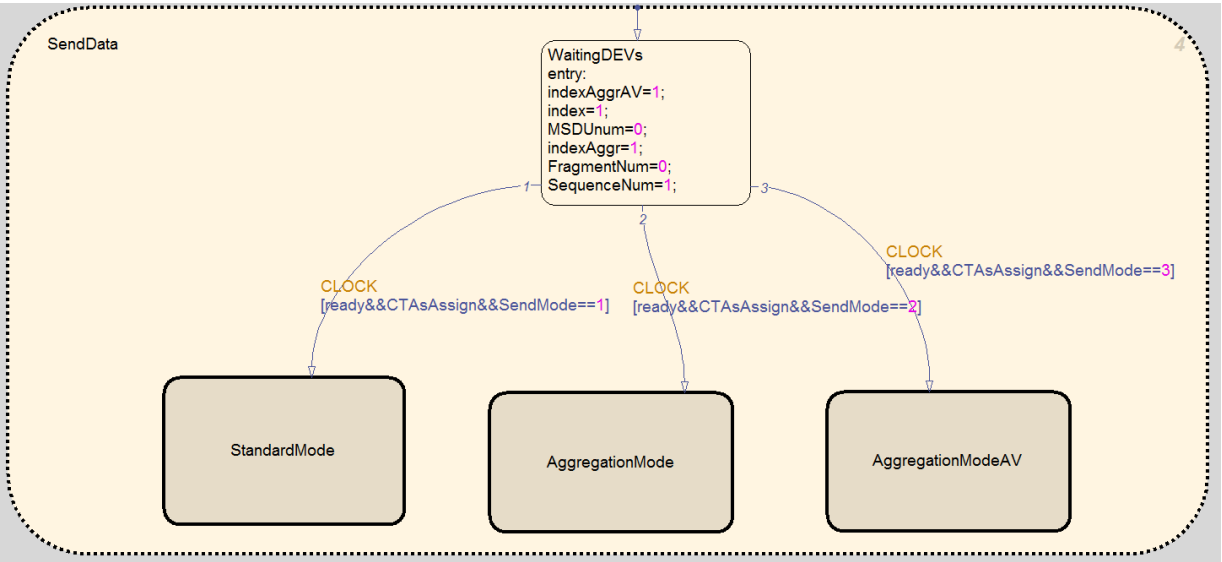


The PNC is all the moment waiting for new CTA requests, in order to wait different request of the associated DEVs. The PNC should answer any request with positive or negative answer.

When it is on the state Waiting CTA request and it receives a Command frame that is the type of CTA request. It should send an CTA response selecting the number of the frame an enabling the Sender, with the answer on the CTAsAssign that will be the ReasonCodeCTA.

The last state is CTA assigned, that mean that the previous DEV was served and the PNC can attend for other request on the state Waiting CTA requests.

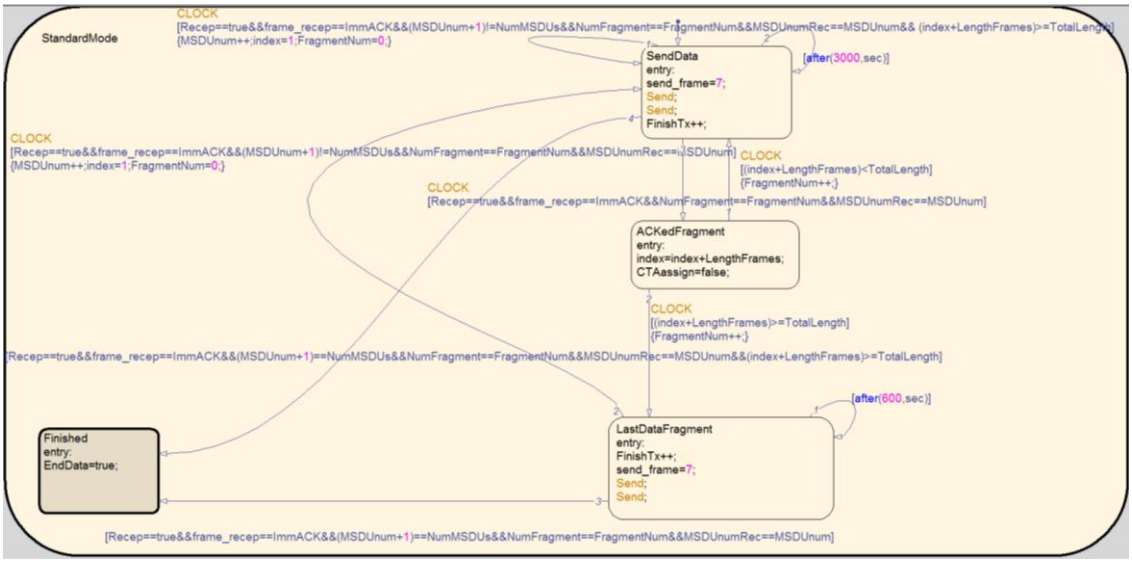
The fourth block is used to send frames, it has the three modes inside:



