the PNC does not send the PS Set Information Response command. Likewise, if there are no current isochronous CTRqBs, then the current PNC will not send the PNC Handover Information command. The new PNC knows that the current PNC will not be sending these commands because the number of CTRqBs or PS Sets in the PNC Handover Request command will be zero. Of course, it is not possible for there to be zero DEVs associated in the piconet. In the case of PNC hand over, there must be at least two DEVs, the current PNC and the DEV that will become the new PNC. Thus, the PNC Information command will always be sent in a PNC hand over.

Once the new PNC has received all of the information from the current PNC and it is ready to take over as the PNC of the new piconet, it sends the PNC Handover Response command. After the current PNC ACKs this command, it puts the PNC Handover IE in the beacon with the beacon number of the first beacon that will be sent by the new PNC. The last beacon that will be sent by the old PNC is the one that has a beacon number one less than the value of the Handover Beacon Number field. Once the PNC has ACKed the PNC Handover Response command, it must stop sending beacons at the time indicated for the new PNC to start sending its beacons. If for some reason the new PNC does not send beacons at the correct time, the old PNC is not allowed to begin sending beacons again. The reason for this is that it is possible that the old PNC is, for some reason, unable to hear the beacons being sent by the new PNC. If it begins to send beacons as well, it could either collide with the new PNC or confuse DEVs that are members of the piconet.

The new PNC is allowed to shutdown the piconet at any point in the hand-over process up until it places the PNC Handover IE in the beacon. The PNC might do this if it has been give a fixed length of time to shut itself down and the hand-over process is taking too long. In that case, it would change the process and simply shutdown the piconet.

Ideally, the hand-over process should not affect any of the data connections in the piconet. The new PNC is required to begin sending its beacon at the same time as when the old PNC was to have sent its beacon. Because the CTRqBs have been transferred to the new PNC, it is also able to continue to allocate channel time for all of the existing streams. However, the new PNC may not be able to support the same number of DEVs as the old PNC was able to support. In this case, the new

INTEL-1010
U.S. Patent 9,265,043

PNC will disassociate DEVs from the piconet until it has reduced them to a number that it can handle. Likewise, if the new PNC cannot support all of the CTRqBs that were handed over by the old PNC, it will terminate the streams of all of the allocations that it is unable to serve.The new PNC decides which DEVs are disassociated and which streams are terminated because it is the one that has best information regarding the streams and DEVs that it wants to keep in the piconet. The old PNC would have to randomly disassociate DEVs or terminate streams because it could not possibly know which ones the new PNC would like to keep.

The PNC is allowed to hand over control of the piconet to a DEV that is currently operating as a dependent PNC. Because the standard does not specify a method to merge (or to split) piconets, the implication is that the DEV that is the dependent PNC would be running two distinct piconets. Simultaneously running two piconets could well exceed the capabilities of a PNC-capable DEV. Thus, a DEV that is a dependent PNC can refuse the PNC hand over. A DEV refuses the PNC hand-over process by sending the PNC Handover Response command to the PNC with the Reason Code field set to "Handover refused, unable to act as PNC for more than one piconet."



The standard does not specify the method that a new PNC will use if it needs to reduce the number of DEVs or CTRqBs that it is retaining once the control of the piconet is handed over. This is an implementation-specific feature that can be used by an implementer to differentiate the quality of their MAC implementation. A correct answer to this problem is non-trivial.

Ending a piconet

If the PNC is shutting down and it does not have another PNC-capable DEV to which it can hand over responsibility or if it does not have time to complete a hand-over process, the PNC begins the shutdown procedure. The PNC cannot let all of the dependent piconets continue operation. If there are two dependent piconets in the current piconet, they would both desire to access to the rest of the channel time and yet they cannot both operate at the same time without the coordination of the old PNC. Because the PNC cannot hope to make an intelligent decision about which dependent piconet to leave control, it simply picks the dependent piconet with the lowest DEVID to remain operating in the channel. Because neighbor DEVIDs are larger than all of the assigned DEVIDs, the neighbor PNC would be selected only if there were not any child PNCs in the piconet.

INTEL-1010
U.S. Patent 9,265,043

The PNC informs the chosen dependent PNC as well as other DEVs in the piconet via the PNC Shutdown IE in the beacon. This IE contains the DEVID of the dependent PNC that will be allowed to continue operation in the channel. All of the other dependent PNCs are required to start their own shutdown process as well. Once the parent PNC has completed its shutdown procedure, the selected dependent PNC is allowed to use all of the time in the channel for its own piconet.



There was some discussion in the Task Group as to the best method for determining which dependent piconet would "survive" the shutdown of the parent piconet (once it was determined that collisions could happen). The reality is that any decision is somewhat arbitrary and so the group consciously chose an arbitrary criteria, the DEVID of the dependent PNC. There really is no deep logic behind this decision except that it does have the nice result that it automatically puts neighbor PNCs at the end of the list, which is likely a desirable outcome.

If the PNC was shutting down because it did not have time to hand over and there are no dependent piconets, one of the remaining PNC-capable DEVs in the piconet is allowed to start a piconet in the same channel using the piconet start procedure. However, if there is a remaining dependent piconet, then that DEV would have to either find another channel to start a new piconet or join the remaining piconet and try to start its own dependent piconet. Other DEVs may also try to join the remaining piconet, or they may search the other PHY channels to find another piconet to join.

JOINING AND LEAVING THE PICONET

The PNC controls membership in the piconet using association, disassociation, and secure membership. The method that the PNC uses to determine which DEVs are authorized to join the piconet with secure membership is outside of the scope of the standard. Authorization will normally involve the intervention of the user to verify that a new DEV will be allowed to join the piconet and potentially may involve cryptographic means to identify or authenticate the new DEV.

INTEL-1010
U.S. Patent 9,265,043

Association

The association process accomplishes two things. The first is to assign a DEVID to the DEV. Although every DEV has a unique DEV address, the 8-octet address is too long for the frame header. Thus, the PNC assigns a shorter, 1-octet DEVID to the new DEV in the association process. The second thing the association process accomplishes is to limit the requirements on the current PNC. For each DEV that joins the piconet, the PNC is required to store some information and to maintain a timer that keeps track of its activity. Because the resources of the PNC are limited, it is important that the PNC is able to refuse to admit new DEVs once it has reached its limit. The PNC may also uses the association process to refuse to allow neighbor piconets in the current piconet. A DEV that wishes to join as a neighbor must state its intention to do so when it associates. The PNC may decide at this point to refuse to allow the DEV to join as a neighbor.



The DEV address used in 802.15.3 is based on the 8-octet MAC address assigned by the IEEE registration authority committee (RAC). Up until late in the sponsor ballot process, the 802.15.3 draft used the older 6-octet MAC address as the DEV address (802.11, for example, uses this type of MAC address). Because 802.15.3 was a new standard and it did not require MAC level routing of frames, the Task Group chose to use the new, longer MAC address offered by the IEEE. Companies and organizations can request groups of addresses from the IEEE RAC, who ensures that these numbers are unique among all network adapters.

Because a DEV that wishes to associate has not been assigned a DEVID, it uses the unassociated ID (UnassocID) for all of its communications until it has been assigned a DEVID by the PNC. All of the associating DEVs share a single DEVID, and so they must use a contention-based access method to send the Association Request command to the PNC. In the 2.4 GHz PHY, the CAP is always available for sending Association Request commands. In addition, the PNC may allocate association MCTAs in the CTAP to allow DEVs to send Association Request commands to the PNC. The complete frame exchange sequence for association is illustrated in Figure 4–6.

INTEL-1010
U.S. Patent 9,265,043

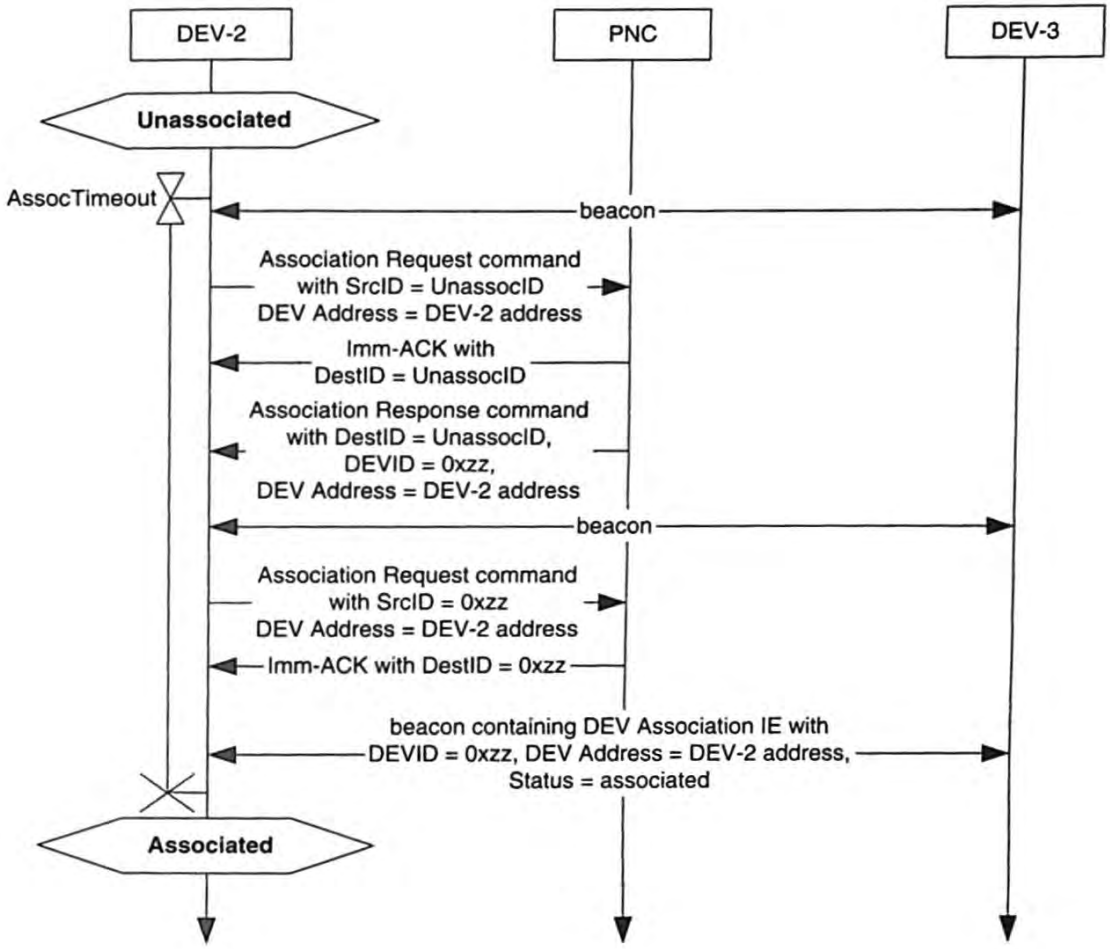


Figure 4–6: Frame exchange a DEV associating with a PNC

The Association Request command contains the DEV address of the new DEV as well as its Overall Capabilities field, and its requested association timeout period (ATP). If the PNC receives this command successfully, it sends an Imm-ACK with the UnassocID as the DestID. The PNC then responds to the Association Request command with an Association Response command that either accepts or rejects the request for association. Because the DEV has not yet learned its new DEVID, the Association Response command is sent with the UnassocID as the DestID and the ACK policy is set to no-ACK. This prevents multiple DEVs from attempting to ACK the frame at the same time. If the PNC accepts the association,

INTEL-1010
U.S. Patent 9,265,043

it sets the DEVID to the newly assigned ID, responds with the ATP that it is willing to accept, and includes the DEV address of the DEV to which it is responding. The DEVs that receive this command look for their DEV address to determine if the PNC is responding to their request. Additionally, the PNC is able to include a Vendor Specific IE in the response. This IE might contain information regarding the authentication method that is being used by the PNC for secure membership.

To acknowledge that it received this frame, the DEV resends the Association Request command, this time with its new DEVID as the SrcID for the frame. If the PNC successfully receives this frame, it sends an Imm-ACK to the newly associated DEV and places the DEV Association IE in the beacon. This IE informs the other DEVs in the piconet that a new DEV has joined the piconet. However, if security is enabled in the piconet, the other DEVs need to wait until the new DEV completes the secure membership process before they can begin communications with the new DEV. The DEVs in the piconet will be informed that the new DEV has become a member of the piconet when the PNC broadcasts the PNC Information command with the new DEVID as an entry and the Membership Status bit set to indicate secure membership.

If the PNC is rejecting the request, it responds with an Association Response command with the DEVID field set to the UnassocID and the DEV address field set to the address of the DEV for which it is rejecting association. The PNC also includes a Reason Code that explains why the PNC is not going to allow the DEV to associate. Reasons why the PNC may refuse association are as follows:

- The PNC is already serving its maximum number of DEVs.
- There is not enough channel time available to serve the DEV.
- There is too much interference in the channel.
- The PNC is turning off with no PNC-capable DEV in the piconet to hand over responsibilities.
- The PNC will not allow the formation of neighbor piconets.[18]
- The PNC is in the process of changing the PHY channel.

[18] This reason is only valid if the DEV has indicated in its capability field that it is associating as a neighbor PNC.

INTEL-1010
U.S. Patent 9,265,043

- The PNC is in the middle of a hand-over process.
- The PNC is denying association for some unspecified reason.

The last reason is a catch-all because the PNC can reject an association request for just about any reason. Depending on the reason why the association was refused, the DEV may want to try again to reassociate. For example, in the case of channel change or hand-over, the DEV can watch the beacon to see when the process has finished and attempt to associate at that time.

Broadcasting piconet information

The PNC broadcasts the piconet information using the PNC Information command after a DEV becomes a member of the piconet. This means that if secure membership is required for the piconet, the PNC will wait until after the secure membership process is complete to broadcast the piconet information with the new DEV. In addition, the PNC is required to send the piconet information for each of the DEVs that are members of the piconet at least once every mBroadcastDEVInfoDuration[19] via a PNC Information command. The purpose of broadcasting the PNC Information command is threefold:

a) The first purpose is to inform all of the members of the piconet that a new DEV has joined. Note that the DEV Association IE is sent when the DEV finishes the association process and it only has a portion of the DEV's capability field. In addition, if security is required for the piconet, the other DEVs need to wait until the new DEV completes the secure membership process with the PNC before they can communicate with the new DEV. The PNC Information command will only contain the new DEVs information once it has successfully become a member of the piconet (i.e., it has been given a management key for the PNC; see "Security policies" on page 193).

b) The second purpose of the broadcast PNC Information command is that it provides the DEVIDs, DEV Addresses and capabilities fields for the other members of the piconet. Thus, once a DEV joins an 802.15.3 piconet, it will immediately find out the information required to begin communications with other DEVs in the piconet.

[19] mBroadcastDEVInfoDuration is a constant defined at the end of Clause 8 in the standard with a value of 64*mMaxSuperframeDuration or just a little under 4.2 s.

INTEL-1010
U.S. Patent 9,265,043

c) The final purpose of the broadcast PNC Information command is that it provides a method by which DEVs in the piconet can update their internal list of the members of the piconet. For example, if a DEV is in a power-save mode, it might miss the association and disassociation announcements that occur in the beacon. Once the DEV receives the broadcast PNC Information command, it can quickly update its internal memory with information about the current members of the piconet.

Disassociation

A DEV ends its relationship with the PNC and the piconet via the disassociation process. The frame exchange sequence for a DEV initiating the disassociation process is illustrated in Figure 4–7, whereas the process of the PNC disassociating a DEV is illustrated in Figure 4–8. If the DEV wishes to disassociate, it sends the Disassociation Request command to the PNC. When the PNC acknowledges the receipt of this command, the DEV considers itself disassociated. Likewise, if the PNC wants to disassociate a DEV, it sends the Disassociation Request command with the appropriate Reason Code to the DEV. When the DEV ACKs this command, the PNC will consider the DEV to be disassociated.

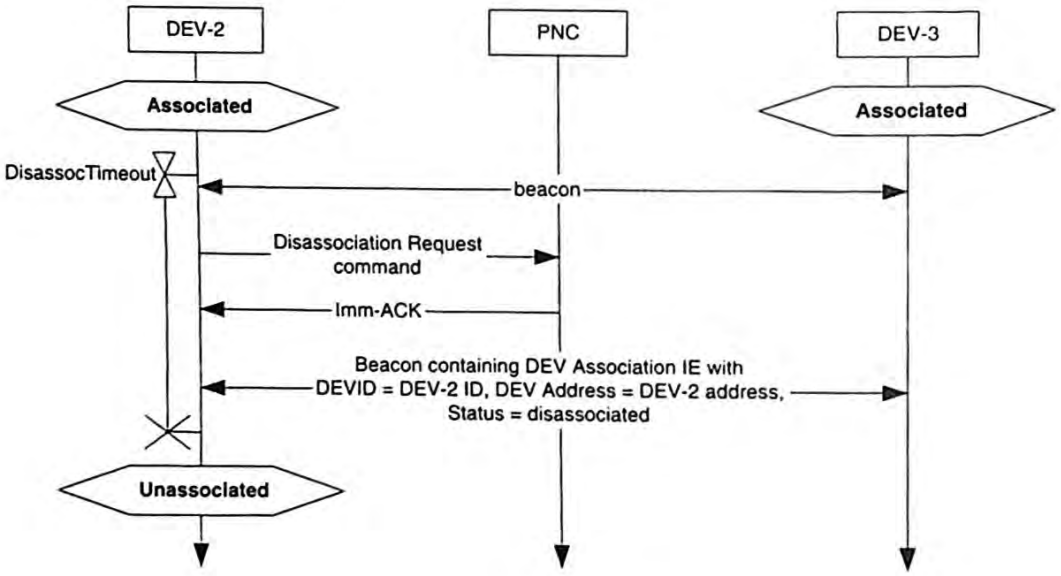


Figure 4–7: Frame exchange for a DEV dissociating

INTEL-1010
U.S. Patent 9,265,043

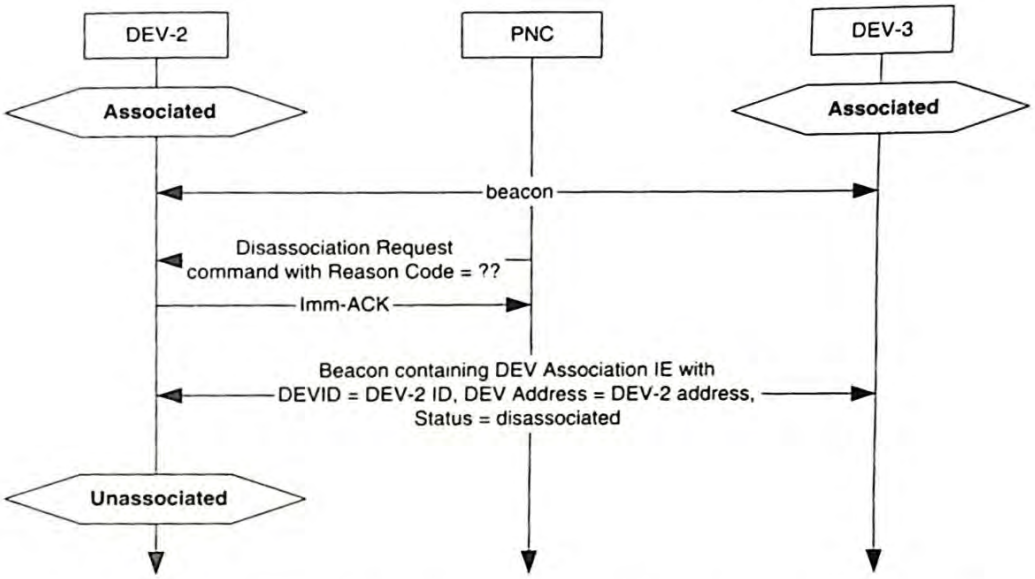


Figure 4–8: Frame exchange for the PNC dissociating a DEV

Valid reason codes for the Disassociation Request command are as follows:

- The DEV's ATP has expired.
- The PNC is having problems with the channel and is unable to continue serving the DEV.
- The PNC is unable to service the DEV.
- The PNC is turning off with no PNC-capable DEV in the piconet.
- The DEV is leaving the piconet.



The PNC may disassociate a DEV because it cannot support the number of DEVs that are currently associated with the PNC ("PNC unable to service DEV"). On the other hand, the PNC may be able to handle the DEV as a a member of the piconet but it would not be able to allocate any additional channel time to the DEV. In this case, the PNC would disassociate the DEV with the reason code "Channel too severe to service the DEV."

The first four reason codes are only valid when the command is sent by the PNC, whereas the last reason code is only valid when the DEV is sending the command. The Disassociation command is also used to stop a neighbor piconet. However, the PNC is required to perform some additional steps for this process, as described in "Parent PNC ceasing operations with dependent piconets" on page 174.

INTEL-1010
U.S. Patent 9,265,043

Because it is possible or even likely that DEVs will move out of range of the PNC without the PNC being aware of it, the standard requires that a timeout interval be kept for every DEV in the piconet. This interval, the ATP, is requested by the DEV during association. However, the PNC has the final say on the length of time allowed for the ATP. Because the ATP is a 2-octet value in milliseconds, the longest possible ATP is approximately 65 s. If the PNC has not received a frame from the DEV within the ATP duration, it considers the DEV disassociated from the piconet. In order to assure that the PNC has received a frame, the DEV should send at least one frame to the PNC within the ATP with the ACK policy set to be Imm-ACK. If a DEV does not have information that it needs to send to the PNC, it can send a Probe Request command that does not request any IEs to the PNC with the ACK Policy field set to Imm-ACK to renew its ATP timer.

Likewise, if the DEV has not heard from the PNC for an ATP duration, i.e., it has missed quite a few beacons, it considers itself disassociated from the piconet and stops using the DEVID that was assigned to it by the PNC. The DEV may look for another piconet to join, or if it is capable, it could start its own piconet as the PNC.

Assigning DEVIDs

The PNC is required to assign DEVIDs in increasing order, except when it wishes to reuse the DEVID of a DEV that has left the piconet. Thus, when a DEV starts a piconet, it takes on the DEVID 0x00[20] as the PNC and the DEVID 0x01 for its DEV personality. Because the DEVIDs are reused, the PNC must make sure that the DEV that previously had the DEVID has been away long enough that it knows that it has been disassociated from the piconet (see "Disassociation" on page 78). To be sure, the PNC needs to wait until at least two times the ATP duration for that DEV has passed.

The DEVIDs are assigned as follows:

- *0x00:* The PNC (PNCID)
- *0x01 to 0xEC:* IDs for regular members of the piconet (DEVIDs)
- *0xED to 0xF6:* Reserved IDs for future use

[20] This book and the 802.15.3 standard use the notation 0x to indicate that the numbers that follow are in hexadecimal notation.

INTEL-1010
U.S. Patent 9,265,043

- *0xF7 to 0xFC:* IDs for the PNCs of neighbor piconets (NbrID)
- *0xFD:* The multicast ID (McstID)
- *0xFE:* The ID for unassociated DEVs (UnassocID)
- *0xFF:* The broadcast ID (BcstID)

MANAGING BANDWIDTH

Once the piconet has been started and a DEV has joined, the next task is to open communications with the other DEVs in the piconet. The method that the DEV uses to do this depends on the type of data and the length of the CAP in the superframe. The DEV also needs to determine which ACK policy it will use in sending the data because this affects when the data can be sent and the amount of time that may need to be allocated for sending the data. If the DEV needs to request channel time for the data, it will choose either an isochronous or asynchronous request method.

Acknowledgements

IEEE Std 802.15.3 provides four acknowledgment (ACK) policies that can be used for data. The ACK policy is a field in the MAC header that tells the receiving DEV what type of ACK the source DEV is requesting. The four types of ACK policies are:

a) No acknowledgment (no-ACK)

b) Immediate acknowledgment (Imm-ACK)

c) Delayed acknowledgment (Dly-ACK)

d) Delayed acknowledgement request (Dly-ACK request)



The ACK policy field is 2 bits and it was arranged so that each bit has a different meaning. The second bit in the field indicates if a response is required. If the second bit is set to zero, then the ACK policy is either no-ACK or Dly-ACK and the receiver will not respond to the frame. On the other hand, if the second bit is set to one, then the destination is supposed to respond, either with an Imm-ACK frame or Dly-ACK frame, if it correctly receives the frame. This allows the MAC implementation to check only a single bit to determine if it will need to switch from receive to transmit mode following the reception of the current frame.

The no-ACK policy is used when there are multiple DEVs that may wish to ACK the frame, e.g., for the Association Response command, for broadcast frames, and

INTEL-1010
U.S. Patent 9,265,043

for multicast frames. In addition, the data may not require an ACK, especially if it is time-sensitive data like streaming video, audio, or two-way voice traffic. In that case, a retransmission may arrive too late and the upper layers will handle dropped frames. In other instances, the ACKs will be handled by a higher layer protocol. For example, the delayed ACK mechanism in this standard could also be implemented at a higher layer by using no-ACK for the data with either a short CTA in the reverse direction or even a sub-rate slot in which the destination DEV reports the frames that it has received.

The Imm-ACK policy is used when the DEV needs to confirm that the destination has received the frame and may want to attempt a retransmission if it fails. The Imm-ACK policy is used for most of the command frames that are sent to a single DEV because the protocol state machines rely on knowing if the message was delivered. The Imm-ACK policy is also appropriate for typical data transfers where even a single missing octet will render a file transfer useless. Because the Dly-ACK policy cannot be used during the CAP or during an asynchronous CTA, a DEV would need to use the Imm-ACK policy during these times if it needs to verify that the destination DEV correctly received the information. When a DEV receives a frame with the Imm-ACK policy set, it verifies that the calculations for the header check sequence (HCS) and frame check sequence (FCS) match the data in the frame. If both checks pass, the DEV sends the Imm-ACK frame one SIFS following the frame that requested the Imm-ACK. Note that it is the responsibility of the sending DEV to make sure that there is enough time remaining in either the CAP or CTA for the destination to send the Imm-ACK frame.



The term MSDU refers to data that is passed between the MAC and higher protocol layers. A MAC protocol data unit (MPDU), on the other hand, refers to data units that are passed between the MAC and the PHY. A more in-depth discussion of the differences between an MSDU and an MPDU is given in "Fragmentation/defragmentation" on page 96.

The Imm-ACK frame indicates that the destination received a frame for which it was the DestID and that the HCS and FCS were valid. The destination may still discard the frame depending on other factors. For example, if the Integrity Code check or a secure frame fails (see "Overview of AES CCM" on page 195), the DEV may discard the frame. The DEV may also discard the frame if its internal buffers are full. If the DEV receives only some of the fragments of a MAC service data unit (MSDU), it will ACK the fragments it receives, but it will discard them if it does not receive all of the

INTEL-1010
U.S. Patent 9,265,043

fragments. Thus, the higher layer protocols need to be aware that the ACK alone will not guarantee that the entire frame was passed up from the MAC.

The third type of ACK policy is the Dly-ACK policy. This policy is used in conjunction with the Dly-ACK request policy to set up and operate an ACK method in which the destination DEV stores ACK information until it is requested by the source. The reason for including Dly-ACK in the standard is that the Imm-ACK method is relatively inefficient in sending data due to the fixed overhead of the two SIFS and Imm-ACK frame required to send what is really one bit of information (i.e., "Did the frame get through?"). The inefficiencies are particularly pronounced for higher data rates, e.g., for data rates faster than 44 Mb/s.



The capability provided by Dly-ACK is sometimes referred to as a group ACK or burst ACK in other standards or specifications. The naming of the Dly-ACK functionality is arbitrary and could have been called Grp-ACK (group ACK) instead. However, the first versions of the draft used delayed ACK as the name and the task group decided to stick with this name for the final draft. The abbreviation did change; the delayed ACK policy was initially abbreviated as Del-ACK.

The improvement in throughput from using Dly-ACK is shown in Table 4–3. Th 110 Mb/s and 480 Mb/s data rates are not defined for the current PHY, but they are included in the table to provide a comparison for what might happen with the data rates proposed for 802.15.3a.