The first state is used to initialize the variables that are going to be used on the data transmitter. In order to choose which method of transmission is going to be used, The variable SendMode is used, that is an input constant variable of the Statefow. The variables read and CTAassign are used to know when a DEV is associated and if it has CTAs assigned.

After with the variable SendMode, the states changes to 1 of this modes that are explained on the next page:

These are the StandardMode sub states:



When the PNC has a new user associated and assigned CTAs, it can start to send the data.

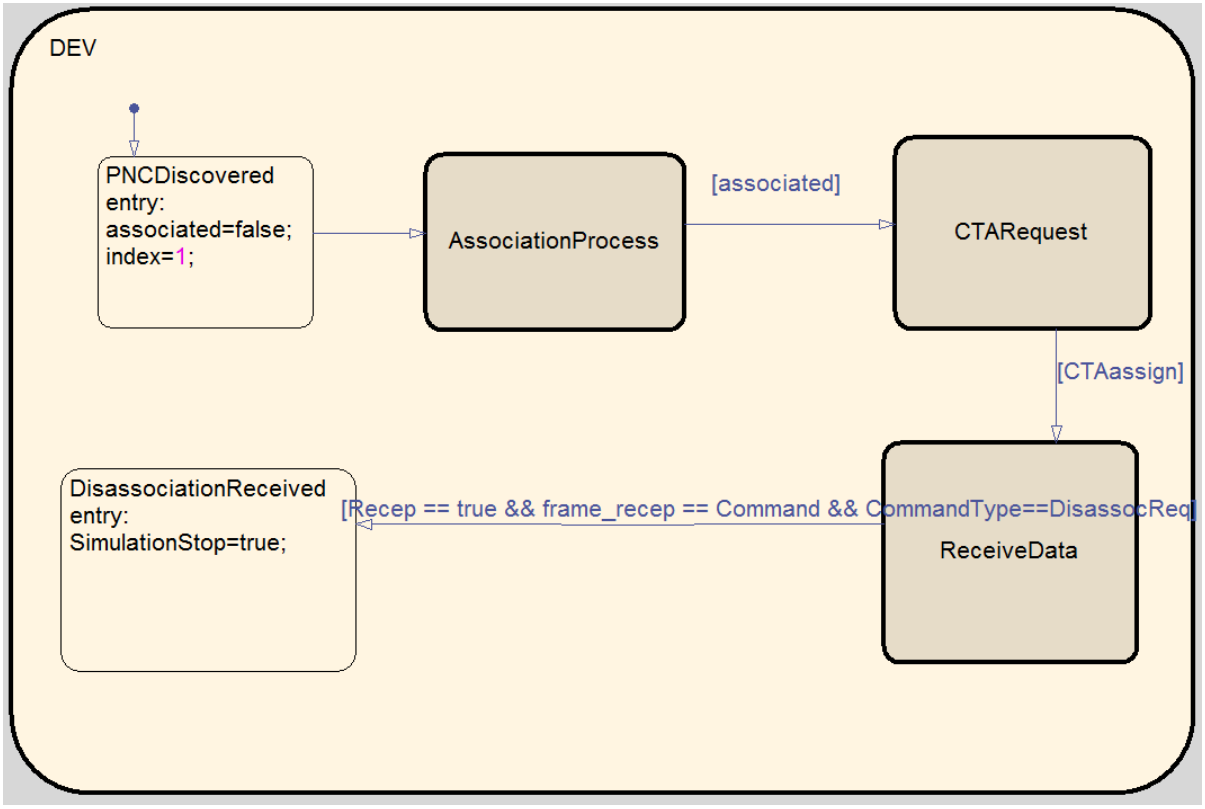
The first state is Send data; in this state it sends a data fragment with the corresponded fragment and MSDU number, the first time decided on the parent state, then it waits until a frame with the type ImmACK is received and compare the fragment and MSDU numbers that comes with this ACK frame, if they were the same like the previous data frame sent, the acknowledgment is correct. If no acknowledgment is received after *timeoutReTx* (timeout until retransmission), it sends again the last data frame with the same number of fragment and MSDU. Could be the possibility that the MSDU don't need to be fragmented, then on the first state, there another way in order to send the data without fragment, don't using the state Last data fragment, an also going directly to finished when the MSDU number expires.