Table 4–3: Throughput improvement using Dly-ACK

PHY data rate	Rate with Imm-ACK	Rate with Dly-ACK	Improvement
11 Mb/s	10.4 Mb/s	10.7 Mb/s	2.4%
22 Mb/s	20.1 Mb/s	21.0 Mb/s	3.8%
33 Mb/s	29.1 Mb/s	30.7 Mb/s	5.6%
44 Mb/s	37.4 Mb/s	40.1 Mb/s	7.2%
55 Mb/s	45.0 Mb/s	48.9 Mb/s	8.8%
110 Mb/s	84.8 Mb/s	94.9 Mb/s	11.4%
480 Mb/s	209 Mb/s	280 Mb/s	33.9%

INTEL-1010
U.S. Patent 9,265,043

In the calculations for Table 4–3, the frame size that was used is pMaxFrameBodySize and the number of frames in a burst is 10. The PHY dependent parameter pMaxFrameBodySize parameter is the limit on the size of a frame that can be sent with a given PHY. For the 2.4 GHz PHY, this is set to 2048 octets (which includes the FCS, but not the PHY preamble, PHY header, MAC header, or HCS). Note that the rates in the table are only the instantaneous delivered data rates and do not include the overhead due to the beacon, CAP, guard times, or the presence of other DEVs in the piconet. The Dly-ACK mechanism is not particularly compelling for the 11 Mb/s data rate, but it becomes more interesting at the higher data rates. At the 55 Mb/s data rate, using Dly-ACK provides an extra 4 Mb/s of instantaneous data rate. The improvement is greater for shorter frames because a larger percentage of the time is spent sending the headers and waiting for the PHY to switch from transmit to receive and back again.



Constants used in the standard are prefixed either with a lower-case "m" for MAC constants or a lowercase "p" for PHY constants. The MAC constants are defined in a table in subclause 8.15 of the standard. The PHY constants are also summarized in a table in this subclause, but the actual definitions are found in the 2.4 GHz PHY definition in clause 11. In PDF copies of the draft, each location of these constants in the text is a hyperlink to the definition so that clicking on the constant should cause the PDF display to jump to the actual definition.

Because the Dly-ACK policy places a burden on the finite resources of the destination DEV, the source DEV is required to negotiate the parameters of the Dly-ACK burst before it can begin using the Dly-ACK method. A burst is the collection of the frames that are pending acknowledgment via a Dly-ACK frame. Because of this, the burst may be spread across multiple CTAs or even across multiple superframes. The Max Frames field in the Dly-ACK frame is the number of frames that the destination is willing to receive in a single burst, regardless of size. In addition, the destination needs to specify the maximum amount of data, in octets, that it can receive in a single burst. This number is communicated in the Max Burst field, which is the number of pMaxFrameBodySize frames that it can receive in a single burst.

The Max Frames field allows the destination to limit the number of "handles" or "bins" that it uses to store the incoming frames. The DEV may have a limit to the number of memory addresses that are assigned for incoming frames. The Max Burst field allows the destination to limit the impact of Dly-ACK based on the

INTEL-1010
U.S. Patent 9,265,043

overall size of its buffer. These two parameters together give the destination control over the amount of data it needs to store to participate in the Dly-ACK mechanism.

The source DEV begins the Dly-ACK negotiation by sending a frame with ACK policy set to Dly-ACK request. If the destination does not want to support the Dly-ACK method, it responds with an Imm-ACK frame, as shown in Figure 4–9. If it is willing to use the Dly-ACK method, it responds with a Dly-ACK frame that contains the Max Burst and Max Frames that it is willing to support, as shown in Figure 4–10.

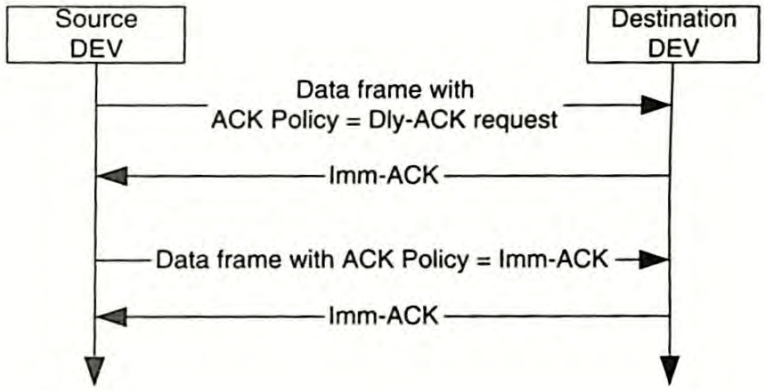


Figure 4–9: Frame exchange for an unsuccessful Dly-ACK negotiation

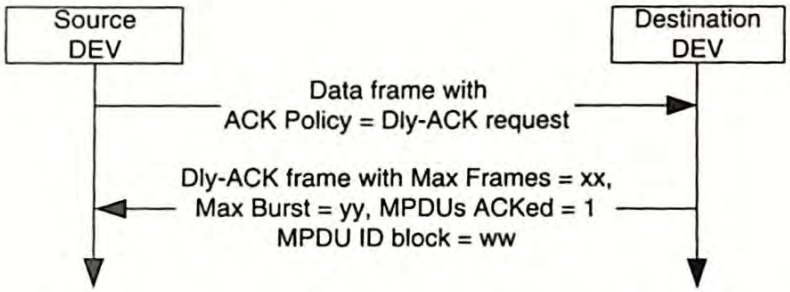


Figure 4–10: Frame exchange for a successful Dly-ACK negotiation

If the destination accepts the request for the use of Dly-ACK, the source is then able to begin sending frames to the destination with the ACK policy set to Dly-ACK. The source DEV can send up to either the Max Burst or the number of

INTEL-1010
U.S. Patent 9,265,043

Max Frames to the destination. The source then requests the ACK information by sending a frame with the ACK Policy set to Dly-ACK request. The destination responds with the MAC service data unit (MSDU) number and fragment number of all of the frames that it successfully received. The Dly-ACK frame also contains a new Max Burst and Max Frames values that the source destination may use to continue the Dly-ACK process.

This process is illustrated in Figure 4–11. The source first requests the use of Dly-ACK and is given a positive response that allows up to two frames to be sent. In the second data frame that is sent, the source sets the ACK Policy to Dly-ACK request and receives a response from the destination that it received MSDU number 3 but did not receive MSDU number 2. In the figure, the MSDUs are not fragmented and only the MSDU number and not the fragment number is listed for clarity. Following the reception of the Dly-ACK frame, the source resends MSDU number 2 as well as MSDUs numbered 4 and 5 and requests the Dly-ACK frame with the frame that contains MSDU number 5. The destination responds with a Dly-ACK frame that acknowledges all three MSDUs.

If the destination is participating in a Dly-ACK process with one DEV and it receives a request from another DEV, it can determine from its internal resources if it is able to support the other DEV at that time. Because the Dly-ACK process is renegotiated every time a new burst is started, the destination is in control of the amount of resources it has promised to the source DEVs with which it is communicating.

If the destination replies with a Dly-ACK frame with the Max Burst field equal to zero, then the source needs to stop transmitting in the current CTA, but it is allowed to restart the Dly-ACK process in the next CTA by sending a frame with the ACK policy set to Dly-ACK request. If the source does not receive a Dly-ACK frame in response to its request, it will repeat the last frame that it sent until it gets an ACK for the frame. The source DEV may also send an empty data frame with ACK policy set to Dly-ACK request to get the receiver to send the Dly-ACK frame. One reason to do this is that an empty data frame has a higher probability of getting through than a long data frame.

INTEL-1010
U.S. Patent 9,265,043

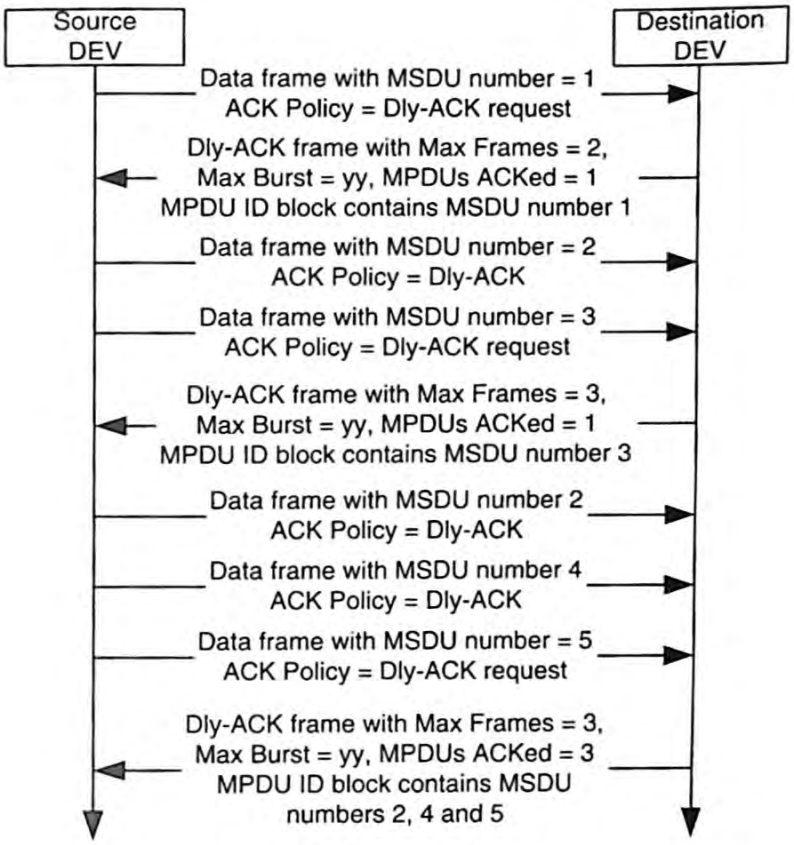


Figure 4–11: Data exchange using Dly-ACK

Asynchronous data

IEEE Std 802.15.3 differentiates between asynchronous and isochronous data connections. An asynchronous data connection is one in which it is OK if there are pauses in the transmission of the data or where a short latency is not required in sending the data. Examples of applications for asynchronous data connection include web surfing, file downloads, backing up data, downloading email, and so on. Although the standard explicitly refers to only one asynchronous allocation method, there are really a total of three ways in which asynchronous data may be communicated in the piconet:

INTEL-1010
U.S. Patent 9,265,043

a) Send the data during the CAP[21]
b) Request channel time for an individual asynchronous allocation
c) Request channel time for a group asynchronous allocation

If the amount of data to be transferred is very small or if the DEV needs to send a command to either the PNC or another DEV, it can use the CAP to send the frame. In the 2.4 GHz PHY, the PNC is required to allow the CAP to be used for data transfer. Using the CAP is a very efficient means for sending a small number of data frames or commands.



One way to understand the difference between an isochronous request and an asynchronous request is that an isochronous request is a request for a data rate, e.g., for a request for 5 Mb/s, whereas an asynchronous request is a request for a specific amount of data to be sent, i.e., a request to send 30 Mb of data.

If the amount of information that the DEV has to send is large enough, if the CAP is too small, or if another PHY allows the PNC to forbid data transfer in the CAP, then the DEV should use the Channel Time Request (CTRq) command to get time to send the data. Because the same CTRq command is used for both asynchronous and isochronous requests, the PNC uses the stream index to differentiate between the two requests. An asynchronous request sets the stream index field to the asynchronous stream index, 0x00. The allocation of asynchronous time is different than that of isochronous time. A isochronous CTRq is requesting that the PNC allocate time in the superframe continually until the source, destination, or PNC request its termination. An asynchronous CTRq, on the other hand, is a request for a total amount of channel time to be allocated when it is convenient for the PNC.

There are two types of asynchronous allocations that the DEV can request from the PNC. The first type is the individual allocation request where each CTRqB corresponds to a request for a CTA to a single DEVID. Using this method, a DEV that needs asynchronous allocations to four different destinations would include four CTRqBs in the CTRq command that it sends to the PNC, one for each of the destinations. The PNC will then allocate the channel time when it becomes

[21] If the PNC indicates in the beacon that data is allowed to be sent in the CAP. For the 2.4 GHz PHY, the PNC is required to allow DEVs to use the CAP for data.

INTEL-1010
U.S. Patent 9,265,043

available with a different CTA for each of the destinations specified in the CTRqBs.

The second type is called a group allocation request. It is a request for a single CTA with multiple destinations. Because the CTA IE does not support multiple destinations, other than the multicast and broadcast addresses, the PNC creates this allocation by placing CTA blocks in the beacon that have the same SrcID, stream index (= asynchronous stream index), start time, and duration, but different DestIDs. Because the SrcID is the same for all of the CTA blocks, there will not be contention for the channel during that CTA. However, all of the DEVs that are listed as destinations will have to listen to the entire CTA in case a frame directed to the DEV is sent.

A DEV is only allowed to use one of the two methods at a time. If the DEV sends a CTRq command with an asynchronous CTRqB where the allocation method is different from its previous requests, the PNC will drop all of the prior asynchronous requests from the DEV. It will then consider the current reques' using the method indicated in the CTRqB.

The frame exchange for negotiating an asynchronous CTA is illustrated in Figure 4–12. The DEV begins the request process by sending a CTRq commanc with an asynchronous CTRqB to the PNC. The PNC does not need to refuse an asynchronous request if it is unable to fulfill the request at this time. Instead, the PNC should retain the request and wait until there is time available in the superframe for the allocation. When the PNC has time available in a superframe for the allocation, the PNC will put the appropriate CTA blocks in the beacon for the asynchronous allocation. Unlike a typical isochronous allocation, the asynchronous request is generally prompted by data arriving at the MAC/LLC interface and not by a specific request from a higher layer to allocate time for communications. Thus, the asynchronous CTRqBs can be thought of as being the current status of the transmit queue for the source DEV.

Because the asynchronous allocation is a transient one, there is not a special process for modifying an existing asynchronous CTRq. Instead, each new request replaces the old one. In that sense, subsequent CTRq commands with asynchronous CTRqBs can be thought of as a modification to the existing allocations.

INTEL-1010
U.S. Patent 9,265,043

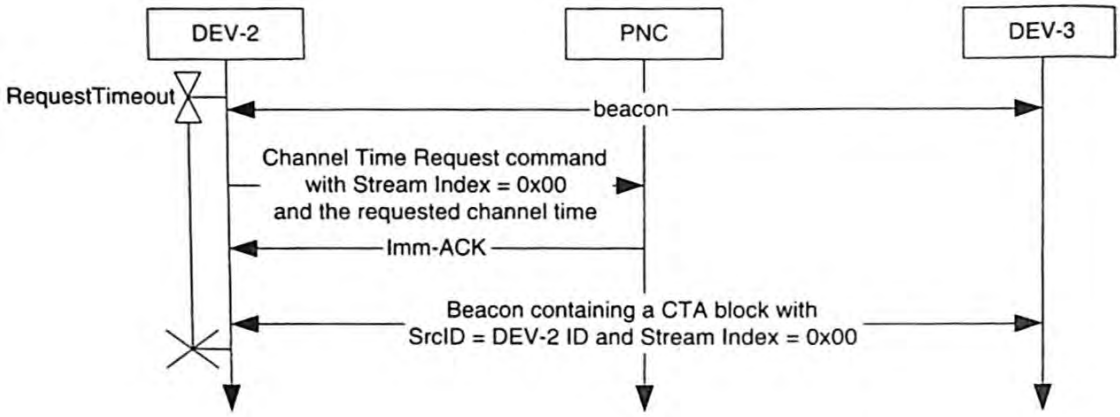


Figure 4–12: Frame exchange for asynchronous CTRq



The CTRq command originally had a Stream Termination bit. However, it was determined that this field was superfluous because the PNC will know that the DEV is requesting the termination of a stream when the DEV sends a CTRq that requests zero channel time. Because it does not make any sense to allocate zero time, this can only be a request to terminate the stream. This is true for both asynchronous and isochronous allocations.

It is possible for either the source or the PNC to terminate an asynchronous allocation. Because asynchronous allocations are transient, the destination is not allowed to request that they are terminated. If the source is sending very short bursts of asynchronous data, then it would be unlikely that either the source or the destination would be able to get the termination command to the PNC before the allocation expires on its own. If the source is sending a large amount of asynchronous data, e.g., a file transfer via FTP, then there may be time to terminate the allocation if the source feels it is necessary. For example, if the higher-layer application sending the data stopped in the middle of the transfer, the source DEV would want to terminate the asynchronous allocation to free up the bandwidth in the network. When the source DEV wants to terminate an asynchronous allocation, it sends a CTRq command to the PNC with an asynchronous CTRqB that has the Target ID List set to the list of destinations whose allocation is being cancelled and the Stream Index and all other fields are set to zero. When the PNC receives this request, it will ACK the command and stop allocating channel time for the asynchronous request. Note that if the original request used the group allocation method, then this termination applies to all of

INTEL-1010
U.S. Patent 9,265,043

the destination DEVs. The frame exchange sequence for the source DEV terminating an asynchronous allocation is illustrated in Figure 4–13.

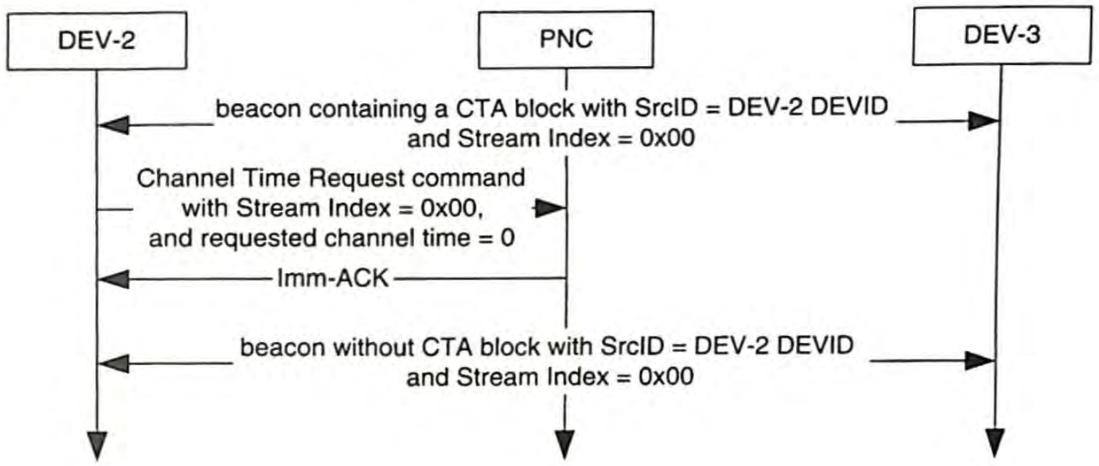


Figure 4–13: Frame exchange for the source DEV terminating

The PNC may also wish to terminate an asynchronous allocation. The PNC's resources may have become overloaded, and so it is unable to handle the asynchronous allocation. If the destination DEV is disassociated from the piconet, either by its own request, by ATP timeout, or because the PNC requests it, the PNC may want to explicitly terminate the allocation and inform the source DEV. When the PNC wants to terminate an asynchronous allocation, it simply stops allocating the channel time and sends a channel time response command to the source DEV. The reason codes that the PNC might send when it terminates an asynchronous allocation include the following:

- Target DEV unassociated
- Target DEV not a member
- Stream terminated by PNC

The PNC would not normally terminate an asynchronous allocation because there is not currently enough channel time. Because it is an asynchronous allocation, it will hold on to the request until it gets a chance to fulfill it. One consequence of saving the pending requests is that the PNC may run out of space to store the CTRqBs. In this case, the PNC would need refuse any new CTRqs until it is able

INTEL-1010
U.S. Patent 9,265,043

to reduce the number of pending requests. The frame exchange for the PNC terminating an asynchronous allocation is illustrated in Figure 4–14.

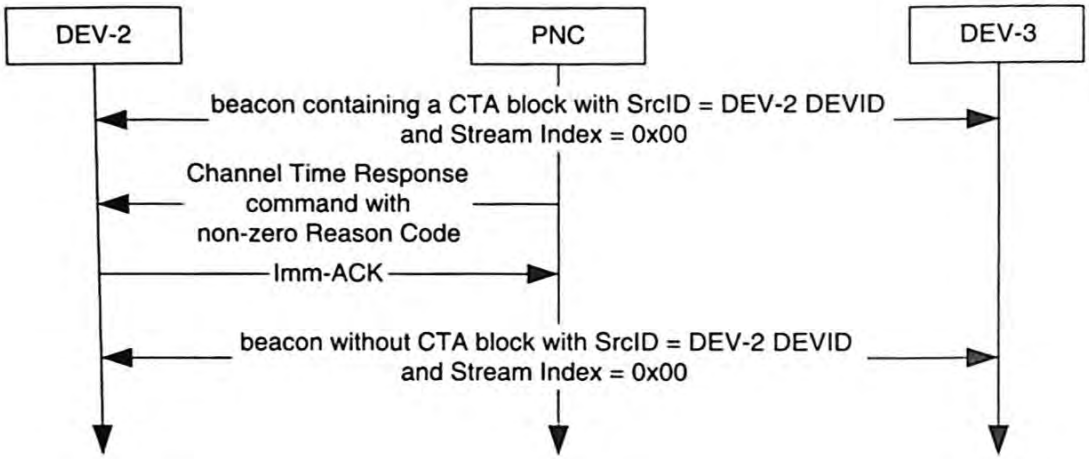


Figure 4–14: Frame exchange for the PNC terminating an asynchronous channel time request

Stream connections

One of the key features of the 802.15.3 protocol is the ability of a DEV to request that the PNC allocate a specific length of time in each superframe or in every n^{th} superframe for the DEV to send frames. Because these allocations occur at regular intervals, they are called isochronous allocations. Isochronous applications need a guaranteed data rate and latency to support the quality of the user experience. A wireless connection, on the other hand, deals only with channel time, frame error rates (FERs), and instantaneous bit rates. One of the key issues in creating connections that support QoS is to map the requested bandwidth and latency into channel time and its repetition, i.e., the latency between allocations.

IEEE Std 802.15.3 initially used a three-way negotiation among the source, the PNC, and the destination. This allocation request included the maximum transmit delay variation, minimum data rate, peak data rate, average data rate, maximum burst size, average frame size, maximum retransmission duration, and receive window size [B16]. The PNC would use these application level requirements to determine the amount and location of the channel time based on the source DEV's requirements, the loading of the superframe, and the superframe duration.

INTEL-1010
U.S. Patent 9,265,043

Although this method works, it requires more complexity in PNC-capable DEVs in order to support complex requests for channel time. The task group felt that it was best if the source DEV simply determined the amount of channel time it needed and requested time from the PNC rather than sending application level requirements to the PNC. In addition, the source DEV has a better understanding of the actual FER between the source and destination than does the PNC. This is particularly true in the case of 802.15.3, where the distance between the source and destination DEVs might be twice as long as the distance from either to the PNC. As the FER increases, the connection requires more time and the PNC would simply be unable to determine this unless it also kept track of lost frames at each of the destinations. Another advantage of this is that the burden of calculating the required channel time is borne by the DEV that needs to use this time.

One drawback of this method is that a really dumb DEV might simply request too much channel time because its algorithm for calculating the required channel time is not very sophisticated. However, there was no simple way to enforce good channel time requests short of specifying the actual algorithm that maps bandwidth, latency, and data buffer sizes directly into channel time and allocation repetition interval. Although there are undoubtedly many good algorithms to accomplish this, it is extremely unlikely that there would be a single one that could get the 75% agreement among the members of the task group that would be required for it to be added to the standard.

The request to create channel time to support the communication of isochronous data is initiated by higher layers in the protocol stack. The method by which the higher layers request this is unspecified in the draft standard. Once the DEV's MAC knows the amount of channel time to request, it follows the process shown in Figure 4–15 to request that time from the PNC. The source DEV begins by sending the CTRq command to the PNC. The PNC will respond to the CTRq command with a Channel Time Response command to either indicate the amount of time that will be allocated or to indicate that the reason why the request was rejected. If the request is accepted, the PNC will allocate the new channel time as soon as possible. If the request is for a sub-rate allocation, a pseudo-static allocation, or if any of the destination DEVs are in a power-save mode, then the PNC will also announce the creation of the allocation using the CTA status IE in the beacon.

INTEL-1010
U.S. Patent 9,265,043

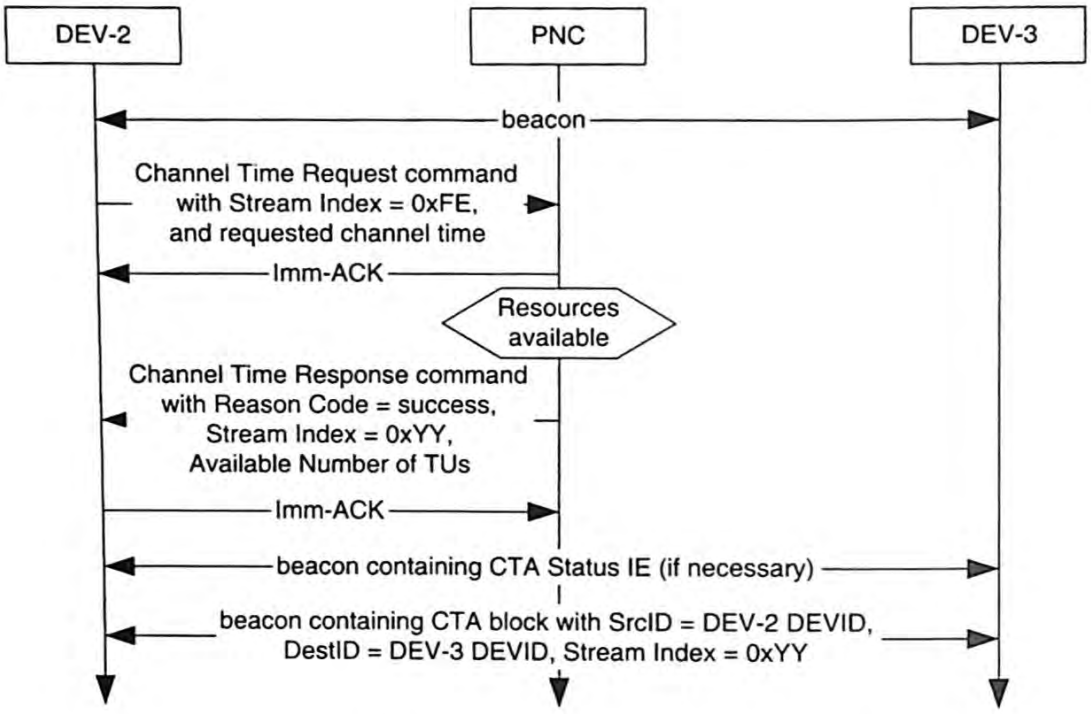


Figure 4–15: Frame exchange for a DEV requesting the creation of isochronous channel time

The channel time request is based on a CTRq time unit which represents the smallest usable amount of time for the connection. The CTRq time unit is normally the length of time it takes to transmit a frame plus the time for the SIFS and the ACK, if necessary. This calculation is illustrated in Figure 4–16 for frames sent using the no-ACK policy. If the DEV wants to use the Imm-ACK policy, then it needs to include the time for the frame, two SIFS duration as well as time for the Imm-ACK frame, as shown in Figure 4–17. In making this calculation, the source DEV needs to know the typical number of bits in a frame and the data rate that can be supported between the two DEVs.



The CTRq TU can take on values from 0 to 65,535 μs, which is exactly the same as the valid range for the superframe duration. However, requesting channel time with the CTRq TU set to the maximum superframe duration will probably cause the request to be rejected.

INTEL-1010
U.S. Patent 9,265,043

The DEV also indicates in the CTRq command the minimum and maximum number of TUs that it needs for the connection. The reason for specifying the requested time in TUs is that it is the most efficient way to allocate the time. For instance, if the frame plus SIFSs and ACK duration was 0.5 ms, then it would not do any good for the PNC to allocate 1.4 ms instead of 1.0 ms because the source DEV can only send two frames during the CTA with either allocation. Although a typical TU calculation would only be for a single frame, the smallest usable time unit for a DEV might be for more than one frame, as illustrated in Figure 4–18.

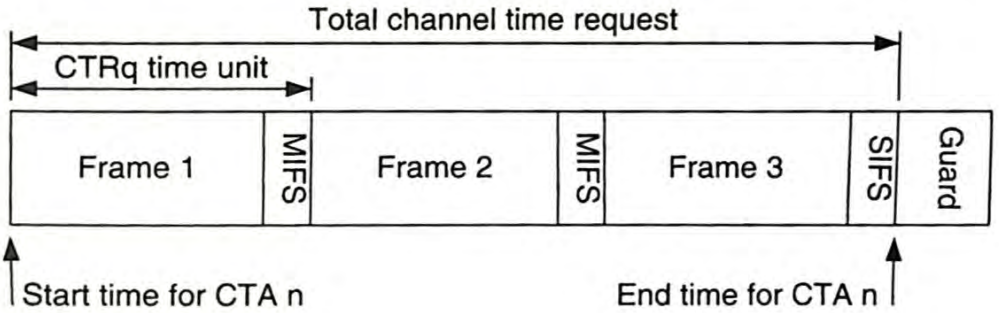


Figure 4–16: Channel time request for frames using no-ACK policy

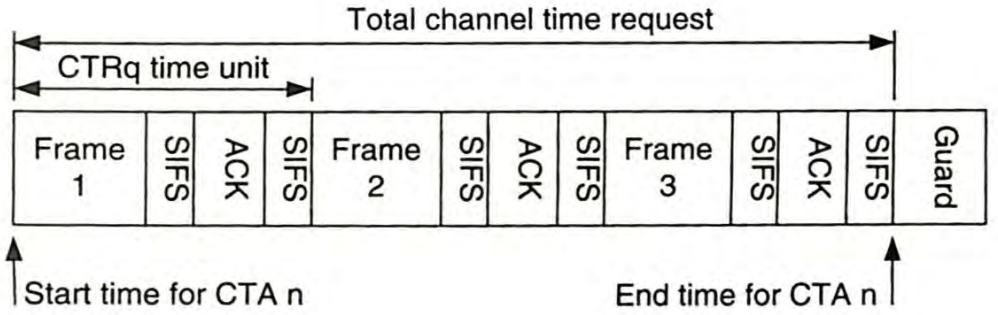


Figure 4–17: Channel time request for frames using Imm-ACK policy

INTEL-1010
U.S. Patent 9,265,043

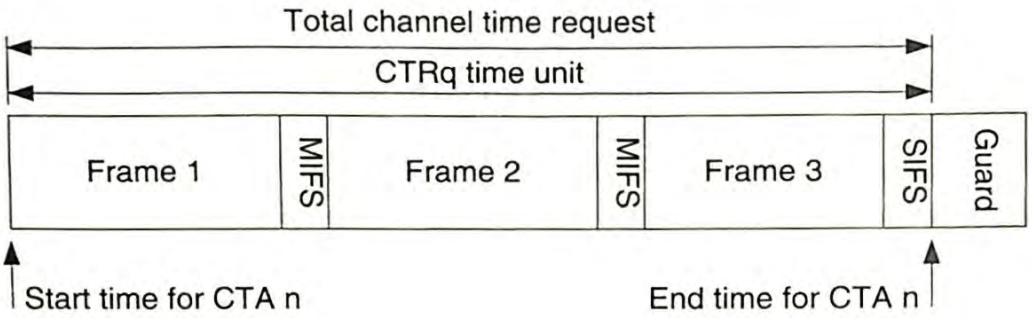


Figure 4–18: Channel time request for multiple frames using no-ACK

Fragmentation/defragmentation

A key part of any networking standard is the ability to split data up for transmission and the reassembly of the data at the receiver. Fragmentation can be done at any layer in the protocol stack. In some instances, the data that is to be sent will be fragmented by two or more of the layers in the protocol stack. IEEE Std 802.15.3 defines a method to fragment frames that would otherwise be too large to send over the PHY. The MAC may fragment a frame for a variety of reasons. The most obvious reason would be that the frame that is passed down from the higher layers is too large to transmit in a single MAC frame. Another, perhaps more subtle reason, is that the MAC may reduce the size of the frames sent over the PHY in order to improve the overall efficiency. Short frames incur a larger overhead as a percentage than do longer frames in a perfect wireless medium. However, as longer frames incur a larger FER for the same bit error ration (BER), the overall throughput of the system may be worse if larger frames are used because more of the big frames will be dropped than would the smaller frames.

In discussing fragmentation, it is important to differentiate between MAC service data units (MSDUs) and MAC protocol data units (MPDUs). The MSDU is the data "blob" that is passed between the MAC and the FCSL. The MPDU, on the other hand, is a data "blob" that is passed between the PHY and the MAC. In the 802.15.3 standard, one MSDU may be fragmented into many MPDUs, but one MPDU can never be associated with more than one MSDU.

INTEL-1010
U.S. Patent 9,265,043

The fragmentation method used in 802.15.3 imposes three main constraints on the DEVs. The first is that the fragments of an MSDU shall be all of the same size, except for the last one. The second requirement is that once the MSDU has been fragmented and the MAC has attempted to deliver the first MPDU, it cannot refragment the MSDU. The third major constraint is that the receiving MAC is not allowed to pass up pieces of the MSDU until it has received all of the fragments. The last requirement implies that the receiving DEV needs to have enough buffer space allocated to store the maximum size MSDU for each of the streams that it wants to be able to receive. The constant pMaxTransferUnitSize[22] is the largest MSDU that can be presented to the MAC for delivery. As this has a direct impact on the physical implementation of the DEV, this constant is PHY dependent. The 2.4 GHz PHY limits this to 2044 octets, which also the largest data payload that is allowed in an MPDU. This limit was originally 64 kbytes and it was reduced in order to save on the memory requirements. For example, to support fragmentation on eight simultaneous streams plus one asynchronous connection with a 64-kbyte pMaxTransferUnitSize would require just under 600 kbytes of RAM. This would make the cost of an 802.15.3 solution too high for the extremely cost-sensitive consumer electronics market. In addition, many other protocols use maximum packet sizes that are smaller.

The numbering of the fragments prompted quite a bit of discussion in the task group. The initial method for numbering and tracking the fragments used an MPDU number, unique for each stream, that incremented for each frame that was sent over the air. In addition, there was a fragment start and a fragment end bit in the MAC header. If the frame was one fragment long, then both the fragment start and fragment end bits would be set. If, on the other hand, one of the two fragment bits was not set, then the MPDU is one fragment of a larger frame. The first fragment would have the fragment start bit set and the fragment end bit not set. The fragments in the middle of the frame would have neither bit set, whereas the last fragment of the MSDU would have only the fragment end bit set. This method provides an unambiguous method for the receiver to correctly assemble a frame.

However, this made the logic in the receiver somewhat more difficult to implement. For example, consider the case of two MSDUs, each split into five

[22] The PHY constants, i.e., those that begin with a "p," are defined in Clause 11 of the standard and are cross-referenced in the last table in Clause 8 of the standard.

INTEL-1010
U.S. Patent 9,265,043

fragments. Suppose the receiver gets the first three fragments of the first MSDU and the last three fragments of the second MSDU. In this case, the destination would be not always be able to immediately determine that it had pieces of two different MSDUs or if it had pieces of a single MSDU. If the fragment sizes used for the MSDUs were different, then the DEV would know that there was more than one MSDU. In the case of no-ACK frames, the receiving DEV would discard all of the fragments because it did not have all of the parts of the MSDU. If the source used Imm-ACK, then the source DEV would not skip fragments unless it was going to give up on an MSDU. In this case, any gap in the MPDU number would indicate that the source DEV has given up on sending the frame, and so the destination DEV should discard any fragments that it has that have a lower number.



At least three different voters in different letter ballots suggested fragment numbering methods that were essentially the same as the original fragmentation numbering method. However, the task group felt that they had already spent enough time discussing the issue and did not want to revisit it.

The only problem with the old fragmentation scheme was with the Dly-ACK mechanism. When the source uses the Dly-ACK mechanism, it may send multiple frames without waiting for individual ACKs and then request the group of ACKs at a later time. With Dly-ACK, it would be possible for the receiving DEV to get the beginning of one frame and the end of another and not know they belonged to different MSDUs. The receiving DEV can hold on to all of the fragments until its internal timer determines that they should be discarded. But without knowing the MSDU from which the fragments came, the receiving DEV would not know how to group the fragments in its memory. In particular, because the fragments are only sent up as part of a complete MSDU, the MAC would typically group the fragments in memory based on the MSDU of which they were a part. Without knowing the MSDU number, it is more difficult, but not impossible, for the MAC to keep track of the memory allocations for the incoming frames.

To assist the MAC in organizing its memory, the new fragmentation process uses 3 octets (one more than before) that contain a 9-bit MSDU number, a 7-bit fragment number, and a 7-bit number that is the fragment number of what will be the last fragment. Thus, 802.15.3 has a limit of no more than 128 fragments for a single MSDU. The MSDU number is incremented on a per stream index basis just

INTEL-1010
U.S. Patent 9,265,043

as the MPDU number was with the old method. With the new method for keeping track of fragments, the destination is able to immediately determine where to put any received frame in its memory. If the MSDU number is new, the receiver sets aside a new buffer based on the size of the received fragment and the number fragments of the MSDU, which is determined from the last fragment number. The received frame can then be put into the correct location because the destination knows the sequence number of this fragment. Although this method is not as efficient in terms of the use of the bits in the frame header, it does make the receive state machine somewhat easier to implement.

Note that the combination of the MSDU number and the fragment number is a total of 2 octets. This was done so that the field for the Dly-ACK frame, which uses these two numbers, would also be an even number of octets.

While a DEV may fragment any MSDU that it desires, only certain MAC command data units (MCDUs) may be fragmented. This restriction was introduced to simplify the implementation of the state machines at both the transmitter and the receiver. However, some of the commands are potentially much longer than the maximum frame length and so the standard allows these to be fragmented. The commands that may be fragmented are limited to the following:

a) PNC Information
b) PNC Handover Information
c) PS Set Information Response

The fragmentation and defragmentation of these commands follows the sa procedure described previously for data frames.

Retransmissions and duplicate detection

In order to support reliable connections, IEEE Std 802.15.3 supports both retransmissions and duplicate detection. If the source DEV is using either Dly-ACK or Imm-ACK, it will know when one of its frames has either failed to arrive at the destination or that the source DEV was unable to receive the ACK. When the source DEV determines that it has lost a frame, it can retransmit that frame so that the destination DEV will eventually get the missing data. In the CAP, the DEV is required to use the contention-based access method (CSMA/CA)

INTEL-1010
U.S. Patent 9,265,043

to determine when it will be able to attempt to retransmit the frame that was lost. On the other hand, in a CTA, the source DEV is not sharing the time with any other DEVs, and so it can retransmit the frame as soon as it is practical.

The other part of reliable communications is the ability of the DEV to detect when it has received a duplicate frame. Duplicate detection prevents more than one copy of an MSDU from being passed up to the higher layer protocols. An 802.15.3 DEV uses the DestID, SrcID, PNID, fragmentation control field, and stream index to determine if a frame is a duplicate of one that it has already received. In the fragmentation field, the DEV keeps track of the MSDU number and fragment number for a given stream index to watch for duplicate frames. In the case of a isochronous stream, it is easy for the DEV to determine that it has already received a frame due to the fact that the MSDU numbers will be sequential and usually will not have missing MSDU numbers. The case is quite different for an asynchronous connection because a single MSDU counter is used for potentially multiple destinations. For example, consider a DEV, DEV-1, that is sending asynchronous data to two destination DEVs, DEV-2 and DEV-3. DEV-1 sends a frame to DEV-2 with MSDU number 0 and then sends 511 frames to DEV-3 with MSDU numbers 1 through 511. If the next frame is to be sent to DEV-2, it would have MSDU number 0 and would appear as a duplicate frame. Thus, DEVs will need to use a timeout mechanism in addition to the fragmentation field to determine if it has received a duplicate asynchronous frame.

POWER MANAGEMENT

After QoS, the next most important feature of the 802.15.3 MAC is its ability to provide advanced power management features. The target market for 802.15.3 includes a large number of portable products where power usage is a critical performance measure. Although it is important to do as much as possible to reduce the active power required by the MAC and PHY, really large power savings can only come from turning off the DEV. For most DEVs, the majority of the time that the DEV is operating is spent in receive mode, waiting for something to happen in the piconet. Because of this, the transmit power requirements are secondary in importance to the receiver power requirements, especially for relatively short range PHYs like 802.11b and the 2.4 GHz PHY of 802.15.3.

INTEL-1010
U.S. Patent 9,265,043

Instead, it is the receiver power that dominates the battery life. Thus, an efficient power savings method needs to make sure that the receiver is only turned on when it is absolutely necessary.

The great improvement in standby time for digital cellular phones over analog phones is not due to using a different modulation or due to the digital nature of the information. The improvements in battery life are a direct result of the change from an always-on receiver for analog phones to a sometimes-on receiver for the digital phones. In digital cellular standards or specifications, e.g., GSM, TDMA, CDMA, and iDEN™, the base station will allow the phone to skip listening periods as long as it wakes up at the specified time to be able to receive an incoming call. These sleep periods can be relatively long so that the phone's radio receiver is only on a small fraction of the time. The length of time that the phone is able to sleep is limited by the latency required by the system. When a call starts, the first latency that a user experiences is measured from when they finish dialing until the other phone rings.

Normally, this latency would require less than one second for the call setup to occur, including waking up the sleeping phone. The system is able to hide the latency due to the phone waking up by sending a ring tone to the calling party even before the destination cell phone rings. The ring tone keeps the calling party from hanging up too early on the call. This allows the cell phone to be asleep for longer periods of time to save power. On the other hand, the iDEN protocol (more commonly known by the Nextel™ brand) requires that the phones listen more often in order to support the two-way walkie-talkie feature. This feature allows the caller to push a button to be "instantly" connected to the other users in the group. Although the lower latency provides some clear user advantages, it also makes it more difficult to get long standby times out of an iDEN phone.

To summarize, a good power management system that supports QoS requirements for portable devices must meet the following criteria:

a) The DEV's receiver needs to be turned off for as much time as possible.

b) The DEV should be given opportunities to turn off for >10 ms, to allow deep sleep modes (<20 μA of current).

c) The latency incurred in waking up the DEV needs to meet the application requirements.

INTEL-1010
U.S. Patent 9,265,043

d) DEVs that need to communicate must be awake at the same time, i.e., their wake cycles are synchronized.

e) The DEV's status as awake or asleep should be available to all members of the piconet.

f) The sleeping DEVs need to be able to preserve QoS for low data rate connections.

g) The DEVs need to be able to maintain current security information.

The 802.15.3 power management modes were designed to enable battery-powered DEVs to meet the criteria listed in a) through g) in order to get longer battery life with good QoS. In the standard, a DEV is in one of four power management modes: ACTIVE, DSPS, PSPS, or APS.[23] A DEV may be in both DSPS and PSPS modes at the same time, but it is not allowed to be in either ACTIVE mode or APS mode at the same time that it is in any other mode. If the DEV is in DSPS, PSPS, or APS modes, the DEV is said to be in a power save (PS) mode. DEVs that are in either DSPS or PSPS modes are said to be in an SPS mode because both modes offer synchronized sleep times. No matter what mode a DEV is in, it may be in one of three states, transmitting, receiving, or off. The mode that the DEV is in determines when it is required to be receiving. The DEV is never required to transmit at a particular time, with the exception of sending ACK frames when requested.

ACTIVE mode is defined as a DEV that is listening to every beacon, the CAP (if present), and all CTAs that have either the BcstID or the DEV's DEVID as the DestID. In addition, the DEV may be transmitting during the CAP or any CTAs where it is the source.

A DSPS or PSPS mode DEV, on the other hand, is only required to listen to beacons periodically. The beacons that the DEVs are required to listen to are called the wake beacons. DEVs that are in an SPS mode and are in the same PS Set share the same wake beacons, and so all of the DEVs in that PS set wake up in a synchronized fashion. This allows DEVs that share similar application requirements to be available at the same time. The latency in sending data due to a DEV waking up is reduced because DEVs that belong to an SPS set will always be

[23] Note that the names of the modes are always capitalized in the standard.

INTEL-1010
U.S. Patent 9,265,043

listening for assigned channel time at the same time. In addition, the Wake Beacon Number of all of the DSPS and PSPS DEVs is included in every beacon so that all of DEVs in the piconet will also know when to expect that the DSPS or PSPS DEV to be listening.

DEVs in APS mode, on the other hand, do not have a wake beacon that is known to either the PNC or to other DEVs in the piconet. Instead, the DEV internally determines when it is going to be awake. The only restriction for the DEV is that it needs to send a frame to the PNC at least once within an ATP to maintain its association in the piconet. For these DEVs, the maximum latency to wake them up is their ATP, which can be quite long (in excess of 65 s). This mode is used for DEVs that wish to enter a very low power state while remaining available for occasional communications.



Timing in the piconet is relative to the beacons, and every beacon is identified by a 2-octet beacon number. The beacon number is incremented for each beacon that is sent or should have been sent (the PNC can stop sending beacons to scan other channels) and rolls over to zero once it has reached the maximum value of 65,535. The beacon number is used to indicate the time of events in the future for the members of the piconet. For example, PNC Handover IE and Piconet Parameter Change IE both use a beacon number to indicate when the event will occur. In the case of a power save DEV, the beacon number is used to indicate the next superframe that the DEV will be AWAKE.

A summary of the four modes for 802.15.3 and their required states is given in Table 4–4.

Table 4–4: Summary of state requirements for the four modes

Superframe portion	ACTIVE	APS[a] in wake superframe	SPS[b] in wake superframe
Beacon	AWAKE	AWAKE	AWAKE
CAP	AWAKE	May SLEEP	AWAKE
CTAs with BcstID as DestID	AWAKE	May SLEEP	AWAKE
CTAs with McstID as DestID	May SLEEP	May SLEEP	May SLEEP
CTAs with DEV as SrcID or DestID	AWAKE	May SLEEP	AWAKE
All other CTAs and unassigned intervals	May SLEEP	May SLEEP	May SLEEP

[a] When the DEV in APS mode, it wakes up periodically to renew its ATP timeout.
[b] In the wake superframe for the SPS set, otherwise the SPS DEV is allowed to sleep.

INTEL-1010
U.S. Patent 9,265,043

In addition to the PS modes provided in the standard, DEVs in ACTIVE mode are only required to listen during portions of the superframe as indicated in Table 4–4. If the DEV is not participating in a CTA as either the source or a destination, it can shut down its transmitter and receiver to save power. This is in contrast to 802.11, where the STAs that are active are required to listen at all times. The reason that STAs in 802.11 need to listen at all times is that they do not know when a frame for which they are the destination will be sent. On the other hand, in 802.15.3, the scheduling of the time slots is given in every beacon, which allows the DEVs to know in advance when their participation will be required.

Another key difference between the power-save modes for 802.11 and 802.15.3 has to do with the network topology. In 802.11 infrastructure mode, all communications go through the AP and so there is no need for sleeping STAs to be synchronized in their wake cycles. Instead, the AP buffers all of the traffic and signals for the STA to wake up with the traffic indication map (TIM). When the STA finally wakes up, it notifies the AP, which then forwards the stored traffic to the STA. However, in 802.15.3, all communication is peer-to-peer, which greatly improves the efficiency of the network, but it also requires that the source and destination are awake at the same time. IEEE Std 802.15.3 provides a power-save mode, DSPS, that enables DEVs to synchronize their wake cycles. An added benefit of this is that these wake cycles limit the latency for data delivery, as opposed to 802.11 which provides no latency guarantee.

Common characteristics of the SPS modes

The DSPS and PSPS modes share the concepts of a wake beacon and a PS set. A PS set is a group of DEVs with similar power save requirements that share the same wake beacons. The PS Set Indices are assigned as follows:

a) PS set 0 is used for APS mode.

b) PS set 1 is used for the PSPS set (there is only one PSPS set).

c) PS sets 2-253 (0x02-0xFD) are used for DSPS sets.

d) PS set 254 (0xFE) is the unallocated DSPS set.

e) PS set 255 (0xFF) is reserved.

INTEL-1010
U.S. Patent 9,265,043

The APS Set Index (0) and the APS mode is a special case, as the DEVs that participate in this mode really do not belong to a set because they do not necessarily share wake beacons with any other DEV. Although the APS DEV will be in AWAKE state on a regular basis to renew its ATP timer with the PNC, none of the other DEVs in the piconet, including the PNC, will be able to predict when this will happen. APS DEVs are assigned PS Set Index 0 in order to reuse the commands and concepts of DSPS and PSPS. In the following discussion, the references to the capabilities of DEVs belonging to a PS set does not apply to APS DEVs.



In the task group discussions, the APS mode was sometimes referred to as "cowboy mode" because the DEV gets to do pretty much anything that it wants as long as it sends one frame to the PNC every ATP. Because the PNC is already required to maintain an ATP timer for each DEV in the piconet, APS mode does not introduce a significant burden on the PNC.

A DEV sends the PS Set Information Request command to the PNC to find out information about the SPS sets that are available. When the PNC receives this command, it replies with a PS Set Information Response command. This process is illustrated in Figure 4–19. The PS Set Information Response command contains information about the number of sets that the PNC supports and a number of PS Set Structures, one for each of the PS Sets that have members. The PNC will support at least 1 DSPS set in addition to PSPS and APS, but it may support up to a total of 252 DSPS sets.[24] The PS Set Structure contains a list of all of the DEVs that are members of that set as well as the Next Wake Beacon (i.e., the next time they will be awake) and the periodicity of those wake beacons, the Wake Beacon Interval.

[24] 252 SPS sets are probably more than any network or application would need to support, but there is no reason to limit the number to less than the octet that is assigned to number the sets. If an implementer is able to support that number in a cost-effective manner, there is no reason for the standard to prohibit it.

INTEL-1010
U.S. Patent 9,265,043

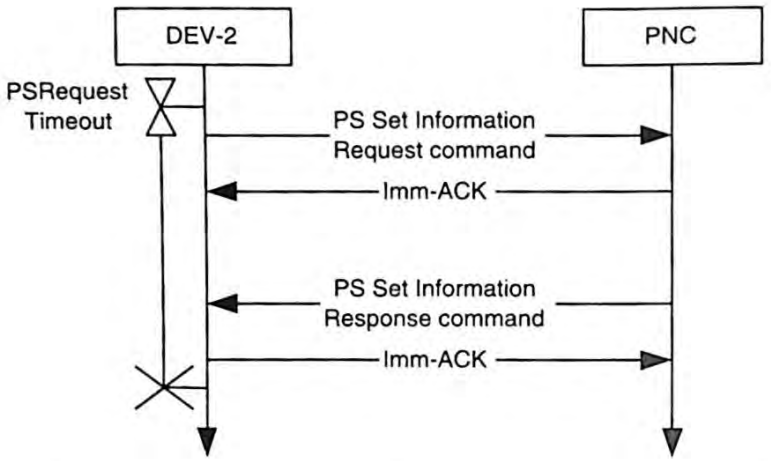


Figure 4–19: Message exchange for a DEV requesting PS set information



A DEV in APS mode is a member of PS Set 0 (the APS set). It may also be a member of other PS sets, e.g., the PSPS set or a DSPS set, while it is in APS mode. When a DEV switches between power management modes, only the APS set membership changes. A DEV remains a member of an SPS set until it is disassociated or requests to leave the set.

A DEV needs to first become a member of one or more SPS sets before it can switch to DSPS or PSPS mode. A DEV may be a member of more than one SPS set and yet not currently be in a PS mode. One of the reasons for this is that an SPS set exists only as long as there are DEVs in the piconet that want to use it, i.e., only as long as there are DEVs that are members of the set. If the DEV left the SPS set every time it switched from an SPS mode to ACTIVE, the PNC would be constantly deleting and creating SPS sets. However, the PNC is only able to support a finite number of SPS sets, and so it is important that unused sets are reclaimed by the PNC to be used for other DEVs. Thus, a DEV should leave an SPS set when it no longer needs to use it for power save operation.

Once the DEV has determined the characteristics of the SPS sets that are available in the piconet, it may do one of three things:

a) Join an existing SPS set (either DSPS or PSPS), because the set has parameters similar to the ones that the DEV requires.

INTEL-1010
U.S. Patent 9,265,043

b) Ask the PNC to create a new DSPS set with the parameters that the DEV requires. The DEV can only do this if the number of DSPS sets already allocated is less than the total number supported by the PNC.

c) Remain in ACTIVE mode because there are not any new sets available and none of the existing sets meet the needs of the DEV.

The option c), remaining in ACTIVE mode, will likely cause the DEV to use power at a greatly increased rate, to the extent that the user will notice the difference in performance. As an example, consider a digital cellular phone that supports analog roaming (i.e., a dual-mode phone). Although the phone might last five days on standby in digital mode, it might get only seven hours of standby in analog mode. As many users know from experience, the battery life is dramatically different if the phone cannot find a digital tower and instead roams to analog mode. In a consumer electronic device, this could lead the user to return the product, which incurs a large cost for both the store and the manufacturer. Unfortunately for the manufacturer, instead of returning the DEV that is the PNC the user will likely return the portable DEV that ran out of power too fast. Thus, even though the capabilities of the PNC determine the battery life of portable DEVs in the piconet, the typical customer will not know this technical detail and will likely blame the DEV that runs out of power. It is for this reason that the standard requires that all PNC-capable DEVs support APS, PSPS, and at least one DSPS set.

Analyzing power save efficiencies

The key characteristic of any power save method is its effect on improving battery life. In most consumer applications, roughly 10% of the total available capacity of the battery can be used for wireless connectivity. Longer battery life is always preferred, but many product designers use a 10% reduction in the battery life as the point of pain above which they would not include an additional feature, such as wireless connectivity. Battery life is generally measured in mAH (the number of mA used in 1 hour) and ranges from 600 to 900 mAH for cell phones to greater than 9000 mAH (when converted to 3 V) for a laptop. This gives acceptable current usage numbers that range from 90 mAH to 900 mAH to provide the wireless connectivity.

INTEL-1010
U.S. Patent 9,265,043

For almost every portable consumer electronic device, most of its life is spent doing nothing; i.e., it is in a sleep state. Thus, the focus on any power savings strategy is to develop a method by which the device spends as little time awake as possible while still maintaining the responsiveness required by the user. The measure of this capability is usually referred to as the standby time of a device. A first-order approximation of the power savings provided can be calculated by summing the total current used when the DEV is in its standby mode with the total current used while the DEV is receiving, or if required, transmitting. In the case of 802.15.3, the power-save modes do not require any significant time for transmission, so the receiver current draw will tend to dominate the power drain.

The total current used during an interval, t_{int}, can be calculated using the currents required for sleep and receive, i_{sleep} and i_{rec}, respectively, and the time required for reception, t_{rec}, using the following equation:

$$\text{total current} = i_{rec}t_{rec} + i_{sleep}(t_{int} - t_{rec})$$

The above equation is a simplification. For example, it does not include any ramp-up or ramp-down time for the DEV that is required for it to prepare to receive. However, this can be accounted for by scaling the i_{rec} value appropriately. For example, if the DEV requires 5 mA for 1 ms in warm-up before it begins to use 30 mA to send a 200 µs frame, then the i_{rec} value can be changed from 30 mA to 55 mA (30 + 5*1000/200) to include the current used during warm-up. The sleep interval can also be represented as the number of superframes that the DEV is allowed to sleep, n_{sleep}, multiplied by the superframe duration, t_{sf}. Likewise, the DEV is normally only required to be in receive mode for a fraction of the superframe; i.e.,

$$t_{rec} = c_{rec}t_{sf}$$

The ratio, c_{rec}, is calculated by dividing the beacon duration by the superframe duration. A 500 octet beacon at 22 Mb/s is approximately 200 µs, whereas a typical superframe is likely 20 ms, giving a value of $c_{rec} = 0.01$. Likewise, the current for the sleep mode can be represented with a similar equation:

$$i_{sleep} = c_{sleep}i_{rec}$$

INTEL-1010
U.S. Patent 9,265,043

Typical values for the ratio of receive current to sleep current, c_{sleep}, range from 0.001 to 0.01.

Using the preceeding equations, the total current can be represented as:

$$\text{total current} = i_{rec} c_{rec} t_{sf} + i_{rec} c_{sleep} (n_{sf} t_{sf} - c_{rec} t_{sf})$$

which can then be simplified to:

$$\text{total current} = i_{rec} t_{sf} (c_{sleep} n_{sf} + c_{rec} - c_{rec} c_{sleep})$$

For a typical radio, the values of c_{rec} and c_{sleep} will likely be small, so that the last term can be deleted as it will be much smaller than the other two terms in the parentheses. Finally, the power savings that is achieved by a method that allows a DEV to sleep for a number of superframes compared with waking up for every superframe, i.e., $n_{sf} = 1$,can be calculated from:

$$\text{increase in battery life} = \frac{c_{sleep} n_{sf} + c_{rec}}{c_{sleep} + c_{rec}}$$

Using values of 0.01 for c_{rec} and c_{sleep}, the increase in battery life is plotted in Figure 4–20. As expected from the simplified equation, the increase in battery life is linearly proportional to the increase in the interval between wake beacons. This is intuitive because the power usage of the DEV will rise roughly linearly with the number of times it needs to be in receive mode, because this dominates the overall power usage.

Doubling the Wake Beacon Interval (802.15.3 only allows powers of 2 for the Wake Beacon Interval) will then lead to a significant improvement in the overall battery life. This result is graphed in Figure 4–21 using 0.01 as the value for c_{rec} and c_{sleep}. As the graph shows, once the Wake Beacon Interval is increased beyond 16 (i.e., 2^4), doubling the Wake Beacon Interval doubles the battery life. Even for relatively short Wake Beacon Intervals, the improvement is at least a factor of 1.6. This result provides the primary motivation in 802.15.3's power save strategy: Allow the DEVs to sleep for as long as they need to support the requirements of their application.

INTEL-1010
U.S. Patent 9,265,043

The preceding calculations do not take into account any sub-rate traffic that the DEV may need to either send or receive while it is in sleep mode. If the time spent on this traffic is significant, then the improvement from skipping wake beacons will be less pronounced. In this case, the sub-rate interval would have a stronger impact on the overall battery life. In any case, using the best sub-rate or Wake Beacon Interval is key to improving the battery life of the DEV.

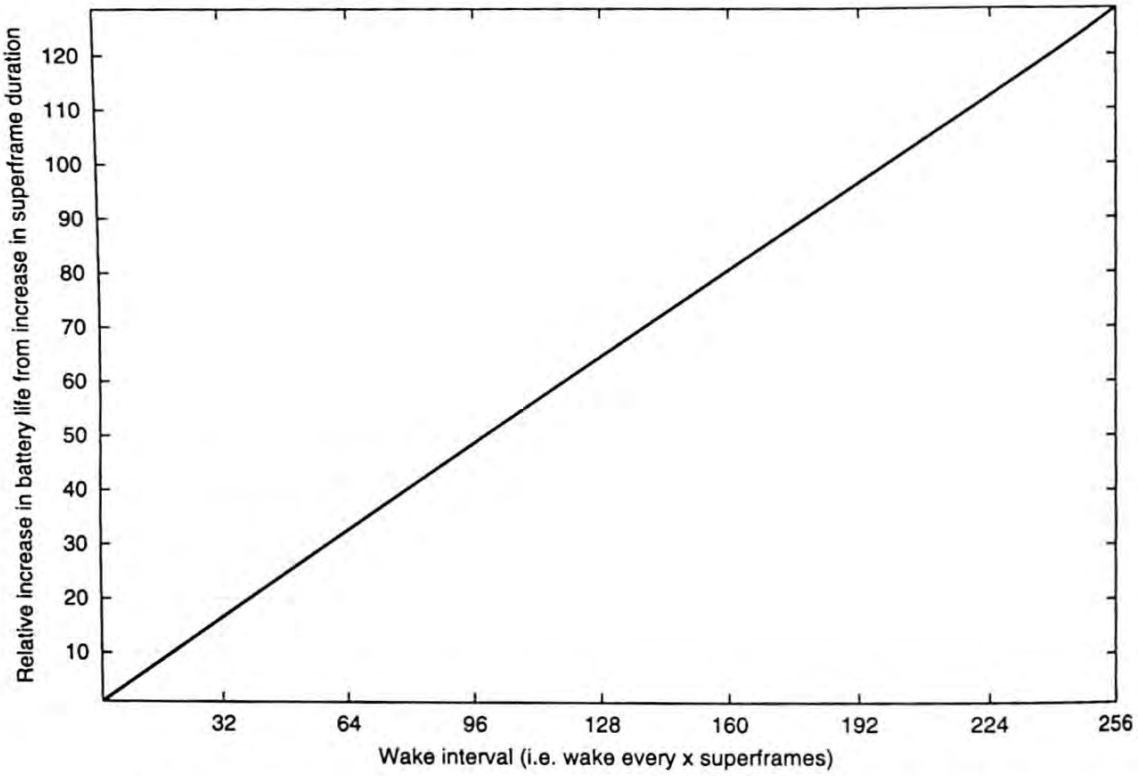


Figure 4–20: Increase in battery life as a function of the number of superframes skipped

Switching PM modes

A DEV changes its PM mode by sending the PM Mode Change command to the PNC with the PM Mode field set to the appropriate value. If the DEV is making a valid request, the PNC will set the bit for the DEV in the appropriate PS Status IE in the next beacon. When the DEV receives a beacon with its bits set, it knows that it can change its PM mode. The PS DEV needs to wait for this indication from the PNC so that other members of the piconet can be informed that the PS

INTEL-1010
U.S. Patent 9,265,043

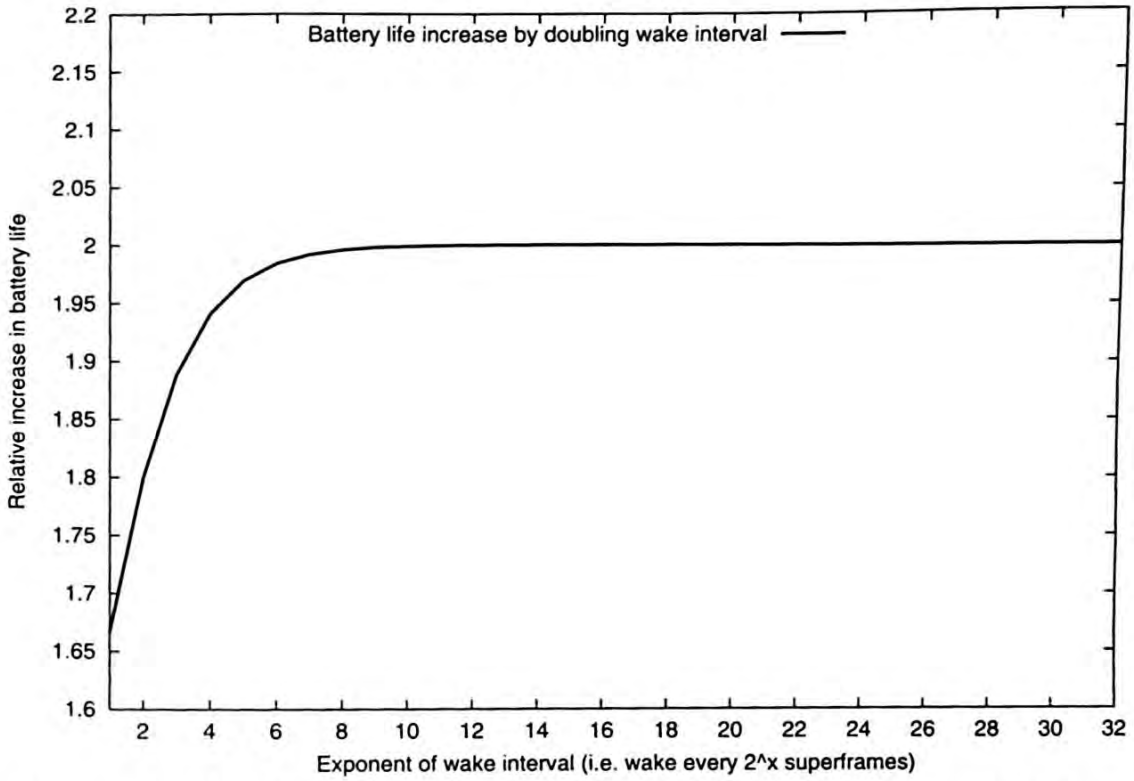


Figure 4–21: Increase in battery life as the wake interval is doubled

DEV will not be listening to every beacon. If the PM Mode field is set to "SPS," the PNC will register the DEV as participating in all of the SPS sets that it has joined. For example, if the DEV is a member of the PSPS set and two other DSPS sets when it sends the PM Mode Change command with the PM Mode field set to "SPS," the DEV will be required to listen to the wake beacons of all three sets. Figure 4–22 illustrates the frame exchange for a DEV changing from ACTIVE mode to DSPS, PSPS, or both.

Normally, the PNC will always grant a request to change the DEV's PM mode. The only reason for the PNC to reject this request is if the DEV is requesting to change to SPS mode and it is not currently a member of a valid SPS set (i.e., a set with a value of 1–253). If the DEV is not a member of a valid SPS set, the PNC will do nothing with the beacon and the DEV will still be required to be in ACTIVE mode and to listen to every beacon.

INTEL-1010
U.S. Patent 9,265,043

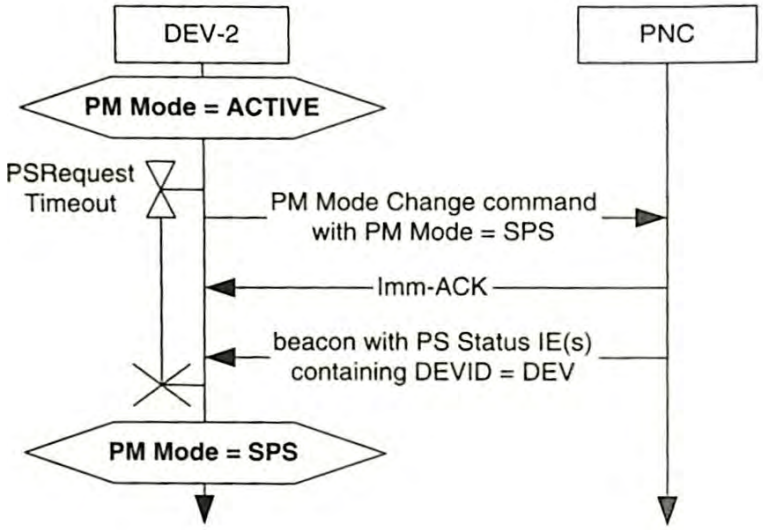


Figure 4–22: DEV changing from ACTIVE mode to DSPS, PSPS, or both

A DEV changes from any PS mode (either APS or SPS) to ACTIVE mode by sending the PM Mode Change command to the PNC with the PM Mode field set to "ACTIVE". This frame exchange is illustrated in Figure 4–23. The DEV will consider itself to be in ACTIVE mode once it has sent the PM Mode Change command to the PNC, even if the DEV doesn't receive an ACK. The reason for this is that the DEV will not harm the piconet operations if it begins listening to every beacon. In additon, there is no reason for the PNC to refuse the request, as there is in the case when the DEV switches from ACTIVE to SPS mode.

Even if the ACK was not received by the DEV, it is still possible that the PNC correctly received the PM Mode Change command. If the DEV does not see a change to its PS status in the beacon after requesting the change to ACTIVE mode, it can re-send the PM Mode Change command to the PNC. If the DEV was in SPS mode, it will still remain a member of its SPS sets after it changes to ACTIVE mode. A DEV retains its SPS set membership until it explicitly requests to leave the set by sending the proper the SPS Configuration Request command to the PNC.

INTEL-1010
U.S. Patent 9,265,043

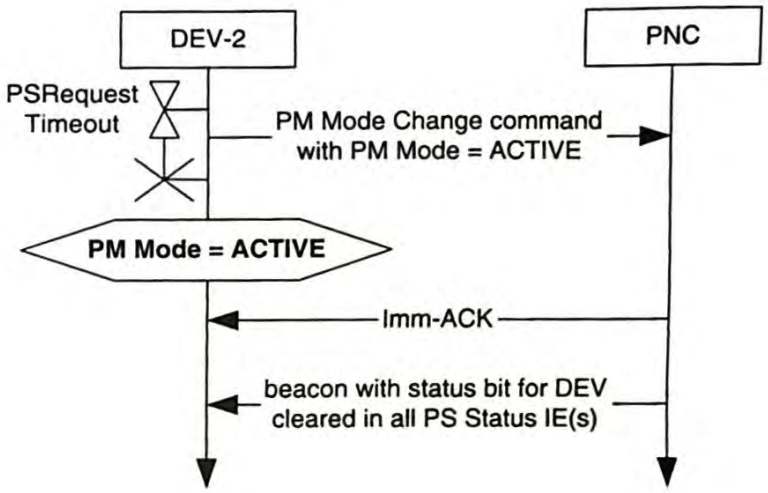


Figure 4–23: DEV changing from any PS mode to ACTIVE

A DEV changes from ACTIVE mode to APS mode by sending the PM Mode Change command to the PNC with the PM Mode field set to "APS," as illustrated in Figure 4–24. The DEV considers itself to be in APS mode when it receives the ACK to its PM Mode Change command. When a DEV changes from any other PM mode to APS mode, all of the streams and asynchronous allocations where the DEV is either the source or destination will be terminated by the PNC.

If the PM Mode field in the PM Mode Change command is set to a value that is the same as the current mode of the DEV, then the DEV is not requesting a change in its PM mode. The PNC takes no action in response to this command and the PS Status IE(s) remain the same after it has received the command.

The PS Status IEs are used to keep the DEVs in the piconet informed of the current power save status of every DEV in the piconet. The PNC places a PS Status IE in the beacon for each PS Set that has a DEV in a PS mode. The PS Status IE contains the PS Set Index, the set's Next Awake Beacon, and a bitmap that indicates the DEVs in the set that are currently in a PS mode. For example, if there is a DEV in APS mode, the PNC places a PS Status IE in the beacon with PS Set Index set to 0 (the APS set) and the bit that corresponds to that DEV set to 1. Likewise, if a DEV has joined the PSPS set and two DSPS sets prior to changing to SPS mode, the PNC will place three PS Status IEs in the beacon, one for set 1

INTEL-1010
U.S. Patent 9,265,043

(the PSPS set) and one for each of the DSPS sets that the DEV has joined. The PNC only places one PS Status IE per active PS set in the beacon.

Because the PS Status IE is in every beacon when there is a DEV in a PS mode, all of the DEVs in the piconet will know which DEVs are in a power-save mode. In addition, the DEVs in the piconet use the Next Awake Beacon field to determine the beacon number of the superframe when the power save DEVs will be awake to listen for traffic. This allows a DEV to determine when it will be able to connect with DEVs that are in a PS mode. The exception to this, of course, is for DEVs that are in APS mode. In this case, the other DEVs in the piconet will be informed that the DEV is in APS mode, but they will not know when that DEV will be awake to listen for traffic.

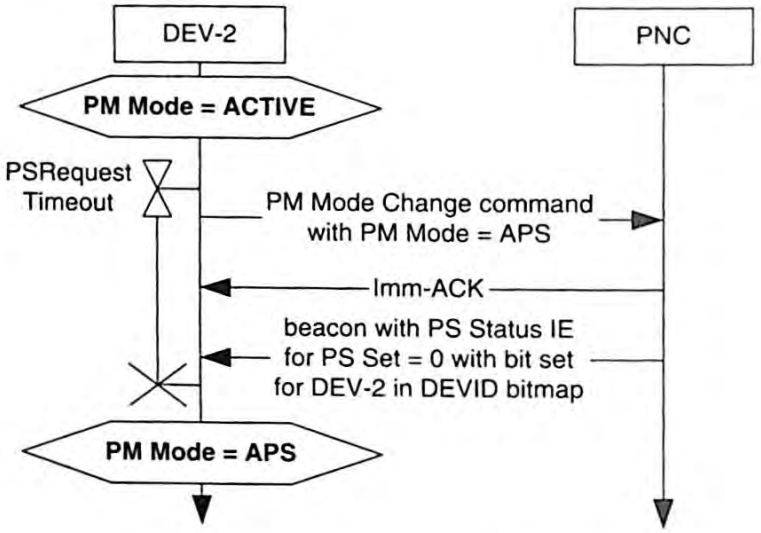


Figure 4–24: DEV changing from ACTIVE mode to APS

Managing SPS sets

Only DSPS sets can be created by a DEV because the PSPS set belongs to the current PNC. If a DEV decides to create a new DSPS set because none of the existing sets meet its needs, it sends an SPS Configuration Request command to the PNC with the PS Set Index equal to the Unallocated PS set. The Unallocated PS set has a value of 0xFE, which is the same value as the unassociated ID used in the association process. The DEV also sets the Operation Type field to indicate "join" and requests a specific beacon interval with the Wake Beacon Interval field.

INTEL-1010
U.S. Patent 9,265,043

The PNC can only accept or reject the request; it cannot modify the requested Wake Beacon Interval.

Although the PNC cannot modify the interval of the wake beacons, it can determine when the first wake beacon will occur. This offers the PNC the opportunity to spread out the wake beacons of the DSPS sets so that all of the DEVs in the piconet are not waking up at the same time. If the PNC accepts the request to create a new PS set, it assigns a number, the PS Set Index, that will be used to identify the set. The PNC then responds to the requesting DEV with the SPS Configuration Response command that includes the new PS Set Index and the Next Wake Beacon for the new PS set. The frame exchange for creating a new DSPS set is illustrated in Figure 4–25.

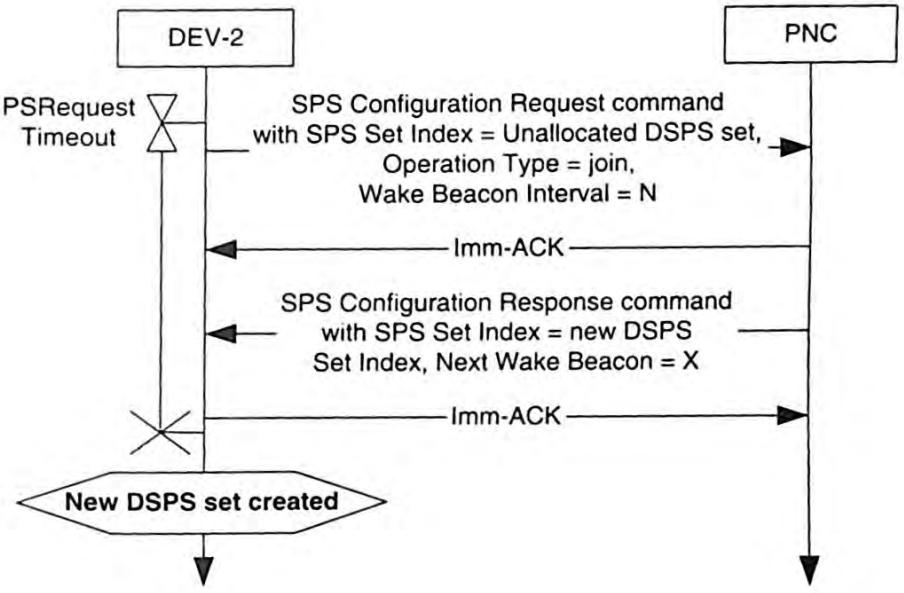


Figure 4–25: Frame exchange for creating a new DSPS set

The 802.15.3 standard uses 0xFE for the unassociated ID and the unallocated PS set. This number was selected because 0xFF had already been selected to be used for the broadcast ID. Traditionally, the broadcast address is one that has all of the bits set in some portion of the address. The next available ID, counting down from 0xFF, was 0xFE and this was selected as the unassociated ID. The task group then adopted this number for the DSPS set creation process because the PNC was assigning an ID, just as it does in the association process.

INTEL-1010
U.S. Patent 9,265,043

Once the DSPS set has been created, the PNC keeps the Next Wake Beacon for that set updated and broadcasts this number in the PS Status IE in the beacon when there are DEVs in that DSPS set that are in DSPS mode. The PNC also keeps track of the Wake Beacon Interval that was requested when the set was created. Although the PNC is allowed to change the Next Wake Beacon, i.e., it is allowed to change the locations of the DSPS set's wake beacons relative to other DSPS sets, the Wake Beacon Interval is kept constant for the life of the PS set. Neither the PNC nor a DEV is allowed to change the interval while the PS set exists.



The original draft of DSPS mode (at the time called EPS mode) allowed the DEV that created of the DSPS to modify the parameters of the set. However, if this DEV left the set, it wasn't clear which of the remaining DEVs in the set should be given the power to modify the set parameters. Instead of trying to create a line of succession of DEVs that would have the right to modify the set parameters, the task group decided that no DEV should be able to modify the set parameters. In one of the last drafts this was modified slightly. The PNC can change the Next Wake Beacon, but Wake Beacon Interval would not be changed while the set was in use.

If the DSPS sets share some of their wake beacons, then in those superframes the total duration of the CTAs requested by all of the DEVs may exceed the superframe duration. This problem is referred to as differential overloading. If every superframe is full, i.e., there is not any additional time available for CTAs in any superframe, then the PNC has a problem of superframe overloading. However, DEVs in a DSPS mode will often request sub-rate allocations that are aligned with their wake beacons to save power. In non-wake beacons, the allocations will not exist and the superframe will be under-utilized. On the other hand, when the wake beacons of multiple DSPS sets overlap, then that particular superframe may be overloaded due to the additional sub-rate slots. However, because the PNC is able to arrange the wake beacons of the DSPS sets, it is able to minimize the potential of differential overloading of the superframe.

Once a set has been created, a DEV may join the set so that it can use it for power save purposes. To join the set, the DEV sends the SPS Configuration Request command to the PNC with the PS Set Index field equal to the index of the set that the DEV wishes to join and the Operation Type set to "join." As the DEV is not allowed to change the set parameters of an existing set, the Wake Beacon Interval field is ignored by the PNC. The PNC can only reject the request if the PS Set Index in the request does not exist. If the DEV is already a member of the PS set,

INTEL-1010
U.S. Patent 9,265,043

the PNC takes no additional action other than to send a response. Otherwise, the PNC will add the DEV to the PS set as a member. Regardless of the result of the request, the PNC sends the SPS Configuration Response command to the DEV with the appropriate Reason Code to indicate if the request failed or was successful. The frame exchange for joining an existing PS set is illustrated in Figure 4–26.

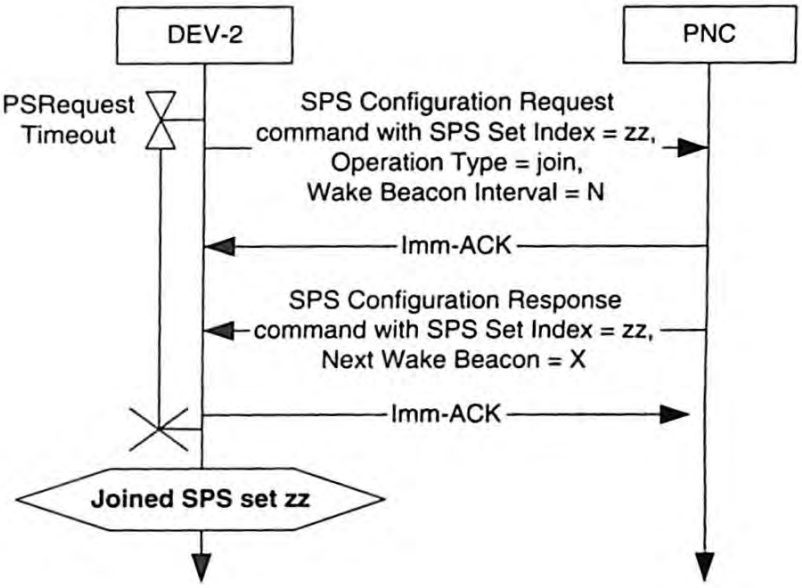


Figure 4–26: Frame exchange for joining an existing SPS set

When the DEV no longer needs to use the PS set, it sends the SPS Configuration Request command to the PNC with the Operation Type set to "leave." All of the other parameters in the command are ignored by the PNC. The PNC will always accept a request by a DEV to leave a PS set and the PNC does not send the SPS Configuration Response command to the DEV. Instead, the Imm-ACK sent in response to the SPS Configuration Request command serves as the notification that the PNC will remove the DEV from the PS set. The frame exchange for leaving a PS set is illustrated in Figure 4–27. When the last DEV has left a DSPS set, the set is terminated by the PNC. Note that the PSPS set is never terminated because it is the PNC's set, and so it exists as long as the piconet exists.

INTEL-1010
U.S. Patent 9,265,043

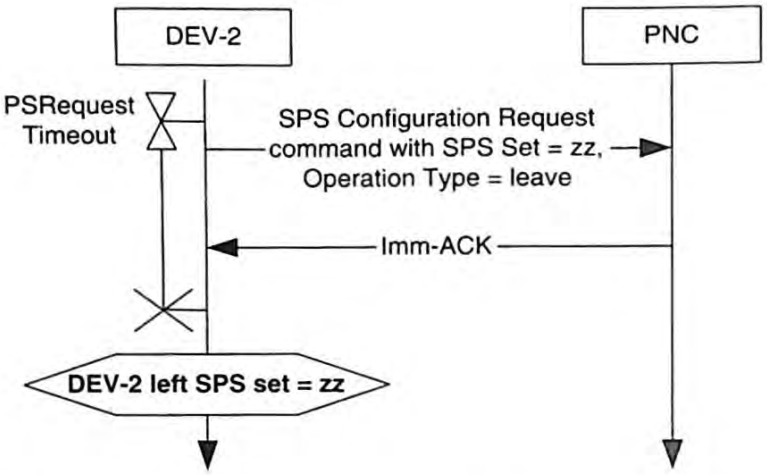


Figure 4–27: Frame exchange for leaving an SPS set

DSPS mode

DSPS mode is designed to give extremely good power savings while allowing the DEVs to control the latency of data transmission in sleeping DEVs. DSPS mode also provides a method by which DEVs that share common requirements for power savings are able wake up at the same time to communicate. The DSPS mode was the basis for the PSPS mode, which trades away some of the power savings in DSPS mode in order to provide a piconet-wide wake beacon for participating DEVs.



The original version of DSPS mode had the name extended power save (EPS) mode. This was later renamed synchronized power save (SPS) mode before the task group settled on DSPS mode.

Once a DEV has joined a DSPS set (see "Managing SPS sets" on page 114), it switches between DSPS and ACTIVE mode depending on the type of traffic that it is handling. If the DEV has infrequent traffic, it can remain in DSPS mode and use a sub-rate allocation to maintain communications with another DEV. If both the source and destination need to save power, then they would both join the same DSPS set. When a DEV changes from ACTIVE to DSPS, the PNC will terminate all streams for which the DEV is the source or destination that are not using a DSPS Wake Beacon Interval. The reason for this is that the DSPS DEV will not be listening to every beacon and hence will not be able to transmit or receive during

INTEL-1010
U.S. Patent 9,265,043

those CTAs. However, when a DEV changes from DSPS to ACTIVE mode, the PNC does not terminate any streams, regardless of the type of allocation.

When a DEV is in DSPS mode, there is no guarantee that it will be able to receive either broadcast or multicast frames. Unless the request for the broadcast or multicast allocation was made with a DSPS Set Index, the PNC will not attempt to align the allocation to the DSPS set's wake beacons. Thus, if a DEV needs to have all of the DEVs in the piconet receive the frame, it will have to repeat the message in at least one wake beacon of each of the PS sets. However, broadcast delivery is not guaranteed anyway, so the source DEV can only provide the opportunity for the DEVs to receive the broadcast frame; it cannot guarantee that they will receive it. The only method to ensure that a DEV receives a frame is to send it as a directed frame with the ACK policy set to either Imm-ACK or Dly-ACK. Likewise, if a DEV requires the opportunity to receive all broadcast and multicast frames, it should not use a PS mode.

Allocating channel time for DSPS DEVs

Although DSPS provides clear advantages in power savings for DEVs in an 802.15.3 piconet, another key advantage comes from the ability to synchronize low power, low data-rate applications while maintaining the required QoS. The DEVs accomplish this by allocating sub-rate streams that are aligned with the DSPS set's wake beacons. A sub-rate stream is one where the allocation occurs only once every *N* beacons. This type of allocation allows the DEVs to maintain a connection with a defined latency and guaranteed bandwidth. Applications that could use this include remote control functionality for A/V equipment and cordless phones (to provide a defined latency to wake up the handset). Another application is a remote security camera that is triggered on movement. It would maintain a low-rate connection to verify its status as active, but it would then switch to a high-rate stream when it had video to transmit due to activity in its area.

The DSPS sub-rate allocations are aligned with the DSPS set's wake superframes, so their allocation interval, the CTA Rate Factor, needs to be either greater than or equal to the DSPS set's Wake Beacon Interval. For example, if a DSPS set uses a Wake Beacon Interval of 8, then a DEV could request DSPS channel time with a CTA Rate Factor of 8, 16, 32, 64, and so on. This is illustrated in Figure 4–28 for

INTEL-1010
U.S. Patent 9,265,043

various Wake Beacon Intervals and CTA Rate Factors. In order to minimize the complexity required for sub-rate allocations and DSPS sets, the PNC is only required to support CTA Rate Factors and Wake Beacon Intervals that are powers of 2 (i.e., 2, 4, 8, 16, ...). In case 1 in Figure 4–28, the Wake Beacon Interval is equal to the CTA Rate Factor and so the CTAs occur in every wake superframe. In case 2, however, the CTA Rate Factor is twice that of the Wake Beacon Interval and so the CTAs occur only in every other wake superframe. Some combinations of CTA Rate Factor and Wake Beacon Interval, however, are not allowed. For example, a CTA Rate Factor of 2 would not be allowed for a Wake Beacon Interval of 8. In this case, the CTAs would occur more often than wake beacons and so some of the allocations would occur in superframes when the DEV is not going to be awake.

The DEV that is creating or modifying a stream uses the PM CTRq Type field of the Channel Time Request command to indicate if the requested allocation needs to be aligned to a DSPS wake beacon. If a DEV requests a DSPS allocation, the PNC will align the channel time with the wake beacons of the DSPS set indicated in the request. If the interval for the sub-rate allocation is greater than the Wake Beacon Interval, then the PNC has flexibility in assigning the CTAs, as illustrated in Figure 4–29.

An ACTIVE mode sub-rate allocation, on the other hand, can occur in any beacon. This gives the PNC the flexibility to spread out the loading of the sub-rate allocations among the superframes. An example of spreading out the sub-rate allocations is illustrated in Figure 4–30. In this example, eight DEVs have joined a DSPS set with a Wake Beacon Interval equal to two. They have also requested sub-rate allocations with a CTA Rate Factor equal to eight. The PNC can then pair the requests and start a different pair every other superframe, reducing a potential load of eight CTAs in a superframe to only two CTAs in a superframe.

Sub-rate allocations, however, do not necessarily have the same priority as super-rate allocations. Because the sub-rate allocations can occasionally align to overload the superframe, the PNC is allowed to skip the allocations and can then take one of three actions to handle the differential overloading problem for DSPS sub-rate allocations:

a) Take no other action, and continue the allocations if time is available in future superframes.

INTEL-1010
U.S. Patent 9,265,043

b) Terminate the stream because it is unable to service the request reliably.

c) Set the DEV's bit in the continued wake beacon (CWB) IE, and allocate the CTA in one of the three following superframes.

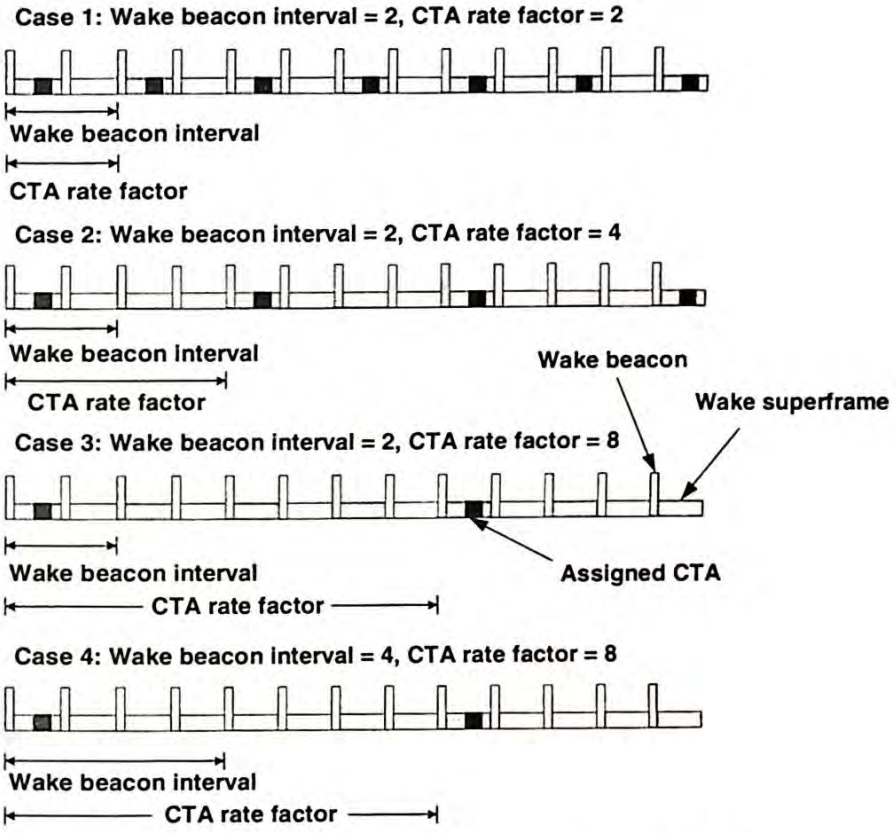


Figure 4–28: Illustration of the relationship between Wake Beacon Intervals and sub-rate CTAs

When a DSPS DEV sees its bit set in the CWB IE, it knows that the PNC was unable to provide one of its allocations in the current wake superframe. The DEV will then listen to the next beacon to see if the allocation appears in that beacon. The PNC can again set the DEV's bit in the CWB IE to keep the DEV awake for another beacon. This process, referred to as chaining wake beacons, allows the PNC to handle infrequent differential overloading problems in the piconet. However, the PNC is only allowed to set the DEV's bit in the CWB IE for up to three consecutive beacons. In general, the PNC should avoid this method as much as possible because it will have a negative impact on the battery life of the DEV.

INTEL-1010
U.S. Patent 9,265,043

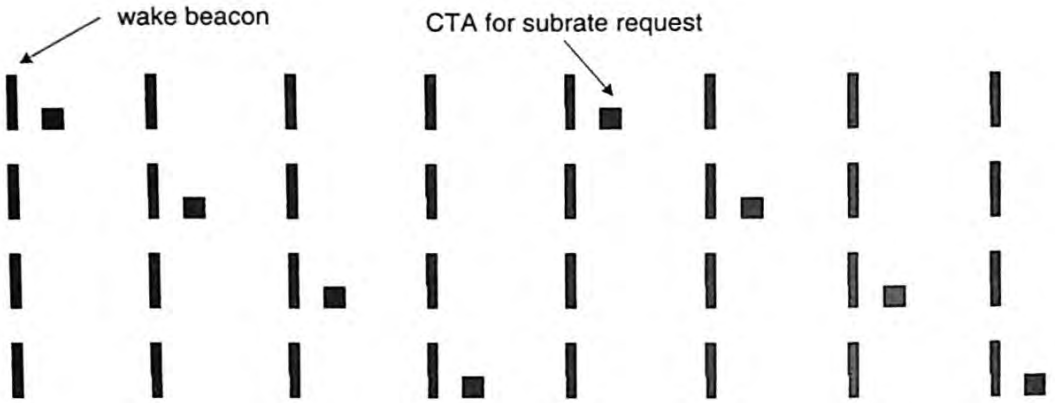


Figure 4–29: Equivalent CTAs for Wake Beacon Interval = 2 and CTA Rate Factor = 8

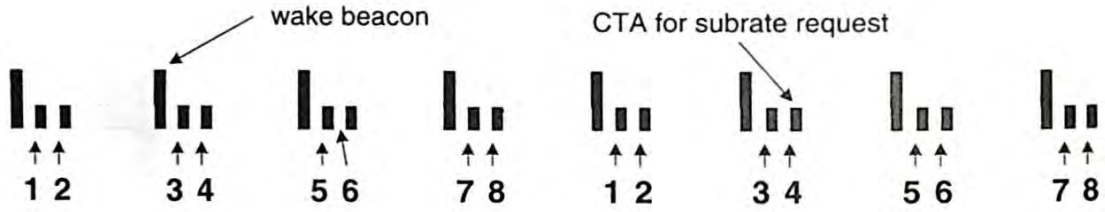


Figure 4–30: Minimum superframe loading of CTAs for 8 DEVs requesting Wake Beacon Interval = 2 and CTA Rate Factor = 8



An ACTIVE mode CTA is one where the time allocation is not intentionally aligned with a DSPS set's wake beacons. A DEV requests an ACTIVE mode CTA by setting the PM CTRq Type bit in the Channel Time Request command to be zero.

One of the requirements of any power save technique is that it provides the ability of other DEVs to "wake up" a DEV that is in a power-save mode. In IEEE Std 802.15.3, the wake up process is linked with the channel time request process. This is a natural linkage because the reason that one DEV needs to wake up another DEV is so that it can begin communications, and hence, it needs to allocate channel time for this to happen. The wake up process begins when a DEV requests an ACTIVE mode CTA with a target DEV that is in DSPS mode. If there is time available for the allocation, the

INTEL-1010
U.S. Patent 9,265,043

PNC will grant the request and send a response to the requesting DEV indicating that the request has been granted, but that the destination DEV is in a power-save mode. The PNC will then set the bit corresponding to the DEV in the pending channel time map (PCTM) IE to inform the sleeping DEV that it has a new allocation, not aligned to its wake beacon that is pending.

When a DEV in DSPS mode receives a beacon with the PCTM IE that has the DEV's bit set, the DEV will send a PM Mode Change command to the PNC. If the DEV is going to keep the allocation, it sets the PM Mode in the command to ACTIVE and the PNC will then begin to place the appropriate CTA blocks in the beacon. If, however, the DEV does not want to keep the allocation, it responds to the PNC with a PM Mode Change command with the PM Mode set to DSPS. The PNC will then terminate the stream using the standard stream termination procedure (see "Asynchronous data" on page 87).

The process of creating ACTIVE mode channel time with a DSPS DEV is illustrated in Figure 4–31.

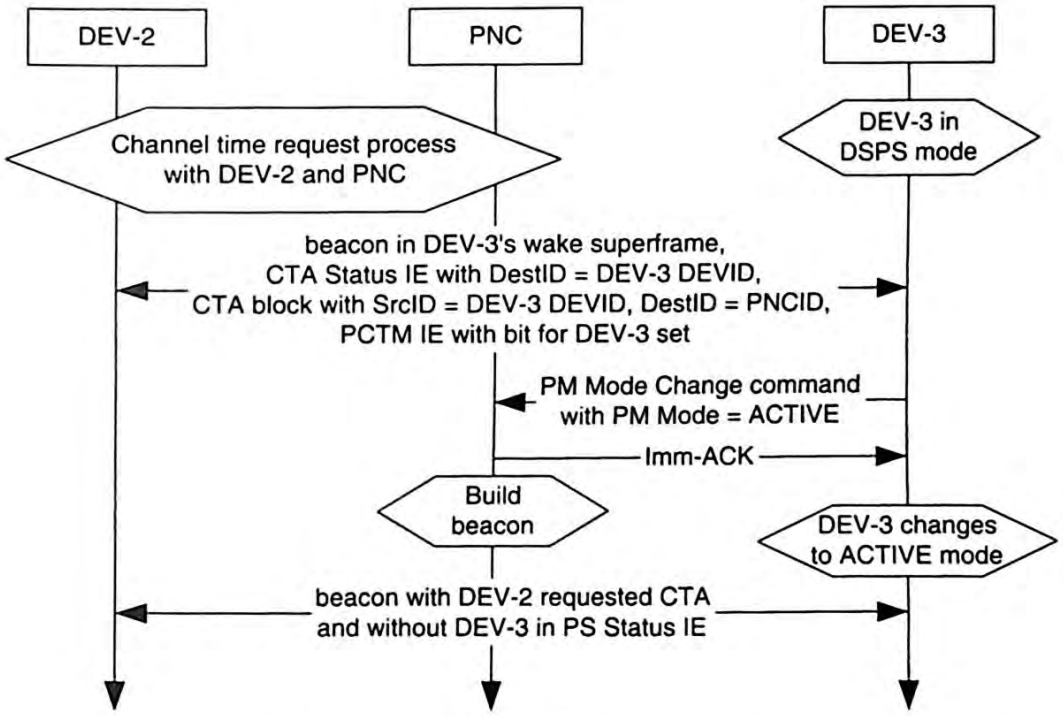


Figure 4–31: Channel time request causing a DSPS mode DEV switch to ACTIVE mode

INTEL-1010
U.S. Patent 9,265,043

PSPS mode

PSPS mode is very similar to DSPS mode. The goal of PSPS mode, however, is very different than the goal of DSPS, and so it has some characteristics that are also different. Although the goal of DSPS is to provide optimum power savings while maintaining QoS, the goal of PSPS is to provide a method for ensuring that sleeping DEVs will be awake during superframes when important announcements might be made. PSPS mode does not use a regular Wake Beacon Interval. Instead, the PNC is allowed to change the Wake Beacon Interval in between every wake beacon. Thus, the two key differences between PSPS and DSPS is that in PSPS mode it is the PNC that determines the Wake Beacon Interval and that the PNC can change that interval in every wake beacon.

When a DEV joins the PSPS set and switches to PSPS mode, it is required to listen to all system wake beacons, i.e., the wake beacons for PS set 1. If it misses receiving a system wake beacon, the DEV is required to continue to listen for beacons until it successfully receives a beacon. One reason for this is that if the DEV does receive a beacon, it will not know the beacon number of the next system wake beacon. A DSPS DEV, on the other hand, does not need to listen to subsequent beacons because it knows that the Wake Beacon Interval is fixed and so it can calculate the beacon number of the next DSPS wake beacon.



The idea for a system-wide wake beacon was first proposed at the IEEE 802 Plenary meeting in St. Louis, MO, in March 2002, as a way to handle broadcast transmission for sleeping DEVs. The idea was that power-save mode DEVs that wanted to hear broadcast traffic would be awake during the system wide wake beacon. Instead of adopting this approach, the group decided not to attempt to guarantee that all DEVs in the piconet would have the opportunity to receive broadcast traffic. If a DEV needs to hear this traffic, it cannot enter a power-save mode. The idea of a system wake beacon appeared again when the PSPS mode was proposed at the IEEE 802 Plenary meeting in Vancouver, BC, in July 2002. This mode was named PSAVE mode in its proposal but was changed to PSPS when it first appeared in a draft.

In PSPS mode, the Wake Beacon Interval is not constant, and so it is not possible for the PNC to align sub-rate allocations with the system wake beacon. Even if the PNC does allocate the first CTA during a system wake beacon, there is no guarantee that this will happen in the future. In general, a PSPS mode DEV will burn much more power than an equivalent DSPS mode DEV. One reason is that the PSPS mode DEV will listen to more beacons to make up for beacons that were

INTEL-1010
U.S. Patent 9,265,043

lost. It will also use more power because the DEV will have to wake up once to listen to the system wake beacon and will have to wake up again in a different superframe to receive frames during its sub-rate allocation. If the system wake beacon is the same interval as the sub-rate allocation, this would have the effect of almost doubling the power drain on the battery due to the DEV being required to listen to an extra beacon. The effect is less pronounced if the sub-rate occurs much less frequently than the system wake beacon, but in all cases, using a DSPS set would result in improved power efficiency.

A final issue that degrades the power save capabilities of PSPS mode is that the PNC and not the DEV determines the interval between system wake beacons. A DEV can send a suggested interval to the PNC, but the PNC is not required to honor this request. For example, suppose three PSPS DEVs request Wake Beacon Intervals of 128 while two PSPS DEVs request Wake Beacon Intervals of 16. If the PNC uses 128 as the system Wake Beacon Interval, then the DEVs with a need for an interval of 16 could suffer from higher latency. However, if the PNC uses an interval of 16, then the other three DEVs will see a dramatic increase in their power usage as compared with an interval of 128. Furthermore, the PNC may have a policy of allowing system Wake Beacon Intervals of no more than 8 to ensure a fast response time for beacon announcements. In this case, none of the DEVs would get their requested beacon interval and the power usage for all of the PSPS DEVs would increase. As shown in Figure 4–21, decreasing the Wake Beacon Interval from 16 to 8 could increase the DEVs impact on the battery life by a factor of 2.

When a DEV switches from ACTIVE mode to PSPS mode, the PNC will terminate any super-rate isochronous streams that have the DEV as the destination. If the PNC did not do this, then the PSPS DEV would either miss the CTAs that occur in every beacon or it would listen to every beacon, which is the same as being in ACTIVE mode. The PNC will allocate asynchronous CTAs that have the PSPS DEV as the destination during the system wake beacons so that the sleeping DEV has a chance to receive this traffic. If the PNC is using the group allocation method and some of the DEVs in the group are in an SPS mode, the PNC will attempt to schedule as many of the asynchronous allocations as possible during the system wake superframe, but it is not required to put any of the allocations in that superframe. DEVs that have broadcast or multicast allocations can use the system wake beacon to reach all of the DEVs that are in PSPS mode.

INTEL-1010
U.S. Patent 9,265,043

The main reason to use PSPS mode is to provide defined beacons in which certain announcements will be made. For example, if there are DEVs in PSPS mode, then the Piconet Parameter Change IEs will occur at least once in a system wake beacon and in at least three consecutive beacons that follow the system wake beacon. This improves the probability that the PSPS DEVs will find out about the upcoming change in one of the key parameters of the piconet. If a DEV misses an important change announcement, e.g., a channel change, it will have to take action to reacquire the piconet. This does not add complexity because searching for and acquiring synchronization with a wireless network is an essential function of all portable wireless devices. The uncertainty of the wireless medium requires the designer to always plan for the inevitable condition where synchronization is lost with the desired network.

APS mode

APS mode was introduced to complement the other two power-save modes. APS has two important power saving capabilities that are different from either DSPS or PSPS. The first capability compensates for the limited number of DSPS sets. Hardware limitations, e.g., storage space in the MAC and processing power, imply that the number of DSPS sets will be limited in a PNC, possibly to a relatively small number, e.g., four. Once those sets are all defined, new DEVs that have different power save requirements, e.g., low sensitivity to latency but high sensitivity to power usage, might be unable to match the desired power save performance to the existing sets. APS mode allows the DEV to choose any Wake Beacon Interval that it requires without increasing the demands on the PNC.



APS was in one of the early drafts of the standard, but was removed because the initial version was broken. A simplified version of APS that fixed the earlier problems was added during Sponsor ballot based on the argument that it added almost no complexity to the PNC. This version was named HIBERNATE, but the technical editor felt that the name was too long for it to be displayed nicely in the standard. Because the other power-save modes were already using PS as a part of their name, APS selected to be the name for the final version of this power-save mode.

The second capability provided by APS is that it does not require that the sleeping DEV keep an accurate estimate of the superframe timing. In PSPS and DSPS modes, the DEVs are required to wake up for specific beacon numbers and will need to determine an estimate of the drift between the PNC's clock and the DEV's

INTEL-1010
U.S. Patent 9,265,043

clock so that the sleeping DEV will wake up at the right time for the wake beacon. In APS mode, however, the sleeping DEV can wake up at any time and listen to the next beacon that arrives.

There are, however, three drawbacks to APS mode. The first is that a DEV in APS mode is not allowed to have any isochronous streams or asynchronous allocations while in APS mode. When the DEV switches from any other PM mode to APS mode, the PNC will terminate all isochronous and asynchronous allocations that have the DEV as either the destination or the source. In the case of a group asynchronous allocation, the PNC will remove the DEV from the target list in the CTRqB without changing the total requested allocation.

The second drawback to APS mode is that other DEVs in the piconet do not know the superframes during which the APS DEV will be listening. Thus, when a DEV requests channel time with an APS DEV, the requesting DEV will not know for sure when the APS DEV will respond to the PNC so that the allocation can be put in the superframe. The requesting DEV can be aware of the ATP value for the APS DEV because that value is included in the PNC Information command, which is periodically broadcast by the PNC. Thus, the requesting DEV will be able to place an upper limit on the length of time that it will take the APS DEV to respond, but that could end up being a very long time (the longest ATP allowed is approximately 65 seconds).



IEEE Std 802.11 provides power-save modes in a manner that is similar to APS mode in 802.15.3. In 802.11, the AP stores all frames intended for sleeping STAs and forwards them to the sleeping STA when it wakes up. The AP uses the traffic indication map (TIM) to inform a sleeping STA that it has frames that need to be delivered. Thus, the power-save modes in 802.11 only work in infrastructure mode and not in ad-hoc mode. This procedure works well because the AP is normally a stationary device that has access to AC power. However, this approach increases the memory and cost for the AP. It also introduces an unpredictable latency in the delivery of the frame, which is unacceptable for applications that require QoS.

The third drawback of APS mode is that the wake beacons of a pair of DEVs in APS mode are not synchronized. Thus, if the DEVs need to send occasional traffic to each other, the latency in sending the traffic can be quite large. For example, if one of the DEVs needs to communicate with the other DEV, it first needs to switch to ACTIVE mode and then request channel time from the PNC. At this point, the other APS DEV may have just finished the process of renewing its ATP and will not be listening to beacons for quite some time.

INTEL-1010
U.S. Patent 9,265,043

A DEV can use APS mode without first joining the APS set. Instead, the DEV simply uses the PM Mode Change command to switch to APS mode (see "Switching PM modes" on page 110). The method used to wake up a DEV in APS mode is the same as for DSPS. The requesting DEV sends a Channel Time Request command to the PNC, and the PNC sets the appropriate bit in the PCTM IE if the request will be granted. When the APS DEV responds to the PCTM IE with the PM Mode Change command, it can either accept the allocation by switching to ACTIVE mode or refuse the allocation by staying in APS mode.

CHANGING PICONET PARAMETERS

DEVs that are members of the piconet synchronize with the PNC based on the PNID, the BSID, the PHY channel, and the timing of the beacon. Whenever the PNC changes one of these key characteristics of the piconet, it uses the Piconet Parameter Change IE to communicate the upcoming change to the members of the piconet. The PNC is required to repeat this announcement in at least mMinBeaconInfoRepeat consecutive beacons. The parameter mMinBeaconInfoRepeat is used as a basis for other announcements and has a value of 4 in the standard. The significance of this number is that it is the same as the number of beacons that a DEV with a pseudo-static allocation can miss before before it must stop transmitting until it sees another beacon. This repetition ensures that DEVs that are the source of pseudo-static allocations will either hear at least one announcement when the PNC is changing the size of the superframe or the location of the beacon or that they will have to stop transmitting until the DEVs receive a beacon.

The Piconet Parameter Change IE indicates the parameter that the PNC will be changing and the beacon number of the superframe in which the change takes effect. The PNC is only allowed to change one piconet parameter at a time; i.e., it is not allowed to indicate a change to the superframe size while it is also indicating an impending channel change. In addition, the PNC is not allowed to change the locations of any pseudo-static CTAs once it is has indicated that it will change one of the piconet parameters.

During the parameter change, there may be DEVs in the piconet that are operating in a power-save mode. If there are any DEVs in PSPS mode, the PNC is required to place at least one occurrence of the Piconet Parameter Change IE in a system

INTEL-1010
U.S. Patent 9,265,043

wake beacon and in at least three consecutive beacons that follow the system wake beacon. All of the DEVs in PSPS mode are required to be awake to receive the system wake beacon, and if the PSPS DEVs miss it, they are required to be awake for subsequent beacon transmissions until a beacon is received. Although it is possible that a PSPS DEV might miss all four beacon announcements, the probability of this is pretty low.

The PNC does not have to align the piconet parameter change announcements with DSPS wake beacons, although the PNC could continue the announcements beyond the minimum of four repetitions so that they occur in DSPS wake beacons as well. It is not possible for the PNC to align the announcements with the wake times of APS DEVs because each APS DEV can wake up at a different time and that time is not known by the PNC. If a DEV in any of the three power-save modes hears the announcement, it is not required to switch to ACTIVE mode. The sleeping DEVs can calculate the effect of the change based on the information in the Piconet Parameter Change IE and adjust their settings such that when they next wake up, they will still be synchronized with the piconet. Thus, there is no reason to force the sleeping DEVs to ACTIVE mode when a piconet parameter changes. On the other hand, requiring the sleeping DEVs to switch to ACTIVE mode would increase the command traffic right before and right after the parameter change as well as needlessly increasing their power usage.

Beacon announcements

From time to time, the PNC is required to announce changes that occur in the piconet. Some of the changes apply to every DEV in the piconet, whereas others only apply to a single DEV. In general, the PNC is required to repeat these announcements so that most of the DEVs in the piconet will have the opportunity to hear the announcements. However, because these announcements are broadcast in nature, there is no guarantee that a DEV will receive the announcement. To handle this case, the content in these announcements can also be retrieved from the PNC using a command, usually the Probe Request command. Some of the beacon announcements are not repeated, but rather appear whenever they are needed. IEs that are put in the beacon on an "as needed" basis include the following:

- Application Specific

INTEL-1010
U.S. Patent 9,265,043

- Pending Channel Time Map (PCTM)
- PS Status
- Continued Wake Beacon (CWB)

Table 4–5 has a listing of the announcements that a PNC might make in a beacon for which there is a minimum number of times that they will be repeated. The table lists the announcements, the minimum repetition for the announcements, and if the announcement is intended for a single DEV or if it is intended for all of the DEVs in the piconet.

Table 4–5: Beacon announcements

Information element	Repetition	Intended for
DEV Association	mMinBeaconRepeatInfo	All DEVs
PNC Shutdown	mMinBeaconRepeatInfo	All DEVs
Piconet Parameter Change	mMinBeaconRepeatInfo	All DEVs
PNC handover	mMinBeaconRepeatInfo	All DEVs
CTA status	mMinBeaconRepeatInfo	DestID

The rules for beacon announcements are simple. All announcements are made in at least mMinBeaconRepeatInfo consecutive beacons beginning with any beacon. If the intended recipient of an IE is all DEVs, then if any DEV is in PSPS mode, the IE announcement is made in a system wake beacon and in at least mMinBeaconInfoRepeat-1 consecutive beacons following the system wake beacon.

If the intended recipient of an IE is a single DEV that is in SPS mode, the following rules apply:

- If the DEV is in PSPS mode, the IE announcement is made in a system wake beacon and in at least mMinBeaconInfoRepeat-1 consecutive beacons following the system wake beacon.
- If the DEV is in DSPS mode, the IE announcement is made in one of the DEV's DSPS set wake beacons and in at least mMinBeaconInfoRepeat-1 consecutive beacons following the DEV's DSPS set wake beacon.

INTEL-1010
U.S. Patent 9,265,043

A CTA Status IE is considered to be intended for all DEVs if the DestID contained in that IE is the BcstID or McstID. Otherwise the CTA Status IE is intended for the DEV defined by DestID.

There is a special exception for the PNC Shutdown IE because the request to turn off the PNC may have a timeout that is too short to support normal beacon announcements. The shutdown announcement is a courtesy to the DEVs in the piconet. From a DEV's point of view, the piconet may disappear at any time without a shutdown announcement as the DEV can move out of range of the PNC in its normal operation. Thus, the PNC will attempt to send shutdown information, but when it needs to stop, it will cease operation.

Despite the efforts of the PNC to communicate information to the DEVs in the piconet via the beacon announcements, some of the DEVs may miss that information. Because the 802.15.3 piconet operates in a dynamic wireless medium, it is likely that these announcements will be missed from time to time and so it is important that a method is provided for the DEVs to get any information that they missed. Table 4–6 provides a list of the beacon announcement IEs and an alternate method for retrieving the information in that IE.

Table 4–6: Alternative methods to retrieve beacon announcement IEs

Information element	Alternative method to get information
DEV Association	PNC Information Request command
PNC Shutdown	Beacons disappear and ATP expires
Piconet Parameter Change	New information is contained in every beacon
Application Specific	Vendor specific
Pending Channel Time map (PCTM)	IE is in every beacon as long as it is needed
PNC Handover	New PNC DEV address is in the new beacons
CTA Status	Probe Request command
PS Status	IE is in every beacon as long as it is needed

INTEL-1010
U.S. Patent 9,265,043

Dynamic channel selection

The wireless environment of 802.15.3 is dynamic and uncontrolled. As an unlicensed user, 802.15.3 DEVs need to be able to handle interference from incumbent users in the frequency band of operation. However, if the PNC determines that the characteristics of the current PHY channel are poor, it can search for a new channel for the piconet. The PNC has three methods by which it can determine the characteristics of the available PHY channels:

a) The PNC can scan all of the available PHY channels (see "PNC channel scanning" on page 150) to determine the level of interference.

b) The PNC can use the Remote Scan Request command (see "Remote scan" on page 152) to ask another DEV in the piconet to scan some or all of the PHY channels.

c) The PNC can use the Channel Status Request command (see "Channel status" on page 148) to ask the DEVs in the piconet to return statistics on the current connections in the piconet.

Although the current PHY uses frequency-based channelization, this method is applicable to other types of channelization, including code-based channelizations like CDMA. The PHY is responsible for mapping the concept of a channel number into the precise characteristics of the PHY channel.

Regardless of how the PNC arrives at the decision to change the PHY channel, the PNC must perform a PNC channel scan of the new channel before it moves the piconet. This last scan is important because the PNC is the center of the piconet, and so the PNC must determine that the new channel is sufficiently clear before it moves the piconet. Once the PNC has decided to change the piconet's PHY channel, it places the Piconet Parameter Change IE in the beacon with the new channel number and the beacon number of the first beacon that will be sent on the new PHY channel.

In the last superframe on the old channel, i.e., the superframe where the beacon number is one less than the beacon number in the Piconet Parameter Change IE, the PNC and the other DEVs will have to cease operations prior to the end of the superframe to ensure that they have enough time to switch channels. The PHY specification incudes a parameter that indicates the maximum time allowed for a channel change. In the case of the 2.4 GHz PHY, the DEVs are allowed a

INTEL-1010
U.S. Patent 9,265,043

maximum of 500 μs to switch channels. Thus, in the last superframe, the last 500 μs of CTAs will go unused as the DEVs switch to the new channel. If possible, the PNC should arrange the CTAs in the last superframe such that there are not any allocations during the time that the DEVs require to change channels.

If the parent PNC changes channels, the dependent PNCs will have to either change channels at the same time as the parent or cease operations as a piconet. If there is only one dependent piconet, it would be possible for that piconet to remain in the old channel, eventually becoming an independent piconet. However, there may be more than one dependent piconet that is a part of the parent piconet. In this case, it is possible that more than one dependent piconet could remain in the old channel and that they would begin to collide with each other if they remained in operation. To avoid this potential problem, IEEE Std 802.15.3 requires that dependent piconets that want to remain in the old channel first shutdown operations and then restart the piconet in the channel.

The frame exchange sequence for changing the PHY channel is illustrated in Figure 4–32.

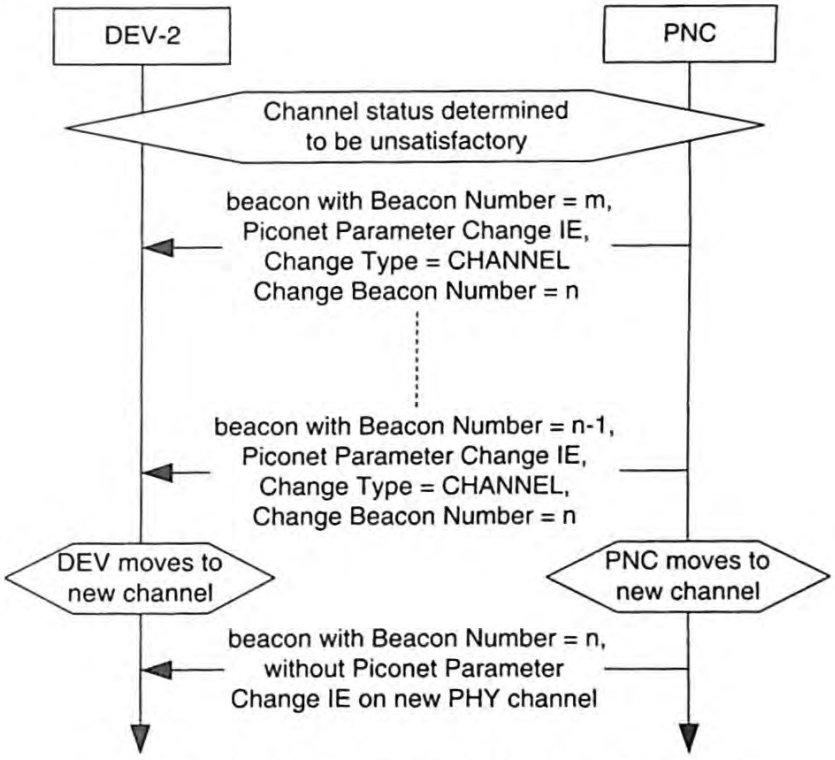


Figure 4–32: Changing the PHY channel

INTEL-1010
U.S. Patent 9,265,043

Changing the PNID or BSID

DEVs in the piconet identify the beacon by checking three values, the PNID, the BSID, and the DEV Address of the PNC. Only when all three match the values that the DEV is looking for can it be certain that it has found the beacon for its piconet. In the case of the DEV Address, the values are required to be unique by the IEEE registration authority committee (RAC). A DEV in the piconet finds out about a change in this parameter via the PNC Handover IE that is sent during the PNC handover process. The BSID and PNID, on the other hand, may not be unique. In the case of the PNID, the PNC that starts the piconet is supposed to pick a random value for the PNID. However, this does not guarantee that the piconet will not encounter another piconet with the same PNID.



The IEEE RAC is a subcommittee of the IEEE Standards Association Standards Board (IEEE-SASB) and its purpose is to assign unique numbers to support standards. Among other things, the RAC is responsible for assigning MAC address groups to companies. The RAC assigns a 3-octet (24-bit) prefix number to companies or organizations to use as an organizationally unique identifier (OUI). The company or organization then uniquely assigns the last 3 octets (for 48-bit MAC addresses) or the last 5 octets (for 64-bit MAC addresses) to products that they produce.

The problem of overlapping PNIDs is very important in 802.15.3 because the unique DEV address is not used in the addressing of frames. Thus, two piconets that are overlapping may have two different DEVs with the same DEVID. Each DEV checks not only for its DEVID as the DestID in the MAC header, but it is also required to ensure that the PNID matches the value that it expects. This prevents the two DEVs in different piconets from both trying to ACK the same frame or respond to the same command. Thus, the PNID is in each frame to help ensure the uniqueness of the 802.15.3 addressing scheme. However, this method does not work if the adjacent piconet also uses the same PNID. The first working group letter ballot pointed out this flaw, and the group developed techniques to enable the DEVs and the PNC to detect the presence of another piconet with the same PNID and to be able to change the PNID in an operating piconet.

INTEL-1010
U.S. Patent 9,265,043

If a DEV determines that there is another piconet within its operational area that is using the same PNID, that DEV is required to report that fact to the PNC. There are two ways that the DEV may determine that this is the case:

a) The DEV receives a beacon from a different PNC with the same PNID. The DEV will know that it is a different PNC by checking the PNC Address in the Piconet Synchronization Parameters field.

b) The DEV receives a frame with the current PNID where the SrcID or DestID is not a valid ID in the current piconet.

In the second case, the DEV needs to be careful that it does not report an overlapping piconet simply because it has not been able to keep up with the changes in the piconet membership.

If the DEV does detect an overlapping piconet with the same PNID as the DEV's current piconet, the DEV will use the Announce command to send the Overlapping PNID IE to the PNC to report this. The PNC will also be performing periodic scans of the current PHY channel as well as other PHY channels, and so it may detect an overlapping piconet with the same PNID as well. Finally, the Remote Scan Response command contains a list of all piconets that the DEV was able to find in its search of the PHY channels. The PNC can examine the Piconet Description Sets that are returned in this command to see if there are any other piconets nearby that are using the same PNID.

If the PNC determines, either by a report from a DEV or from its own scans, that there is an overlapping piconet in any of the valid PHY channels, it is required to change the PNID. Note that this requirement applies even to piconets that are currently occupying a PHY channel that is different from the current one. Although the overlapping piconet may currently be in a different PHY channel, it could change channels into the piconet's PHY channel. Therefore, the PNC must change the PNID to avoid the addressing problems that would occur if the overlapping piconet that is using the same PNID changed into the current channel.

Unless an overlapping piconet is found, the PNID remains constant for the life of the piconet. If the PNC shuts down the piconet without handing over control, it can reuse the PNID from the previous piconet. Otherwise, the PNC needs to randomly select a new PNID for the new piconet. When the PNC hands over

INTEL-1010
U.S. Patent 9,265,043

responsibility to a new PNC-capable DEV with the handover process, both the PNIDs and BSIDs remain the same.

Although the PNID is used in every frame, the BSID appears only in the beacon. The BSID is a text field of 6 to 32 ASCII characters that is intended to provide a user-friendly method to identify a specific piconet. If a DEV has lost track of the piconet for some time, it can regain synchronization with the piconet using the BSID. The PNID or the DEV address of the PNC may have changed in the meantime, but the BSID will likely remain constant. Using a text string instead of a hexadecimal number makes it possible for applications to display the names of piconets that are discovered in a manner that can be understood by a typical user, e.g., "Bob Jones' piconet" instead of a MAC address of 00-09-6B-50-3C-7A.

Because the BSID is used for scanning and by the user to select a piconet, the PNC is not required to change the BSID if it detects another piconet nearby that is using the same BSID. In the case of a home network, the user may wish for all of the piconets to use the same BSID. The PNC is allowed to change the BSID using the same procedure that is used to change the PNID. However, the PNC is not allowed to change both the PNID and the BSID at the same time.

The PNC changes the PNID or BSID using the Piconet Parameter Change IE with the Change Type field set to either "PNID" or "BSID" in the beacon. In the superframe indicated by the Change Beacon Number field, all of the DEVs in the piconet, including the PNC, will start using the new PNID or BSID. The announcement of this IE follows the same rules as other beacon announcements, including alignment with system wake beacons, if required.

Moving the beacon or changing the superframe duration

The PNC needs to be able to control the timing in the superframe, and so it is allowed to change the superframe duration or location of a beacon relative to a previous beacon. The PNC may change the superframe duration, as illustrated in Figure 4–33, to handle more traffic or to change the latency for allocations in the superframe.

INTEL-1010
U.S. Patent 9,265,043

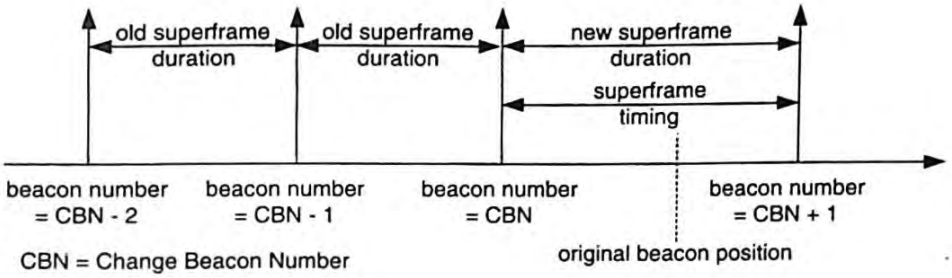


Figure 4–33: Changing the superframe duration

Likewise, the PNC may also wish to change the position of the superframe relative to previous superframes, as illustrated in Figure 4–34. This capability is required for dependent piconets because the parent PNC is allowed to move the CTA position for the dependent piconet. Thus, the dependent PNC needs to have a method by which it can inform DEVs in its piconet that the superframe will be changing position relative to a previous superframe. As the figure shows, the PNC is only allowed to delay the start of the next superframe, it cannot advance the start of a subsequent superframe. If the PNC were allowed to advance the start time of the following superframe, then the next superframe would overlap with the current superframe, potentially causing collisions with the last CTAs and the start of the next superframe. The PNC can achieve the same effect as advancing the superframe position by a small amount by setting the Superframe Timing value that is equal to the superframe duration minus the amount that the superframe position needs to move. Using only a positive value for the move also made the specification simpler in the draft.

The PNC notifies the DEVs in the piconet of a change in the superframe position or superframe duration by placing the Piconet Parameter Change IE in the beacon with the Change Type field set to either "MOVE" or "SIZE" and the Superframe Timing field set to the appropriate value. If the Change Type field is set to "MOVE," then the Superframe Timing field is the delay in the start of the superframe that has the beacon number indicated in the Change Beacon Number field. If the Change Type field is set to "SIZE," then the Superframe Timing field is the duration of the superframe that has the beacon number indicated in the Change Beacon Number field.

INTEL-1010
U.S. Patent 9,265,043

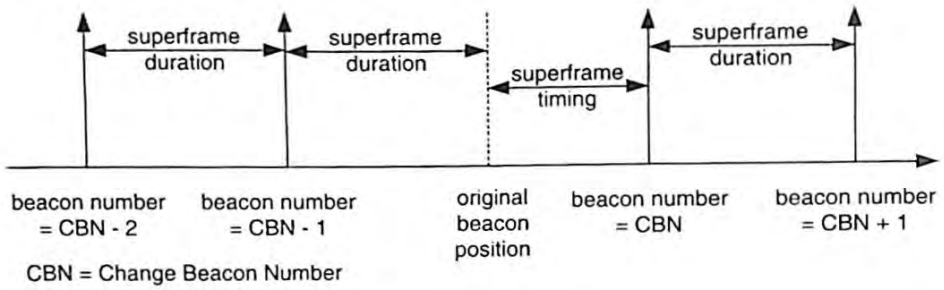


Figure 4–34: Changing the relative position of the superframe

A dependent PNC includes the Piconet Parameter Change IE in the beacon with the Change Type field set to "MOVE" and the Superframe Timing field set to zero when the parent PNC is handing over control of the piconet. During this procedure, the new PNC may move the dependent piconet's allocation when it takes over as the PNC, and the dependent PNC needs to inform its DEVs that a change is about to happen that may affect the timing in the piconet. The DEVs in the dependent piconet are all required to wait until they successfully receive a beacon from the dependent PNC following this change before they can transmit again. This rule is always in effect for DEVs with dynamic CTAs, so that this does not create a new requirement for those DEVs. However, DEVs in the dependent piconet with pseudo-static CTAs need to know that the location of their allocation may be changing as well.

FINDING INFORMATION

A key component of any protocol is that it provides a method for a participant to find out information about other participants in the network as well as the condition of the network. In 802.15.3, there are seven defined methods for sending or retrieving information about either the DEVs in the piconet or the current conditions in the piconet. These seven methods are as follows:

a) *Probe Request and Probe Response commands:* These commands are used to request IEs and to respond with the requested IEs.

b) *Announce command:* This command is used to send an IE to a DEV that did not request the IE.

INTEL-1010
U.S. Patent 9,265,043

c) *PNC Information Request and PNC Information commands:* These commands provide information about the capabilities of the DEVs that are members of the piconet.

d) *Channel Status Request and Channel Status Response commands:* These commands allow the DEVs to determine the link quality based on the number of lost frames.

e) *PNC channel scanning procedure:* The PNC searches all or a portion of the PHY channels to determine the best channel for the piconet.

f) *Remote Scan Request and Remote Scan Response command:* The DEVs in the piconet report the status of scanning a list of PHY channels.

g) *Piconet Services command and IE:* This command and IE provide support for service discovery. The DEVs report vendor-specific information about the higher layer capabilities that they provide.

Probe

The Probe Request and Probe Response commands are used to obtain an IE from the PNC or from another DEV in the piconet. In all but the last two drafts of the standard, a single command—the Probe command—took the place of the three commands now in the standard; Probe Request, Probe Response and Announce. This single command could be sent back and forth between two DEVs with no end in the frame exchange sequence, giving it a "ping pong" quality. This versatility made it almost impossible to accurately specify its behavior in the MLMEs and in the MSCs. The more that the task group worked with this "super command," the more that they disliked the way that it was defined. Finally, a proposal was presented that split the Probe command into the more manageable trio that is now used to request, respond and send IEs. A comparison of the description of the Probe command in previous versions of the draft with the descriptions of Probe Request, Probe Response, and Announce in the final version of the standard clearly show that the new commands are much simpler to describe and implement. This is most clearly demonstrated in the simplicity of the new MSCs versus the complex combination of MSCs in the older drafts which did not even completely describe the Probe command's behavior.

The procedure for requesting an IE from the PNC or another DEV in the piconet begins by sending a Probe Request command to a member of the piconet, as

INTEL-1010
U.S. Patent 9,265,043

shown in Figure 4–35. This command contains an Information Requested field and a Request Index field. The Information Requested field is either a bitmap that identifies the IEs or a binary representation of an IE's element ID. When the Information Requested field contains a bitmap, the DEV is able to request multiple IEs as long as their element ID is less than 31[25]. Note that the Vendor Specific IEs all have element IDs larger than 0x80 and so they cannot be requested using the bitmap. Instead, these IEs must be requested one at a time with separate Probe Request commands.

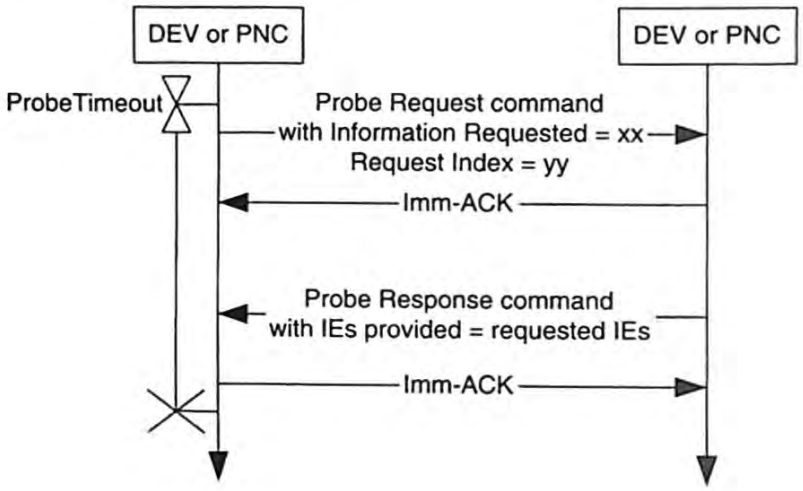


Figure 4–35: DEV using Probe Request command to obtain an IE

The Request Index field is used in the Probe Request command to indicate the specific IE that is requested when more than one would be appropriate. This field was introduced to enable a DEV to request a specific CTA Status IE. The PNC is able to provide CTA Status IEs for each of the allocated streams in the piconet. Thus, when a DEV requests the



The Request Index field is 2 octets although only a single octet is currently used and it has only been defined for a single purpose. The field was named generically and made larger so that it would support updates to the standard without changing the format of the command. In addition, the ASIE or vendor specific IEs may also use this field and it was felt that it was better to allow more room in case it was needed.

[25] There are currently only 16 IEs specified in the standard, but this may be increased in future revisions.

INTEL-1010
U.S. Patent 9,265,043

CTA Status IE from the PNC with a Probe Request command, the PNC needs to know if the DEV wants all of the possible CTA Status IEs or if it just wants information about a particular Stream Index. The Request Index field in this case contains the Stream Index for which the DEV is requesting the CTA Status IE. If least significant octet of the Request Index field is the BcstID, then the DEV is requesting that the PNC send all valid CTA Status IEs to the DEV.

When a DEV receives a Probe Request command, it first checks to see if it is allowed to respond to the request. The DEV is forbidden to respond to a request for some IEs. If the DEV is allowed to respond with any of the IEs that were requested, it sends a Probe Response command to the requesting DEV with the some or all of the requested IEs. The DEV is allowed to split its response over multiple Probe Response commands if the length of the IEs exceeds the maximum frame length or if the DEV does not want to send long Probe Response commands due to the channel characteristics. However, the Probe Response command cannot be fragmented.

Some of the IEs used in the standard are not intended to be sent or received from DEV to DEV or they may have validity for only a limited period of time, e.g., the CTA IE, Piconet Parameter Change IE or the PS Status IE. Therefore, the DEVs are forbidden to request certain IEs from other DEVs or from the PNC, depending on the type of IE and whether the entity making the request is a regular DEV or is the PNC. The rules for requesting IEs are summarized in Table 4–7.



When the now obsolete Probe command was being used, a DEV requested a CTA Status IE by first sending a CTA Status Request IE to the PNC with the stream index. The PNC then responded with the correct CTA Status IE(s). Because of this, it was not possible to directly request either the CTA Status IE or the CTA Status Request IE. Unfortunately, when the standard was updated to use the new Probe Request/Probe Response method, the tables that indicate the allowed requests and responses was not updated. Thus, there is an error in these tables in the standard; a DEV is allowed to request this IE from the PNC and the PNC is allowed to respond. The restrictions are correctly listed in these tables.

Similarly, not every request that a DEV receives in a probe command should receive a response. In this case, the rules are different depending on the source and destination of the request. For example, it makes sense for the PNC to request the Overlapping PNID IE from a DEV, but a DEV should not request it from the PNC.

INTEL-1010
U.S. Patent 9,265,043

When either a DEV or the PNC receives an IE request in a Probe Request command, it is either required to respond to the request, allowed to respond to the request, or forbidden to respond to the request. The rules for responding to a request for an IE in the Probe Request command are summarized in Table 4–8. Note that there are only three types of requests possible. The fourth, PNC requesting from the PNC, is not a possible interaction.

Table 4–7: Rules for requesting IEs in a Probe Request command

Information element	PNC allowed to request?	DEV allowed to request?
CTA	Shall not request	Shall not request
BSID	Shall not request	May request
Parent Piconet	Shall not request	May request
DEV Association	Shall not request	Shall not request
PNC Shutdown	Shall not request	Shall not request
Piconet Parameter Change	Shall not request	Shall not request
Application Specific	May request	May request
PCTM	Shall not request	May request
PNC Handover	Shall not request	Shall not request
CTA Status	Shall not request	May request
Capability	May request	May request
Transmit Power Parameters	May request	May request
PS Status	Shall not request	Shall not request
CWB	Shall not request	Shall not request
Overlapping PNID	May request	Shall not request
Piconet Services	May request	May request
Vendor Specific or Reserved	May request	May request

INTEL-1010
U.S. Patent 9,265,043

Table 4–8: Rules for responding to requests for IEs in a Probe Request command

Information element	DEV received request from DEV	DEV received request from PNC	PNC received request from DEV
CTA	Shall ignore	Shall ignore	Shall ignore
BSID	Shall ignore	Shall ignore	Shall respond
Parent Piconet	Shall ignore	Shall ignore	Shall respond
DEV Association	Shall ignore	Shall ignore	Shall ignore
PNC Shutdown	Shall ignore	Shall ignore	Shall ignore
Piconet Parameter Change	Shall ignore	Shall ignore	Shall ignore
Application Specific	May respond	May respond	May respond
PCTM	Shall ignore	Shall ignore	Shall respond
PNC Handover	Shall ignore	Shall ignore	Shall ignore
CTA Status	Shall ignore	Shall ignore	Shall respond
Capability Information	Shall respond	Shall respond	Shall respond
Transmit Power Parameters	Shall respond	Shall respond	Shall respond
PS Status	Shall ignore	Shall ignore	Shall ignore
CWB	Shall ignore	Shall ignore	Shall ignore
Overlapping PNID	Shall ignore	May respond	Shall ignore
Piconet Services	May respond	May respond	May respond
Vendor Specific or Reserved	May respond	May respond	May respond

Announce

The Announce command is used to send unsolicited information to another member of the piconet. Because the information was not requested, the sending DEV does not receive a response that the information was received, other than the Imm-ACK. The process of sending the Announce command is illustrated in Figure 4–36. The destination DEV is allowed to drop the incoming Announce command for various reasons, e.g., its internal buffers are full, it is too busy, etc.,

INTEL-1010
U.S. Patent 9,265,043

even if it ACKed the frame. Thus, the sending DEV may need to use another method, not explicitly specified in the standard, to verify that the IE was received and processed. The only IE where this is really important is for the Overlapping PNID IE. In this case, the DEV will know that the PNC received this IE correctly when it receives a Piconet Parameter Change IE in the beacon that is changing the PNID.

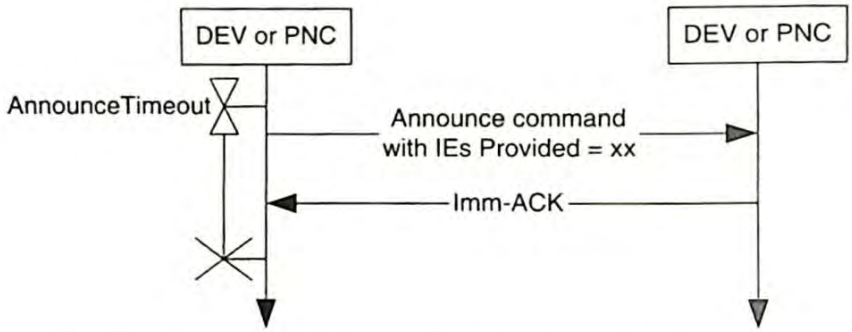


Figure 4–36: DEV using Announce command to send information

Not all of the IEs are allowed to be sent unrequested to another DEV in the piconet with the Announce command. For example, the CTA IE is sent in every beacon, but it is not allowed to be sent in the Announce command. The CTA IE is meaningful only at the beginning of the superframe when it is sent in the beacon. If it is sent in an Announce command during the superframe, it would be possible that the time in the CTA IE had already passed. Even if the allocated time in the CTA was after the reception of the Announce command, the DEV might not have enough time to prepare to use the CTA. For other IEs, it only makes sense if either the PNC (e.g., PS Status IE) or the DEV (e.g., the Overlapping PNID IE) sends it. The rules for sending IEs to other DEVs in the piconet are listed in Table 4–9.

Another use of the Announce command is to send it to a DEV to determine if it is still in the piconet. To do this, the DEV sends an Announce command with an empty IE provided field to another DEV in the piconet. The Imm-ACK response by that DEV verifies that it is still in the piconet and is within the range of the requesting DEV. If the target DEV is in a power-save mode, the originator will need to send the command in one of its wake beacons.

INTEL-1010
U.S. Patent 9,265,043

Table 4–9: Rules for sending unrequested IEs in an Announce command

Information element	PNC allowed to send?	DEV allowed to send?
CTA	Shall not send	Shall not send
BSID	Shall not send	Shall not send
Parent Piconet	Shall not send	Shall not send
DEV Association	May send	Shall not send
PNC Shutdown	May send	Shall not send
Piconet Parameter Change	May send	Shall not send
Application Specific	May send	May send
PCTM	May send	Shall not send
PNC Handover	May send	Shall not send
CTA Status	May send	Shall not send
Capability Information	May send	May send
Transmit Power Parameters	May send	May send
PS Status	May send	Shall not send
CWB	Shall not send	Shall not send
Overlapping PNID	Shall not send	May send
Piconet Services	May send	May send
Vendor-Specific or Reserved	May send	May send

PNC Information

When a DEV associates with a PNC, it passes a set of information that identifies it and indicates its capabilities as both a DEV and as a PNC, if it is PNC capable. The PNC needs to distribute this



In a non-secure piconet, a DEV becomes a member when it completes the association process. In a secure piconet, however, a DEV becomes a member once it has received a management key from the PNC. The DEV gets this key using a process that is outside of the scope of the standard. The other DEVs in the piconet will be informed that a new DEV is a member of the piconet when they receive a PNC Information command from the PNC with the new DEV listed as a member.

INTEL-1010
U.S. Patent 9,265,043

information to the piconet and needs to tell the new DEV about the capabilities and identities of the other DEVs in the piconet. To accomplish this, the PNC uses the PNC Information command to broadcast the information about the DEVs in the piconet whenever a new DEV becomes a member of the piconet. The broadcast PNC Information command contains the DEVID, the DEV address, the Overall Capabilities field, the DEV's ATP, and the DEV's requested System Wake Beacon Interval for each of the DEVs in the piconet. In addition, each entry for a DEV indicates if the DEV is a member of the piconet or if it is only associated with the piconet. The PNC also uses this command to send information about the DEVs in the piconet to the new PNC in the PNC handover process (see "Handing over control" on page 66).

For the other DEVs in the piconet, the PNC Information command carries some key information. The most important information is the pairing of the DEVID and DEV address. Because a DEV might join the piconet multiple times, it could end up using different DEVIDs. The other DEVs in the piconet will use the DEV address associated with the DEVID to determine the DEVs with which they need to communicate. Similarly, the higher-layer protocols will identify the DEVs by the unique MAC address, as the DEVID is only used in the MAC layer.

There are also two PHY-related characteristics that are contained in the overall capabilities field that is a part of a DEV Info field in the PNC Information command. The first characteristic is the data rates that are supported by the DEV. The PHY may have one or more mandatory data rates, but it may also have some optional data rates. If the DEV needs to use one of the optional data rates, then it must first determine that the destination supports this data rate using the information in the DEV Capabilities field. The second characteristic is the preferred fragment size. This field indicates to the other DEVs in the piconet the fragment size that this DEV prefers for incoming data frames.

The Overall Capabilities field included for each DEV in the PNC Information command also contains three pieces of information that can be used to help schedule a DEV's transmissions. When a DEV associates, it has the option to indicate that when it is in ACTIVE mode, it will be listening for frames addressed to it during certain portions of the superframe. Three options are available in the Overall Capabilities field:

INTEL-1010
U.S. Patent 9,265,043

- *Always AWAKE*—The DEV will be listening during the entire superframe for frames addressed to it.
- *Listen to Source*—The DEV will listen during all CTAs where the SrcID is the DEVID of a DEV which it is already the target of a CTA. For example, if DEV-1 sets the Listen to Source bit to be true and there is a CTA assigned with DEV-2 as the source and DEV-1 as the destination, DEV-1 will listen to all CTAs with DEV-2's DEVID as the SrcID, even if DEV-1 is not the destination.
- *Listen to Multicast*—The DEV will be listening to all Multicast CTAs, regardless of the SrcID or Stream Index.

The purpose of these three indications is that they make it possible for the source DEV to use all of its collective channel time to handle variable transmission demands to various destinations. For example, some video encodings result in a variable bit rate stream. If a DEV was sourcing three different video streams, it could request CTAs that were only large enough for the average bit rate of each of the three streams. If the destinations were listening to all of the source DEV's allocations, the source would be able balance the load among the three video streams, using left over time from a stream that was currently operating at a low data-rate to handle extra traffic from a stream that was currently operating at a high data-rate. The standard does not specify how a DEV might schedule the data for this, but by providing a method by which DEVs can indicate when they will listen, an implementation is given the opportunity to take advantage of this to improve throughput. One consequence of stating that the DEV will be listening more often than it needs to is that the DEV will use more power because its receiver will be on more often. The implementer will need to weigh the throughput advantages against the increase in power usage based on the target application and usage of the DEV.

Because a broadcast command is not ACKed, the PNC cannot guarantee that every DEV in the piconet will be able to receive the PNC Information command. If a DEV needs the information about either a single DEV or about all of the DEVs in the piconet, it sends the PNC Information Request command to the PNC. If the requesting DEV needs the information for a single DEV, it sends the request with the Queried DEVID field set to the DEVID about which it is seeking information. A DEV might send this request if it sees a CTA with a new DEVID or receives a frame from a DEVID that it does not recognize. When the DEV

INTEL-1010
U.S. Patent 9,265,043

determines that there is a DEVID for which it does not have the appropriate information, it will send the request to the PNC to get this information. If the DEV has determined it is out of sync with the membership list of the piconet, it sets the Queried DEVID field to the BcstID. The PNC will recognize this as a request for the entire list of associated DEVs, including the fields corresponding to the PNCID and the NbrIDs, if any neighbor PNCs are associated with the piconet. A new DEV that joins the piconet but misses the broadcast of this command would also request the entire list so that it will know the membership of the piconet as well as the relevant DEV capabilities. The frame exchange involved in requesting and receiving information from the PNC using the PNC Information Request and PNC Information command is illustrated in Figure 4–37.

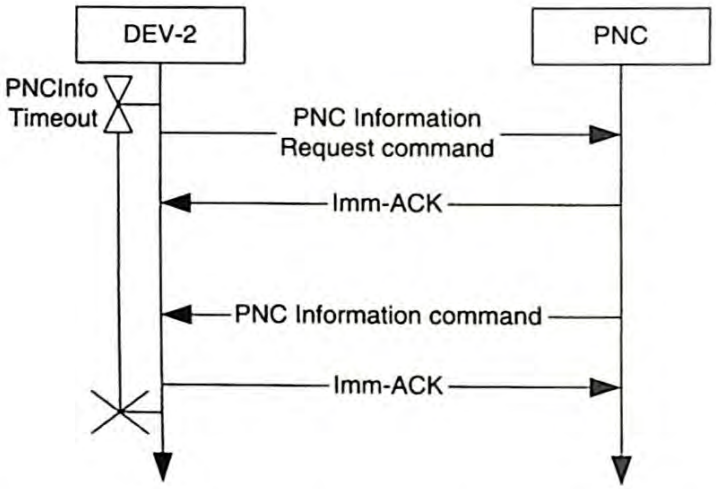


Figure 4–37: Requesting and receiving DEV related information from the PNC using the PNC Information Request and PNC Information commands

Channel status

In order to maintain the QoS requirements for the higher layers, the MAC and PHY need to be able to determine the quality of the link between two DEVs. The link quality affects the QoS in two ways. First, the source DEV needs to select a data rate that will have the highest throughput. If using a higher data rate increases the FER, then the DEV may have to use a lower PHY data rate to improve the

INTEL-1010
U.S. Patent 9,265,043

FER. Second, the DEV will use the data rate information along with the FER and the characteristics of the data to calculate the amount of time required in each superframe to support the data throughput required by the application.

The DEV can determine some information about the FER and, hence, the link quality by keeping track of the number of retransmissions required for frames that it sends to the destination. That gives part of the picture, but it does not include information about the number of times that the destination DEV ACKed the frame but where the source DEV was unable to hear the ACK. The Channel Status Request command provides a method for the originator DEV to request link quality information from the target DEV. The target DEV responds to the originator with the statistics of the number of frames it attempted to transmit, the number of frames it received, the number of those that were in error, and the number of no-ACK frames that were missed. The target DEV also indicates in the Measurement Window Size field the length of time over which this information was gathered.

In the totals reported in the Channel Status Response command, the target DEV will only count directed frames that were sent between the two DEVs. The DEV will count a frame as having been received in error if the HCS validation passes but the FCS validation fails. The DEV will keep track of missing no-ACK frames using the gaps in the MSDU numbers and report this number in the RX Frame Loss Count field. Because asynchronous traffic shares the same MSDU numbering among multiple destinations, the RX Frame Loss Count field only applies to isochronous streams.

The frame exchange sequence for determining the channel status between two DEVs is illustrated in Figure 4–38.

The PNC can also use the Channel Status Request command to collect information from all of the DEVs in the piconet. The PNC would use this information to help determine the quality of the current channel. To do this, the PNC sends the Channel Status Request command as a broadcast frame, i.e., with DestID set to the BcstID and the ACK Policy field set to no-ACK. DEVs that correctly receive this command will respond to the PNC with a directed Channel Status Response command when they get an opportunity to send commands to the PNC. The DEVs will report the aggregate status for all streams that have that DEV as the destination. If the PNC is not using the CAP for commands, then it

INTEL-1010
U.S. Patent 9,265,043

will need to allocate MCTAs to each of the DEVs in the piconet to allow them time to respond to the broadcast request. Not all of the DEVs in the piconet will receive this command, and because it is sent with no-ACK policy, the PNC will not know which DEVs were unable to receive the command until it begins to receive the responses. In particular, DEVs in a power-save mode will likely not receive the message. However, because these DEVs are saving power, it is also likely that their traffic is light and the statistics would not be as meaningful as those from ACTIVE mode DEVs.

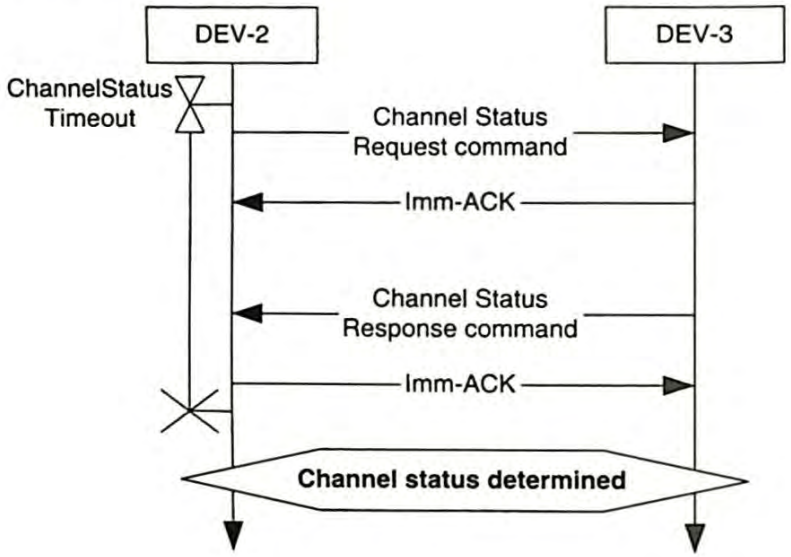


Figure 4–38: Requesting channel status information from another DEV

PNC channel scanning

In addition to maintaining the timing and granting channel time to DEVs, the PNC also needs to make sure that the piconet is operating in the best possible channel. Only the PNC can make the decision to change the channel, and so the PNC needs to have as much information as possible about not only the current channel, but also about other channels that it could potentially use. Every PNC-capable DEV is required to be able to scan the PHY channels to determine the ones that already have piconets before it can start its own piconet. The PNC channel scanning procedure builds on this ability to give the PNC channel status information while the piconet is in operation.

INTEL-1010
U.S. Patent 9,265,043

The PNC, or any DEV for that matter, is always allowed to use time when it is not required to be receiving or transmitting to passively scan any PHY channel. The PNC should periodically scan the current channel during quiet times in the superframe to detect overlapping piconets or other sources of interference. The PNC can create "quiet" time in the superframe either by placing a specific allocation in the beacon with the SrcID and DestIDs set to the PNCID or by leaving gaps in between CTAs in the superframe. However, if the PNC wants to scan other channels with this procedure, it needs to stop sending the beacon for one or more superframes. The PNC stops the traffic in the current piconet because the energy from transmissions in the current channel in general will be nonzero in other PHY channels. If the piconet continues operation, the PNC may consider a DEV that is in close proximity to the PNC as interference from another piconet. The PNC could potentially not select that channel even though it was empty.

Although most of the DEVs in the piconet will cease transmitting when the beacon stops, DEVs with pseudo-static allocations will continue operation. The PNC can handle this issue in one of three ways:

a) The PNC can place the Piconet Parameter Change IE in the beacon with the Operation Type Set to "MOVE" and the Superframe Duration field set to zero. This will cause DEVs with pseudo-static allocations to stop transmitting after the Change Countdown field reaches zero. The DEVs will restart transmissions when the PNC begins sending beacons again.
b) The PNC can keep track of the times allocated as pseudo-static CTAs and ignore interference data collected during these times.
c) The PNC can stop sending beacons for at least mMaxLostBeacons before collecting the channel status information.

Once the PNC has scanned the alternative channels, it returns to the current channel and begins sending the beacon again at the time when a beacon would have been expected. The PNC will increment the beacon number so that it takes on the value that it would have been if the PNC had not stopped sending the beacon. Having completed the PNC channel scanning procedure, the PNC has four options:

a) Remain in the current channel.
b) Change the piconet channel (see "Dynamic channel selection" on page 132) to a channel that the PNC scanned.

INTEL-1010
U.S. Patent 9,265,043

c) Change the maximum TX power allowed in the piconet (see "Transmit power control" on page 157).

d) Perform another, unspecified, action.

The final option was provided so that the standard would not unnecessarily restrict implementers in developing new techniques for handling interference in the PHY channel. The algorithm for determining when the PNC will change the channel is outside of the scope of the standard.

The PNC is required to perform the PNC channel scanning procedure before changing the channels because it is the center of the piconet. The PNC may find through the Remote Scan Request/Remote Scan Response commands (see "Remote scan" below) that another channel is best for most of the DEVs in the piconet. However, if that channel is not also a good one for the PNC, it will not matter that it is a good channel for the other DEVs in the piconet. The PNC must be able to send and receive commands reliably in order to act as the coordinator of the piconet. Thus, in the end, the PNC must confirm with its own scanning that the new channel is better for piconet operations than the current one.

Remote scan

Although it is of paramount importance that the PHY channel for the piconet is relatively quiet from the PNC's point of view, it is also important for the PNC to be able to determine the characteristics of the channel at points in the piconet other than where it is. There may be two channels where the location near the PNC is relatively quiet, but one channel may have an interferer that is relatively close to DEVs that are near the outside edge of the piconet's operational area. The PNC may not be able to detect this interference even though it is causing problems with traffic for those DEVs. Using another DEV to perform the scan also allows the PNC to continue the piconet operations uninterrupted while still receiving information about potential sources of interference in the current channel and the status of other PHY channels.

The PNC uses the Remote Scan Request command to ask a DEV to perform a scan of one or more PHY channels. The Remote Scan Request command contains a list of the channels that the PNC is requesting that the DEV scan. The PNC may also place quiet time in the beacon, as described in "PNC channel scanning" on

INTEL-1010
U.S. Patent 9,265,043

page 150, for the DEV to perform the channel scan. If the DEV accepts the request, it uses the normal channel scan procedure that it would use to find a piconet to join. The DEV will keep track of all of the piconets that it finds and records pertinent information about them. The DEV will pass back the PNID of the piconet that it found and the channel index where it found the information. If the DEV found only a frame from the other piconet, this is the only information that it can report. If the DEV receives a beacon from another piconet, it will also report the DEV address of the PNC, the BSID of the piconet, and the Parent Piconet IE, if it is present in the beacon. The DEV will indicate if it found either a beacon or non-beacon frame from another piconet and if the piconet is a dependent piconet.

In addition to reporting all of the piconets that the DEV found, the responding DEV will also rate the channels that it scanned for interference, listing them from best (least interference) to worst (most interference). The DEV does not respond with an absolute measure of the interference nor does it give a relative rating or interference, i.e., that channel 4 is three times as bad as channel 2. An absolute measurement of interference is difficult to define. The DEV would need to identify the type of interference it found; i.e., is it wideband or narrowband? Then it would need to report the percentage of time that the interference appeared. Finally, the PHY would need to be calibrated so that it gave relatively accurate measurements for the interference. Instead of creating all of these requirements, which would make it difficult to develop low-cost implementations, the standard requires only that the DEVs give an ordered rating using whatever criteria are appropriate for that DEV.

When the DEV has completed the scan, it will respond to the PNC with the Remote Scan Response command containing a listing of the channels that it scanned, the relative ranking of the channels for interference, and the list of all of the piconets that it found. The PNC is then able to use this information in determining if a channel change is required and to help in selecting the new channel if it needs to change channels.

Because channel scanning can take a long period of time, the DEVs are allowed to refuse the request to scan the channels for the PNC. If the DEV is not going to perform the channel scan, it responds to the PNC with the Remote Scan Response command with a Reason Code field set to "Request denied." If for some reason

INTEL-1010
U.S. Patent 9,265,043

the PNC requested a scan of a channel that is not valid, the DEV will respond with the Remote Scan Response command with the Reason Code field set to "Invalid channel requested." The frame exchange sequence for the PNC to request a remote channel scan and for the DEV to respond with the information is illustrated in Figure 4–39.

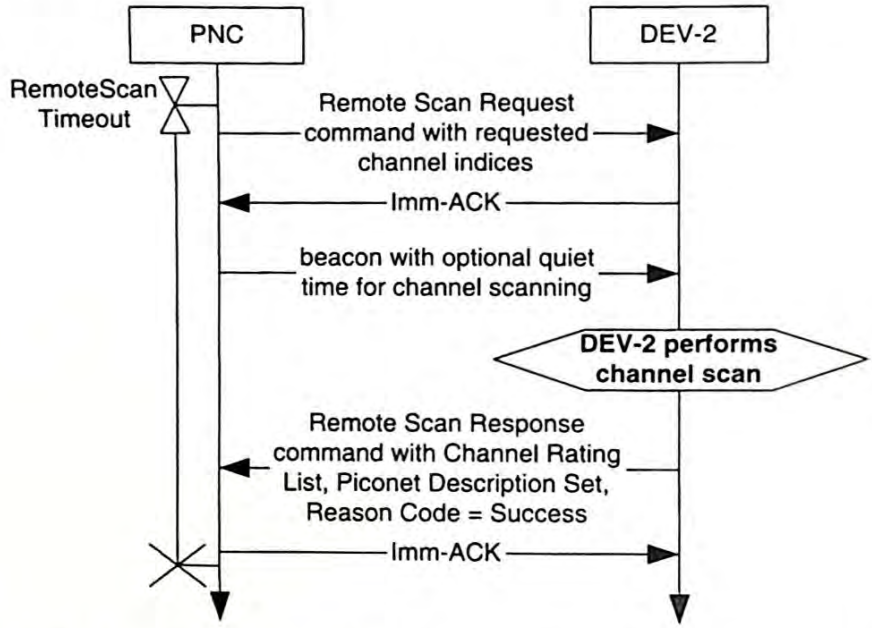


Figure 4–39: PNC requesting that a DEV perform a remote scan

A DEV may also send an unsolicited Remote Scan Response command to the PNC if it is receiving a beacon from another piconet. The DEV will not report any piconet that it determines to be a child or neighbor of the current piconet. It is possible, however, that the DEV will mistakenly report a child or neighbor piconet as an overlapping piconet. It is up to the PNC to determine which reports that it receives from the DEVs in the piconet are from overlapping piconets as well as the action required to handle any interference issues.

Piconet services

From a user's point of view, the most important characteristic of a DEV in the piconet is the application level services that the DEV provides. For example, when a video camera finds a piconet, it will want to know if any of the DEVs have

INTEL-1010
U.S. Patent 9,265,043

nonvolatile storage capabilities, e.g., a hard-drive, DVD writer, etc., or display capabilities, e.g., a flat-panel display or television. The ability of one entity to determine the application level characteristics of another entity is referred to as service discovery, and the process to determine this information is a service discovery protocol. Because application level information is outside the scope of a MAC/PHY standard, IEEE Std 802.15.3 cannot provide a complete specification of a service discovery protocol. There are higher layer protocols that do support service discovery, e.g., IEEE Std 1394 [B17], [B18], Jini™, or Universal Plug and Play™, so it is not necessary for 802.15.3 to provide another incompatible definition.

However, it is important for DEVs to be able to determine very early on the capabilities that are offered in a piconet. Ideally, the DEV would be able to determine these by passive scanning, but continually broadcasting these capabilities would needlessly waste bandwidth. Instead, IEEE Std 802.15.3 provides a method in the association process whereby an associating DEV can request the broadcast of the collected capabilities in the piconet. The goal is that within a very short period of time, ideally less than 1 second, a DEV can find a piconet, join it, and determine the application level services that are available. As an example, a person carrying a digital camera should be able to walk in the vicinity of a photo finishing kiosk and, within 1 second, find an icon for that kiosk on the camera's display. The user could then select print, select the kiosk, and the pictures would be transferred quickly to the kiosk for printing. The ability to quickly provide useful information to the user has always been a key criteria in the development of IEEE Std 802.15.3.

If an associating DEV requests the piconet services information in the Association Request command, the PNC will broadcast the Piconet Services command with all of the Piconet Services IEs that the PNC has collected from DEVs in the piconet. Because IEEE Std 802.15.3 does not define the format of this data, each IE includes an organizationally unique identifier (OUI) to enable the DEV that receives this command to determine if it can understand the contents and to determine how to process the contents if the DEV does support the OUI. If the PNC supports the Piconet Services command, it only has to store the IEs. The PNC is not required to be able to understand any of IEs that it stores and it can set a limit on the number or total size of the information that it will store.

INTEL-1010
U.S. Patent 9,265,043

Even though the PNC may not understand the contents of the IEs, this still provides a very useful service to the DEVs in the piconet. For example, suppose a DEV made by Manufacturer A joins a piconet with a PNC made by Manufacturer B that supports the Piconet Services command but does not support the decoding of Piconet Services IEs from Manufacturer A. When the DEV associates and requests the list of the piconet services, the command might contain three IEs from DEVs made by Manufacturer A that list services that the DEV needs. In this case, the DEV would be able to quickly, i.e., in less than 1 second, find out that a piconet offered an important service that it needed, e.g., printing, storage, WAN connectivity, and so on.

The capabilities of a DEV may be a security concern. For example, if the PNC in a home network always responded to a request for the Piconet Services by listing the number of TVs, VCRs, set-top boxes, and so on, it would be providing an inventory to potential thieves. On the other hand, some DEVs will always want to broadcast their services because the owners use them for making money. Examples of these types of entities would include photo-printing kiosks, public internet access hot-spots, and so on. The standard allows the DEV to refuse to broadcast this information either as a DEV or as the PNC. This allows the user to control the level of openness provided by these commands and IEs.

If a DEV wants to add its services to the list for the piconet, the DEV sends the Piconet Services IE to the PNC in an Announce command. If the PNC supports storing the Piconet Services IEs and it has storage space for the IE, it will save the IE for when the next DEV requests the information. The PNC will also send the Piconet Services IE to all the DEVs in a broadcast Announce command so that the existing DEVs in the piconet will see the services offered by the new DEV.

If a DEV sends a Probe Request command to the PNC requesting the Piconet Services IE, the PNC will respond to the DEV with Probe Response commands that contain the Piconet Services IEs for all of the DEVs in the piconet. If a DEV did not send a Piconet Services IE to the PNC, the PNC will place a Piconet Services IE in the Probe Response command with the DEV's DEVID and a zero Vendor OUI field and zero length Piconet Services field. If the PNC did not have sufficient space to store the Piconet Services IE submitted by the DEV, it will put a Piconet Services IE in the Probe Response command with only the DEV's DEVID; i.e., the IE will have length 1. Thus, the requesting DEV will receive an

INTEL-1010
U.S. Patent 9,265,043

IE for every DEV in the piconet, and it will know if any are missing in the response as well as knowing when the response is complete.

A DEV may also request a Piconet Services IE from another DEV in the piconet using the Probe Request command. If the DEV supports the Piconet Services IE and its current policy allows the broadcast of this information, the DEV will respond by sending its Piconet Services IE in the Probe Response command. Otherwise, the target DEV will respond with Probe Response command that contains a zero length Piconet Services IE which is an IE with only the Element ID and Length fields.

OTHER CAPABILITIES

IEEE Std 802.15.3 also provides a variety of capabilities that enhance the performance of the standard. This section describes how the PNC and DEVs can control the transmit power in the piconet as well as the rules for handling DEVs that support more than one data rate.

Transmit power control

IEEE Std 802.15.3 provides two types of transmit power control: a piconet-wid limit on transmit power and a negotiated level for DEV-to-DEV communications. The transmitter power used in the beacon and the CAP help to determine the physical size of the piconet. In order to constrain the size of the piconet, the PNC may set a maximum transmitter power that applies to the beacon, the CAP, and any directed MCTAs. The PNC is allowed to set power level to any value as long as it is not less than pMinTPCLevel, which is a PHY-dependent parameter. For the 2.4 GHz PHY, this parameter has a value of 0 dBm, which the lowest TX power level that a compliant DEV is required to support for this PHY. The PNC communicates the maximum TX power level to the DEVs in the Max TX Power Level field in the beacon. The DEVs have up to the 10*mMaxLostBeacons (= 40) superframes after receiving a beacon with a change to this field to comply with the new power level. The DEV is required to set its TX power to a level less than the value in this field. If the PNC is not going to put a limit on the maximum transmit power, it sets this field to its maximum value of 0x7F,[26] which corresponds to

[26] The field has units of dBm and is encoded in two's complement notation, so it has a minimum value of 0x80 or −128 dBm and a maximum value of 0xFE or +127 dBm.

INTEL-1010
U.S. Patent 9,265,043

+127 dBm (which will likely exceed the realistic transmit power of any imaginable PHY).

In addition to the piconet-wide limit on transmit power, DEVs that are exchanging data in a CTA can also negotiate a change in the transmit power level. Because this occurs in a specific CTA, a change in the transmit power will not adversely affect the other DEVs. This is not true during a CSMA/CA period like the 802.15.3 CAP or in an 802.11 WPAN. During a CSMA/CA period, all of the participants need to transmit with as much power as possible. If one of the participants was to decrease its transmit power, then other DEVs in the piconet would not sense the frame transmission and therefore would not suspend the backoff counter, leading to more collisions. Because a CTA is a TDMA allocation, the DEVs using the allocation can negotiate a lower transmit power to reduce interference and save power without adversely affecting the network or their throughput.

A DEV requests a decrease or increase in another DEVs transmit power by sending the Transmit Power Change command to the other DEV. This command includes a field that specifies a positive or negative change in decibels to the other DEV's transmit power. The field is encoded in two's complement notation, which allows a range of requested changes from –128 dB to +127 dB[27]. Because the requesting DEV would not know the current transmit power of the remote DEV, it simply asks for a change from whatever the current setting is. The actual transmit power that the other DEV is using is not relevant to the receiver; only the level of the received power relative to the receiving DEV's sensitivity is important. Thus, sending requested changes to the remote DEV is a simple and effective way to communicate a request for a transmit power change.

When a DEV receives a request for a transmit power change, the DEV will set its power level as close as possible to the requested change. The DEV will use this new power level for all CTAs that are assigned to the two DEVs. The message exchange for negotiating a different power level between two DEVs is illustrated in Figure 4–40.

[27] This range should be sufficient for any realistic PHY.

INTEL-1010
U.S. Patent 9,265,043

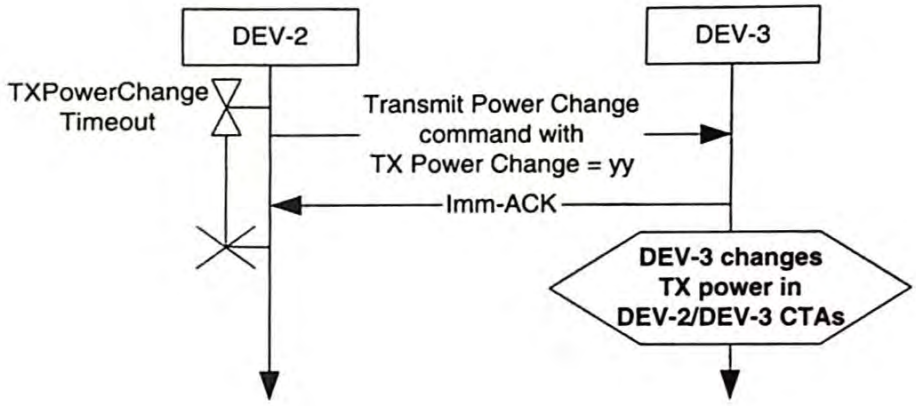


Figure 4–40: Requesting a change in a DEV's transmit power

Multirate capabilities

IEEE Std 802.15.3 was developed to support PHYs with multiple data rates where some of the data rates are optional. In the case of the 2.4 GHz PHY, there is one mandatory or base rate, 22 Mb/s, and four optional rates, 11, 33, 44, and 55 Mb/s. Because not all DEVs support the optional rates, broadcast and any traffic where the supported data rates of the DEV are unknown have to be sent at the base rate. For example, the Association Request and Association Response commands must be sent at the base rate because the DEV may not know the rates that the PNC supports and the PNC may not know the rates that the associating DEV supports. Once a DEV has joined the piconet, it can find out the supported data rates of another DEV in the piconet by using any of the following methods:

a) Check the DEV capabilities field for the DEV in the broadcast PNC Information command.

b) Send a Probe Request command to the DEV (at the base rate) requesting that the DEV send the Capability IE.

c) Send the PNC Information Request command to the PNC to request the record of the DEV.

Once the DEV has determined the data rates supported by the other DEV, it can use any one of the supported rates in directed communications with that DEV. The PNC's supported rates can be obtained either by examining the DEV Info field for

INTEL-1010
U.S. Patent 9,265,043

the PNCID or by examining the DEV Info field that corresponds to the PNC's DEV personality by searching for the PNC's DEV address.

All ACKs are sent at the same rate as the frame that requested the ACK. In the case of the 2.4 GHz PHY, this only changes the ACK for an 11 Mb/s frame. In all other cases, the ACK is the same for all data rates because it only includes the MAC header, which is sent with the 22 Mb/s mode. In the 11 Mb/s mode, however, the MAC header field is sent twice, once at the base rate of 22 Mb/s and once at the 11 Mb/s rate. If the source DEV is using Dly-ACK, all of the frames in the burst must use the same data rate and the Dly-ACK frame is also sent at that rate. The rules for using a data rate are summarized in Table 4–10.

Table 4–10: Allowed PHY data rates for each frame type

Type of frame	Allowed PHY rates
All broadcast and multicast addressed frames (including the beacon and commands)	Base rate
Imm-ACK	Same rate as the frame that is being ACKed
Dly-ACK	Same rate as the last frame of the burst being ACKed
Association Request command	Base rate
Association Response command	Base rate
Disassociation Request command	Base rate
Directed command Frame	Any rate supported by both the source and destination
Directed Data Frame	Any rate supported by both the source and destination

EXTENSIBILITY OF THE STANDARD

IEEE Std 802.15.3 was developed with the understanding that it would need to be both flexible and extensible. To accomplish this goal, the standard provides specific IEs and commands that are reserved for vendor-specific extensions that are either outside of the scope of the standard or are yet to be defined. In the case of 802.11, the standard reserved all of the unused IE numbers for future use in the

INTEL-1010
U.S. Patent 9,265,043

standard. However, the vendors needed to extend the standard with additional functionality, and so they began to use some of the reserved IE numbers. As IEEE Std 802.11 began to use more of the numbers to support enhancements to the standard, it increased the possibility of using a number already in use by a specific vendor.

IEEE Std 802.15.3 has explicitly called out a set of IE element IDs and command types that are reserved for vendor-specific use. In addition, IEEE Std 802.15.3 requires that the vendors identify these elements in standardized manner using a unique identifier to prevent interoperability problems with DEVs made by different manufacturers. Each Vendor Specific command and Vendor Specific IE, as well as the application specific information element (ASIE), all have an OUI field following the length field of either the command or IE. The OUI is defined to be the organizationally unique identifier issued by the IEEE RAC. Every manufacturer will be required to get an OUI in order to issue DEV addresses for the products that they create. They can use the same OUI to provide enhanced functionality for DEVs that understand these IEs or commands.

The inclusion of the OUI is necessary to prevent interoperability problems with other DEVs that are also using the Vendor Specific commands. Without it, a DEV might think that a Vendor Specific command that is supposed to say "respond if you are a printer" instead says "begin transmitting continuously for test mode." By using an OUI, the vendors will ensure that there is no confusion in the commands and IEs. If not, that vendor will experience a high return rate of their products, which will quickly end their participation in the market.

The first element that was proposed to provide extensibility to the standard was the ASIE. The ASIE was introduced to enable an implementation to place information in the beacon to be used by DEVs that supported the features offered by the PNC. The PNC can put multiple ASIEs in the beacon, and they can be placed either in the beacon frame or in the beacon extension. The contents of the ASIE are specific to the OUI and are outside the scope of the standard.

The standard has also set aside a block of element IDs for IEs and command types for commands. For IEs, the element IDs of 0x80–0xFF are reserved for vendor-specific IEs and will never be used by the standard (the 122 still available for the standard should be sufficient). In the case of vendor-specific commands, a modest 65,000+ command types are available for use as vendor-specific commands. The

INTEL-1010
U.S. Patent 9,265,043

command type is two octets, and the standard has only used or reserved for use the first 256. Again, this should be sufficient for the foreseeable future for 802.15.3. If the standard ever did run out of element IDs or command types, it could overcome this problem by securing an OUI and using this to open up new element IDs or command types.

The content and use of the vendor-specific IEs and vendor-specific commands is outside the scope of the standard.

EXAMPLE OF THE LIFE CYCLE OF A DEV

The standard provides information on each of the various events that a DEV might encounter but does not put this together to provide a beginning to end description of the life cycle of a DEV. Because the application requirements for each DEV are different, the life cycle of each DEV will be different as well. This section provides a set of MSCs that give an example of a typical DEV's interaction with a piconet. In the MSCs, the ACKs have been left out for clarity. The first MSC, Figure 4–41, shows a DEV that joins a piconet, allocates channel time and exchanges data with another DEV in the piconet.

The second MSC, Figure 4–42, illustrates the same DEV leaving the piconet and the actions that the PNC takes when a DEV leaves a secure piconet.

INTEL-1010
U.S. Patent 9,265,043

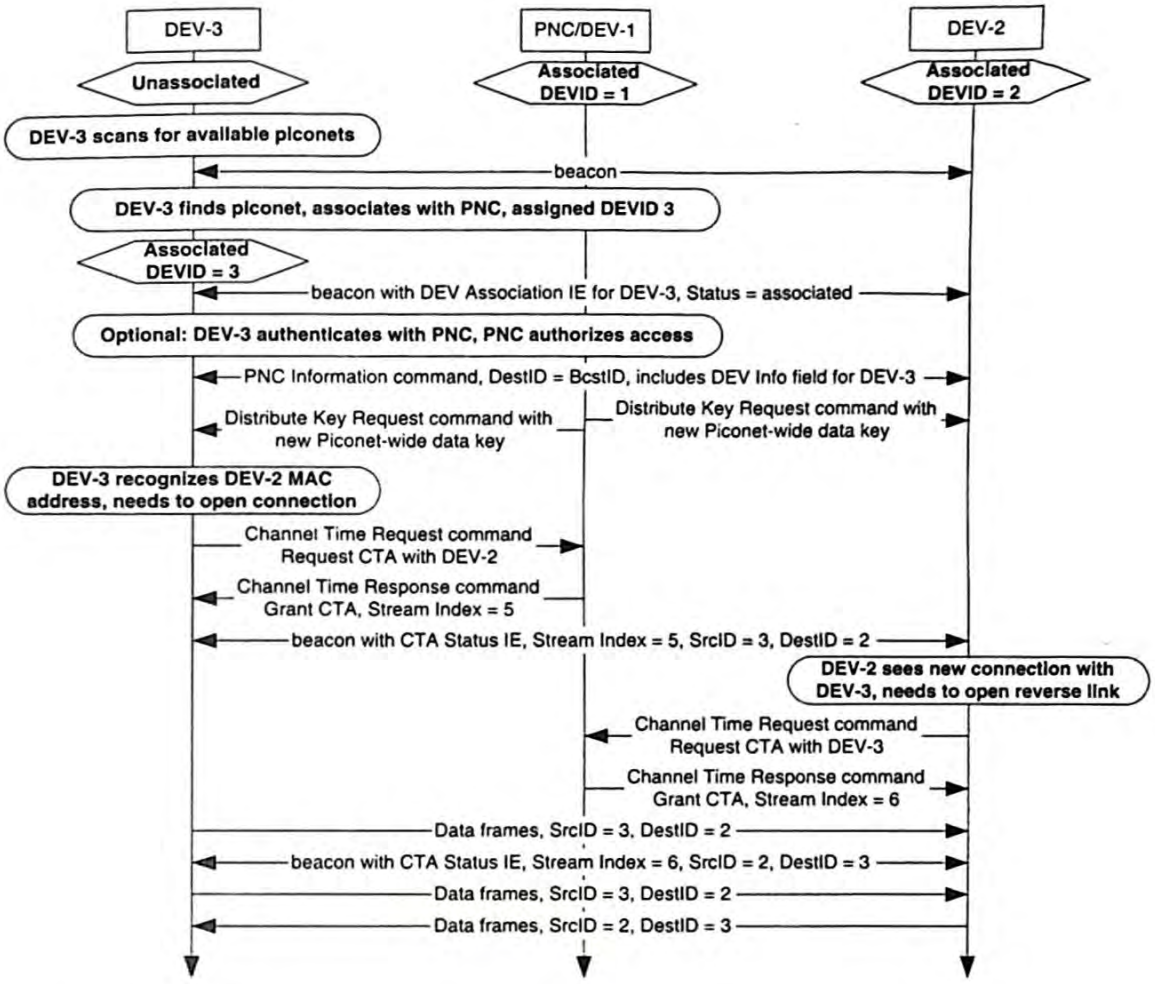


Figure 4–41: Example of a DEV joining a piconet and allocating channel time

INTEL-1010
U.S. Patent 9,265,043

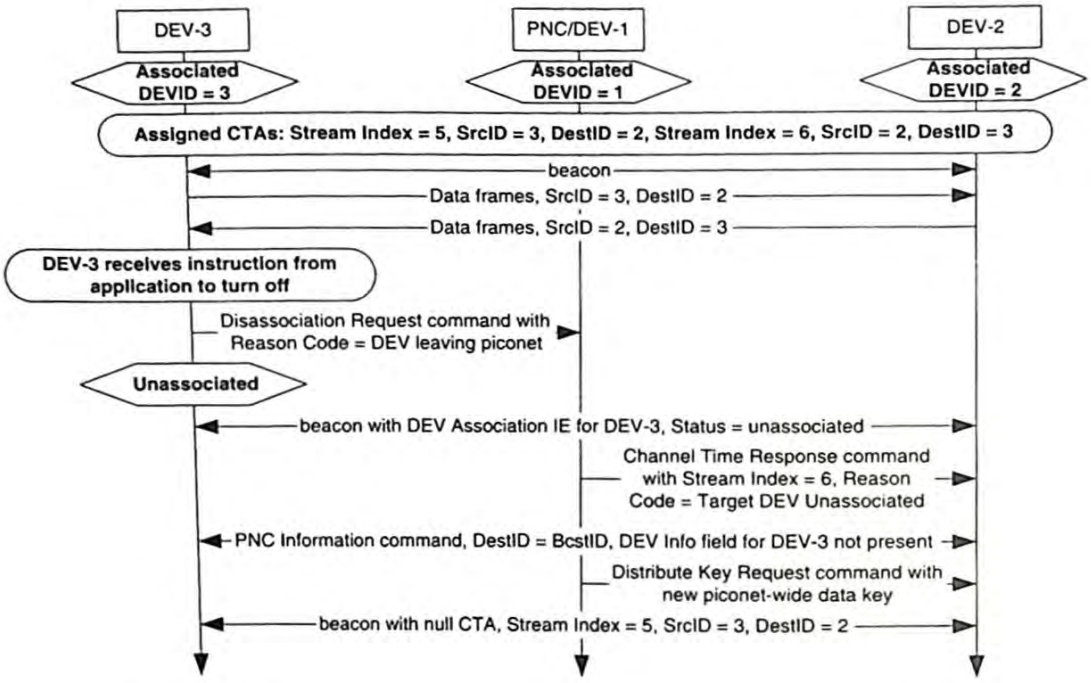


Figure 4–42: Example of a DEV leaving a secure piconet

INTEL-1010
U.S. Patent 9,265,043

Chapter 5 Dependent piconets

INTRODUCTION

Dependent piconets were added to the standard to enable a variety of applications for IEEE Std 802.15.3. There are two types of dependent piconets defined in the standard: child and neighbor. The key difference between a child and a neighbor piconet is the membership status of the dependent PNC. A child PNC is a full member of the parent piconet. It can exchange data with other DEVs in the piconet and can send commands to any of the other DEVs in a piconet. A neighbor PNC, on the other hand, is not a full member of the parent piconet and so it is restricted in its communications in the parent piconet. The neighbor PNC is associated in the parent piconet, but it is not required to become a secure member of the parent piconet. Thus, the neighbor PNC is not a full member of the parent's piconet.

A DEV may decide to form a dependent piconet for a variety of reasons. If all of the channels are occupied with 802.15.3 piconets, other wireless networks, or with strong interferers, the DEV may want to share channel time with the existing piconet. The dependent piconet capability allows the DEVs to share the time on the channel while maintaining the synchronization required for the TDMA protocols in each piconet. Splitting the channel time between the two piconets allows them to achieve a higher level of coexistence. This coexistence capability is also available to other, non-802.15.3 wireless networks through the use of neighbor piconets.

Another reason for forming a dependent piconet is to extend the range of communications for a DEV. The parent PNC will only have a limited range, and this will limit the number of remote DEVs with which a given DEV can communicate. A DEV in the piconet may recognize that there are other DEVs with which it needs to communicate that are within range of the DEV but are not in range of the PNC. This DEV could then form a dependent piconet to communicate with those DEVs while still maintaining its connections in the parent piconet.

INTEL-1010
U.S. Patent 9,265,043

Security considerations can also be a reason for choosing to form a dependent piconet. In a piconet, all broadcast traffic is protected with the group key, and all of the DEVs in the piconet are able to hear this traffic. If a DEV wants to be able to restrict the distribution of this broadcast traffic, it could form a dependent piconet and only admit the DEVs to which it intends to send the broadcast data. Because the DEV would be the PNC of the dependent piconet, it can set the security policies to a level that is consistent with its applications. The DEV may also wish to form a dependent piconet if the parent PNC is not supporting security or if it using a security level that is not strong enough for the DEV's applications.

Once a DEV has decided that it wants to form a dependent piconet, it then needs to determine if it will form a child or a neighbor piconet. In general, the DEV will select to form a child piconet, but there are many reasons why a DEV would choose to create a neighbor piconet instead of a child piconet. One reason could be that the neighbor PNC is unable to enter into a secure membership with the parent PNC. An example of this is an apartment building where there are multiple piconets, each in a different apartment. In order to share the time on the air, the piconet of one apartment might want to get channel time from a nearby piconet, but it would be unable to authenticate to the parent PNC because the piconet did not belong to the same user as that of the parent PNC. Another reason for creating a neighbor piconet instead of a child piconet is that the neighbor PNC may be from another type of network, e.g., 802.11b, that only supports the minimal functionality required for a neighbor PNC. This DEV would be incapable of operating as a fully functional 802.15.3 DEV but might be able to support the commands required to join an 802.15.3 piconet as a neighbor and request channel time.

The most obvious difference between a child and a neighbor piconet is the DEVID that is used in the parent piconet. The PNC of a child piconet has a regular DEVID assigned to it in the parent piconet while the PNC of a neighbor piconet is assigned one of the reserved NbrIDs (see "Assigning DEVIDs" on page 80). The PNC of a dependent piconet is assigned a minimum of three different DEVIDs. The first DEVID is the one that is assigned to it in the parent piconet, either a regular DEVID or the NbrID. The second ID is the PNCID that it uses for command-related traffic in the dependent piconet of which it is the PNC. The dependent PNC also assigns itself a DEVID to use for data traffic in the dependent

INTEL-1010
U.S. Patent 9,265,043

piconet, just as the PNC of the parent piconet has a DEVID assigned for data traffic.

Another key difference between a neighbor piconet and a child piconet is the frames that are allowed to be sent by the DEV in the parent piconet. Because the child PNC is a full member of the parent piconet, it can send data frames and any valid commands to any of the DEVs in the piconet. The neighbor PNC, on the other hand, is allowed to send only the following commands in the parent piconet:

- Association Request command
- Disassociation Request command
- Channel Time Request command
- Vendor Specific command
- Security Message command
- Any Probe Request, Probe Response or Announce commands
- Any required Imm-ACK frames

Because the neighbor PNC is a full member of its dependent piconet, it is allowed to send and receive data and commands during the operations of its piconet.

IEEE Std 802.15.3 does not require that all PNC-capable DEVs implement dependent piconet functionality. This capability is more complex than requirements for a regular PNC, and so the task group did not want to burden every PNC-capable DEV with this added cost. The PNC of a piconet is not required to allow neighbor piconets to be formed within its own piconet. The PNC is required to respond to a private channel time request from a DEV that could be used to start a child piconet, but it is not required to grant the channel time. The algorithm for granting channel time is outside of the scope of the standard, and the PNC may decline the request because it either cannot support pseudo-static allocations or the current traffic in the piconet is too dynamic to support the allocation of a pseudo-static CTA. This would have the effect of not allowing any child piconets without explicitly refusing them because the request is for a child piconet.

The initial proposals for creating dependent piconets used special commands for neighbor association and channel time requests. However, after working with dependent piconets in early stages of the draft, it was clear that some

INTEL-1010
U.S. Patent 9,265,043

simplifications could be made. First, the dependent piconet really only required channel time from the parent PNC. Other than that, the PNC did not need to be involved at all in the operations of the dependent piconet. The second attribute is based on the idea that in an 802.15.3 piconet once a DEV has been allocated channel time, it can use that time for essentially anything that it wants to do. The only restriction is that the DEV needs to ensure that its transmissions or the transmissions that it prompts, e.g., ACKs or allocations in a dependent piconet, do not extend beyond the temporal boundaries of the CTA. Thus, the dependent PNC can use the regular channel time request procedures to ask for time to be set aside for the operations of the dependent piconet. Finally, the idea of a pseudo-static CTA was used to assist the dependent PNC. Because this type of allocation is changed infrequently, it works well as the allocation for a dependent network.

STARTING A DEPENDENT PICONET

The first task in starting a dependent piconet is to associate with the PNC of the parent piconet. The DEV joins the piconet using the Association Request command (see "Association" on page 74) and follows the normal association process. If the DEV is going to become a child PNC, the association process is indistinguishable from the association of any other DEV. If the DEV is intending to become a neighbor PNC, there a few small differences in the association process.

First, the DEV sets the Neighbor PNC field to one in the PNC Capabilities field that it sends to the PNC in the Association Request command. When the PNC sees that this bit is set, it knows that the DEV is not requesting to become a full member of the piconet, but rather that it is asking to become a neighbor PNC. Because the PNC is not required to support neighbor piconets, the PNC can refuse the association request by sending an Association Response command with the Dest-ID set to the UnassocID, the requesting DEV's DEV address, and with the Reason Code set to "Neighbor piconet not allowed." If the DEV receives this type of a rejection to its association request, it should not try to associate with this piconet as a neighbor PNC as long as the parent PNC remains unchanged. The neighbor PNC DEV will know that the PNC has changed by examining the PNC Address field in the Piconet Synchronization Parameters field in the beacon.

INTEL-1010
U.S. Patent 9,265,043

The PNC may refuse the association for other reasons; for example, it may be in the middle of the PNC hand-over process or it may be changing channels. Depending on the Reason Code in the Association Response command, the DEV may wish to retry the request at a later time, for example, after the handover or channel change has completed. In general, the DEV can use the beacon to determine if the conditions have changed in such a way as to allow the DEV to associate as a neighbor PNC. For example, if the Reason Code is "Already serving maximum number of DEVs," then when the DEV sees a beacon with a DEV Association IE that indicates that a DEV has just disassociated, it may want to retry the association process with the PNC. If the PNC grants the association request for a neighbor PNC, it will assign the DEV one of the available NbrIDs to use for communications with the parent PNC.



The NbrID is not used for any data traffic in the parent piconet or for any data or command traffic in the neighbor piconet. During the time allotted for the neighbor piconet, the neighbor PNC uses either the PNCID or its DEVID in the neighbor piconet for communications. Thus the neighbor PNC has 3 DEVIDs: the NbrID in the parent piconet, the PNCID in the neighbor piconet and a regular DEVID in the neighbor piconet.

When the association process is complete, either for a child or neighbor PNC, the next step is to request channel time for the dependent piconet. The DEV will use a regular Channel Time Request command with the following requirements:

- The Target ID will be the DEV's DEVID (either a regular DEVID or the NbrID).
- The Stream Index will be set to the Unassigned Stream Index (see "Stream connections" on page 92) This implies that the DEV is only able to request only an isochronous allocation as an asynchronous allocation would not make sense for this application.
- The PM CTRq Type field shall be set to ACTIVE. (A power-save mode allocation is not possible because the PNC is always required to be ACTIVE.)
- The CTA Rate Type field is set to indicate a super-rate allocation.
- The CTA Rate Factor is set to 1 to indicate one allocation per superframe.
- The CTA Type field is set to request a pseudo-static allocation.

All other fields in the Channel Time Request command can be set to values that are appropriate for that request. In particular, the dependent PNC will need to

INTEL-1010
U.S. Patent 9,265,043

specify the minimum time that is required for it to support the operations of the dependent piconet.

If the SrcID for the CTRq command is one of the NbrIDs, the PNC will recognize it as a request for time for the neighbor piconet. If the request by a neighbor PNC is malformed in some manner, e.g., the Target ID is not its NbrID or the request is for a sub-rate allocation, then the PNC will reject the request. However, if the request comes from a regular DEV, the PNC will attempt to grant the request, even if it does not adhere to the above rules. The PNC will attempt this allocation because the child PNC is a member of the piconet, and so it is allowed to request channel time to other DEVs in the piconet or for purposes other than a dependent piconet. The PNC will consider each request based on the channel time available in the superframes, including the differential loading due to sub-rate and power-save mode allocations.

The PNC will send a Channel Time Response command to the DEV to indicate the result of the request for channel time. If the request is unsuccessful, the DEV can retry the request with different values for the timing parameters if it thinks this will improve the probability that the request will be granted. If the request is successful, then the PNC will begin to place the CTA blocks in the beacon to allocate the time for the DEV to operate as the dependent piconet. At this point, the DEV is now the PNC of the dependent piconet and it is allowed to begin sending a beacon, associating new members and allocating channel time. The dependent PNC must select a PNID that is distinct from the parent piconet's PNID. If the dependent PNC did not do this, then DEVs in the parent piconet that were in receive mode during the dependent piconet's CTA might ACK frames or respond to commands that were intended for a DEV in the dependent piconet that had the same DEVID as the DEV in the parent piconet. The process of a DEV becoming a child PNC is illustrated in Figure 5–1.

If the dependent piconet is a child piconet or an 802.15.3 neighbor piconet, then the dependent PNC will need to set aside time in the beacon that corresponds to the time in the parent piconet that is not allocated to the dependent piconet. The child or 802.15.3 neighbor PNC does this by putting a CTA block in the beacon with the SrcID and DestID set to the PNCID and the start time and duration corresponding to the time that is reserved for the parent piconet. Because 802.15.3 has strict rules regarding when a DEV is allowed to transmit, the child PNC is

INTEL-1010
U.S. Patent 9,265,043

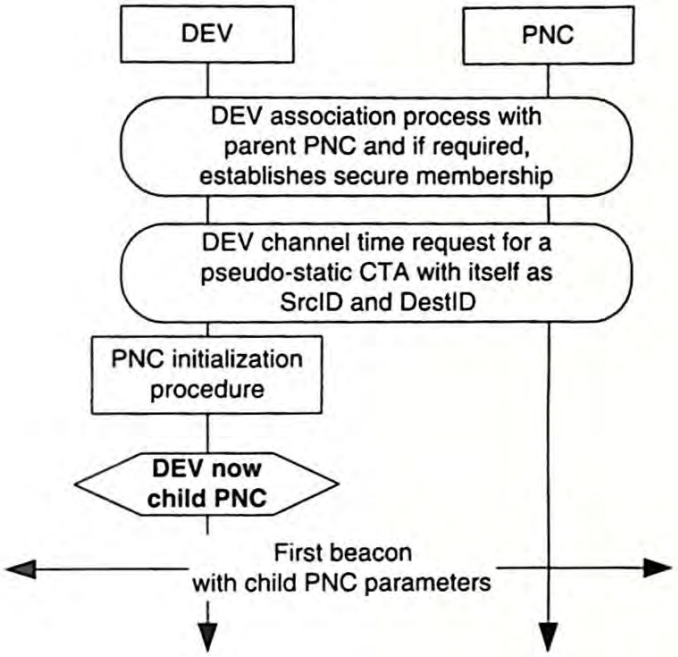


Figure 5–1: Process for starting a child piconet

able to ensure that all of the DEVs in its piconet do not violate the time allocated for the dependent piconet operation. The alignment of the child and parent superframes is illustrated in Figure 5–2.



The parent PNC may communicate with the child PNC during time allocated for the parent piconet. The child PNC uses the DEVID that was assigned as its DEV personality in the parent piconet.

Figure 5–2 shows that in a child piconet, the start time of the reserved channel time is equal to the end of the allocation for the dependent piconet. Likewise, the end of the reserved channel time corresponds to the start time of the CTA for the dependent piconet. Because the child is a member of the parent piconet, there are times during the parent superframe when the child PNC can communicate with the parent PNC or with other DEVs in the parent piconet. This is indicated in Figure 5–2 as blocks of time marked with "C-P." When either of the PNCs, child or parent, is sending a beacon, communication is not possible (other than the information passed in the beacon). The parent PNC is also not allowed to communicate with the dependent PNC during the blocks of time

INTEL-1010
U.S. Patent 9,265,043

marked "C-C." Although the child PNC is a member of the parent piconet, the parent PNC is not necessarily a member of the child piconet, and so it cannot communicate with the other DEVs in the child's piconet. The parent PNC may join the child piconet if it wants to communicate with DEVs that are associated in that piconet but are not associated in the parent piconet. If the parent PNC joins the child piconet, it will be as a regular DEV and not as the PNC.

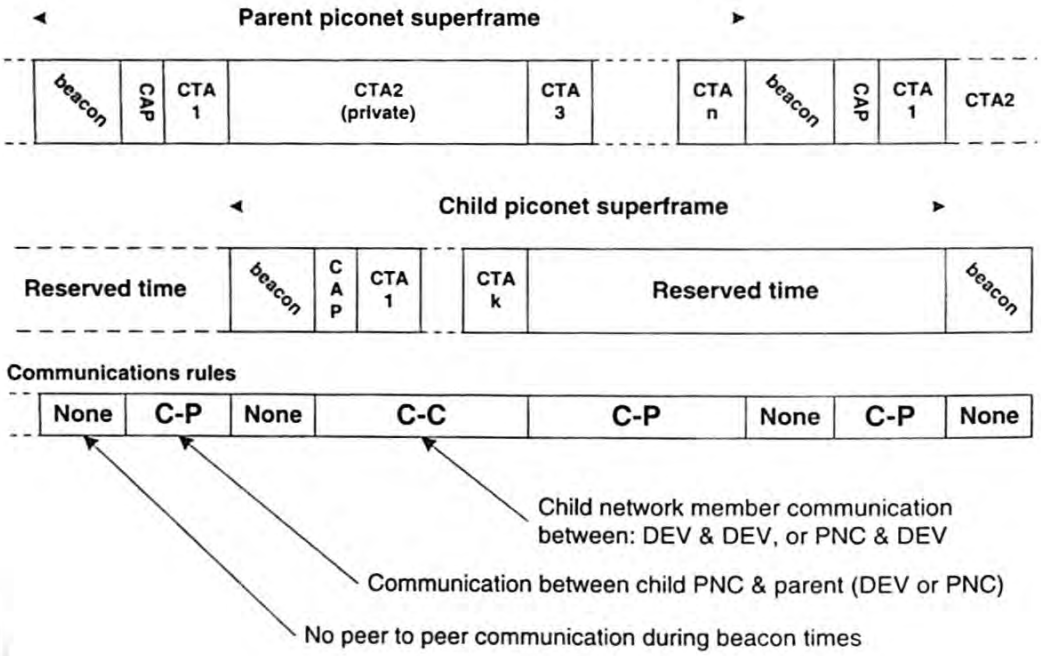


Figure 5–2: Relationship of the parent superframe to the child superframe

As Figure 5–2 shows, the child PNC can act as a bridge between the two piconets, the parent and the child piconet. The child PNC is allowed to implement either a bridging protocol, such as 802.1d, or a routing protocol to enable data to be passed between the two piconets. However, these protocols are functions that exist at the higher protocol layers and so are out of the scope of IEEE Std 802.15.3 MAC and PHY. An example of this bridging is illustrated in Figure 5–3.

INTEL-1010
U.S. Patent 9,265,043

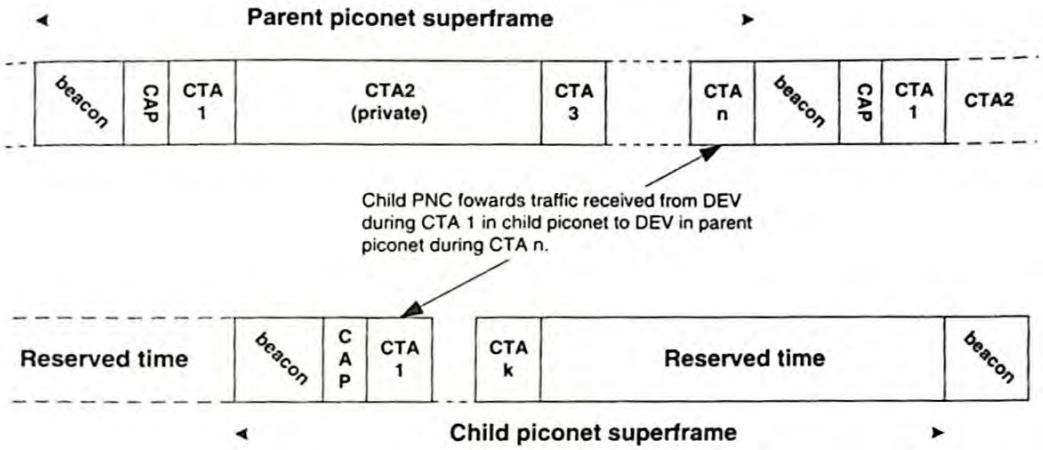


Figure 5–3: Child PNC bridging data between networks

The process for a DEV becoming a neighbor PNC is illustrated in Figure 5–4. As the figure indicates, the neighbor DEV may not be operating an 802.15.3-compliant network. However, if the DEV wants to join as a neighbor PNC, it is required to support the minimal functionality described in IEEE Std 802.15.3. In addition, the neighbor PNC must make sure that all of the members of its wireless network do not transmit outside of the time allocated for the dependent piconet. If the neighbor PNC is unable to guarantee this, it is not allowed to join the 802.15.3 piconet as a neighbor PNC.

The relationship between the superframes of the parent piconet and the neighbor piconet are illustrated in Figure 5–5. The parent and neighbor PNCs can carry on limited communications during the time periods marked "N-P," although this communication is restricted to the commands necessary to maintain the neighbor piconet. The neighbor PNC is not allowed to communicate with DEVs in the parent piconet because it is not a member of that piconet. Likewise, the parent PNC is not allowed to engage in any frame exchanges during the time set aside for the neighbor piconet because it is not a member of that piconet. As with the case of the child piconet, the parent PNC is allowed to join the neighbor piconet so that it can communicate with the DEVs in that network (assuming it supports the protocol that is being used in the neighbor piconet).

INTEL-1010
U.S. Patent 9,265,043

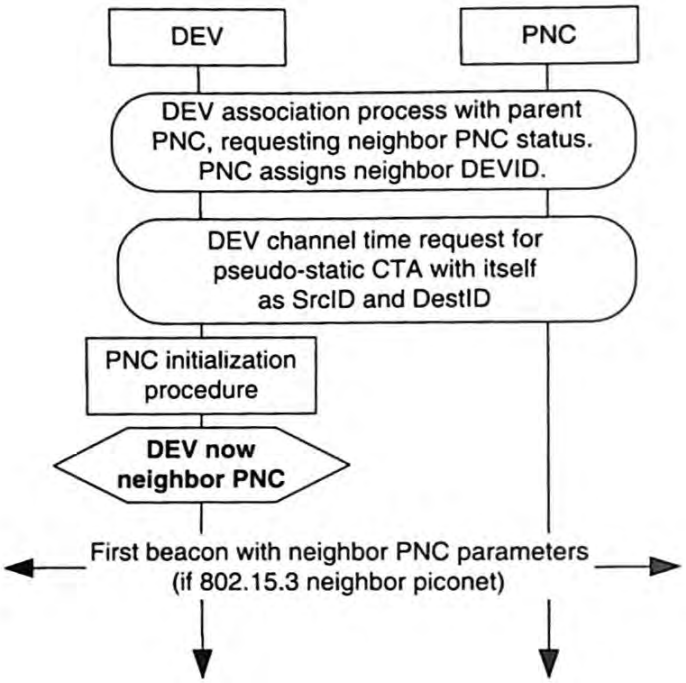


Figure 5–4: Process for starting a neighbor piconet

ARENT PNC CEASING OPERATIONS WITH DEPENDENT PICONETS

When the parent PNC stops the parent piconet using the shut down procedure, the dependent piconets will need to adjust for the loss of their synchronization. If there is only one dependent piconet that is part of the parent piconet, that piconet will be able to continue operations. However, if there is more than one dependent piconet that is a part of the parent's piconet, then they both cannot continue to operate. Once the parent PNC stopped sending beacons, the dependent PNCs might assume that they can use the entire time in the channel, expanding to fill the time that was allocated for traffic in the parent piconet. However, if all of the dependent piconets do this, they would collide, causing lost frames and poor QoS.

One way to resolve this problem would be to require that all dependent piconets cease operations if the parent piconet also ceases operations. However, it is possible for one dependent piconet to remain as long as the others know to cease operations. The task group solved this problem by adding a field to the Piconet Shutdown IE that contains the DEVID of the dependent piconet that is allowed to

INTEL-1010
U.S. Patent 9,265,043