The variable *Index* is used to fragment the source data that is going to be sent. This variable is incremented in *LengthDataFrame,* fragment's size defined on the "General.m", each time a data is well acknowledged. When left just one fragment, the *Index* value is on one step to arrive to *TotalLength*, total data's size, it changes the state to Last data fragment, in order to send the last frame and initialize the fragment number and increment by one the MSDU number.

If the MSDU number arrives to the limit, the transmission is finished

The SC/HSI and the AV aggregated modes are the same like that, but incrementing the index per eight/seven. And the retransmission are not regulated by a period of time, it is regulated by the BlkACK variable, which is inside the BlkACK frame.

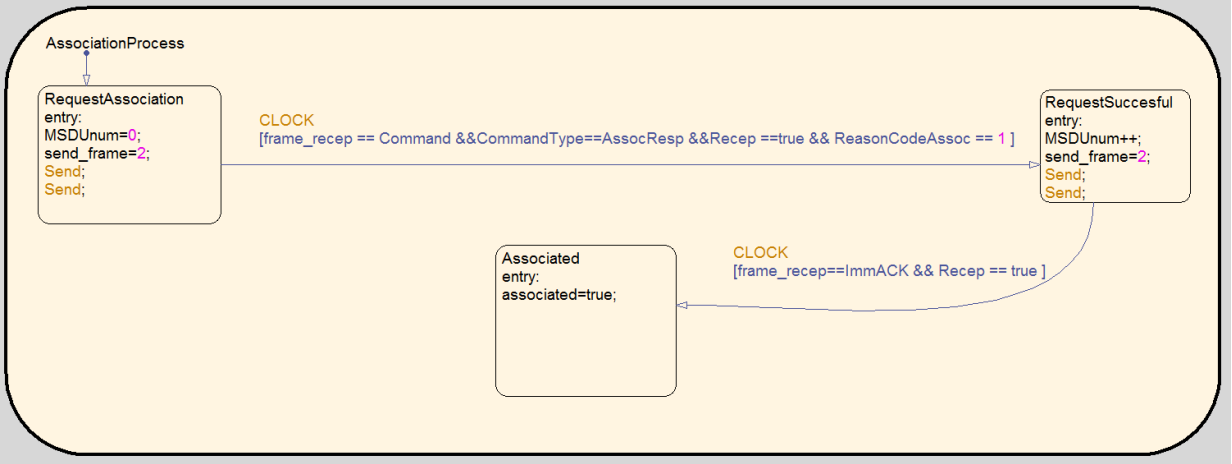
DEV State machines



These are the state machines used by the DEV, when a PNC is discovered, receive beacons, it should start the Association process, when the DEV is associated it should ask for CTAs, if CTAs are available and assigned, then it can start to receive the data. All these processes are explained in the section *3*. When it finishes and the PNC disassociates the DEV the simulation is stopped

On the next pages, there are explained these processes that are used by the DEV.

This is the substates of the state AssociationProcess, it is used by the DEV to request association:



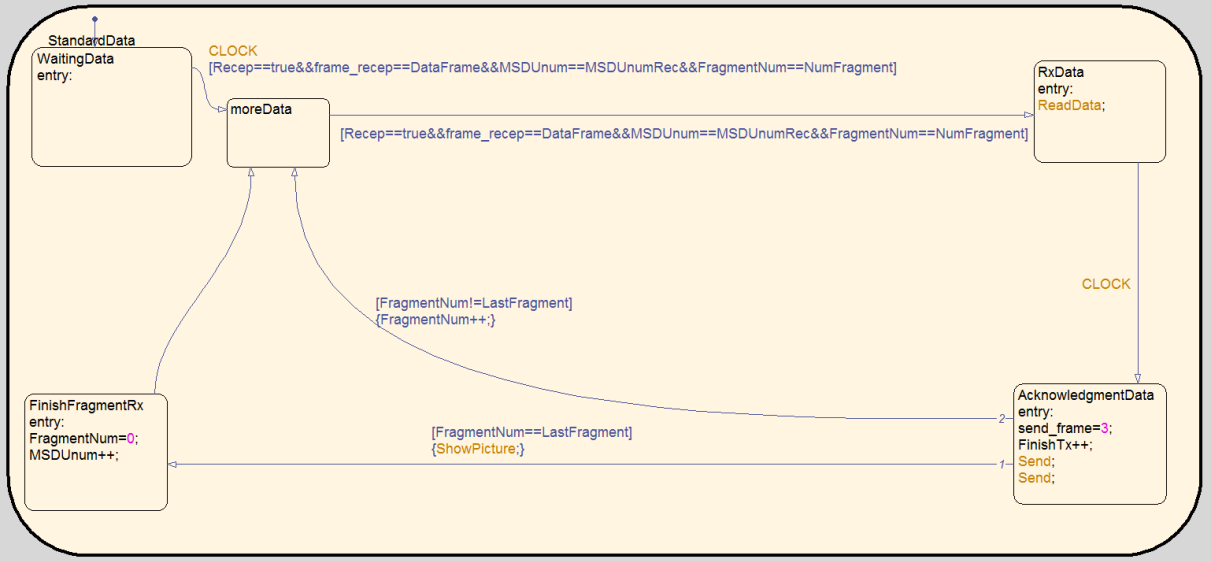
In this case, the first state of the DEV is RequestAssociation, this state is used to send an Association Request frame to the PNC, selecting the frame with send_frame and enabling the Sender block as well as the PNC makes, and wait for and answer of it. When an answer is received it should check the ReasonCodeAssoc. When the ReasonCodeAssoc is positive, it means that the DEV has a place on the association list of the PNC and the three conditions are accepted, then it pass to the next state RequestSuccessful, here it response with another Association Request in order to provide with all the rest information that the PNC need.

Finally, it will change to the last state if it receives an ImmACK and the Association is successful.

The second process is the CTARequest, in this state is just asking for a CTA available and waits for the response:

The DEV is sending the frame CTArequest selecting it and enabling the Sender block, then it waits until arrives a CTAResp, it should read the ReasonCodeCTA, if it is positive the process finishes and the variable CTAassign becomes true.

The last state is ReceiveData, this state is used to receive the data frames, but depends on the SendMode, as well as the PNC state. That de receiving method depends in which mode is working. Between Standard mode, SC/HSI mode or AV mode. Here is it a screenshot of the receiving StandardMode:



Since has been assigned CTAs to the DEV, it can start to share data with the PNC.

The DEV waits on the state WaitingData until a Data frame arrives. When a Data fragment is received and its number of fragment corresponds to the fragment that the DEV is waiting for, it has to start to save those with the event ReadData that goes directly to the Receptor block, in order to reorder the fragments when all of them have been received. Then it comes the state Acknowledge data, in this state its sends an ImmACK frame with the number of fragment and MSDU previous received on the last Data frame, in consideration of making known to the PNC that the Data was well received. And it goes to the state moreData, incrementing the FragmentNum, waiting for the next fragment.

If a Data fragment is received and it corresponds with Last Fragment of the MSDU, it means that the MSDU was completely received and then it can save the MSDU completely with the event ShowPicture and draw it on the screen. After that it continues receiving data but initializing the count fragment, in order to receive other completely MSDU.

The other 2 aggregated modes are the same but using the negative acknowledgment, if a data frame receive has the BlkACK positive, it means that the CRC checker has detect errors, then it should acknowledge the frame with a negative acknowledgment.

Test data simulation

In order to test the simulator are used a series of pictures drawn with bits, acting together consecutively like a video, that is the easiest mode to test it without entering on multimedia concepts. Each picture is one MSDU.

The pictures should be imported to Matlab and reshaped to a matrix of (TotalLength x 1), on the script "General.m" should be specified the dimensions of the picture in order to reconstruct them, when the pictures are reshaped, they should be coupled all together, on the way that finally we have a variable with (TotalLength x MSDUnumber), then this data should be called TestData. The script "General.m" is the manager to adapt this data depend which mode of transmission is going to be used, it should add Padding in order to make a number exact of fragments in order to work with fixed size signals on Simulink.

In this case, the simulation has implemented a default movie of a snail that is moving along the picture, here we have an example: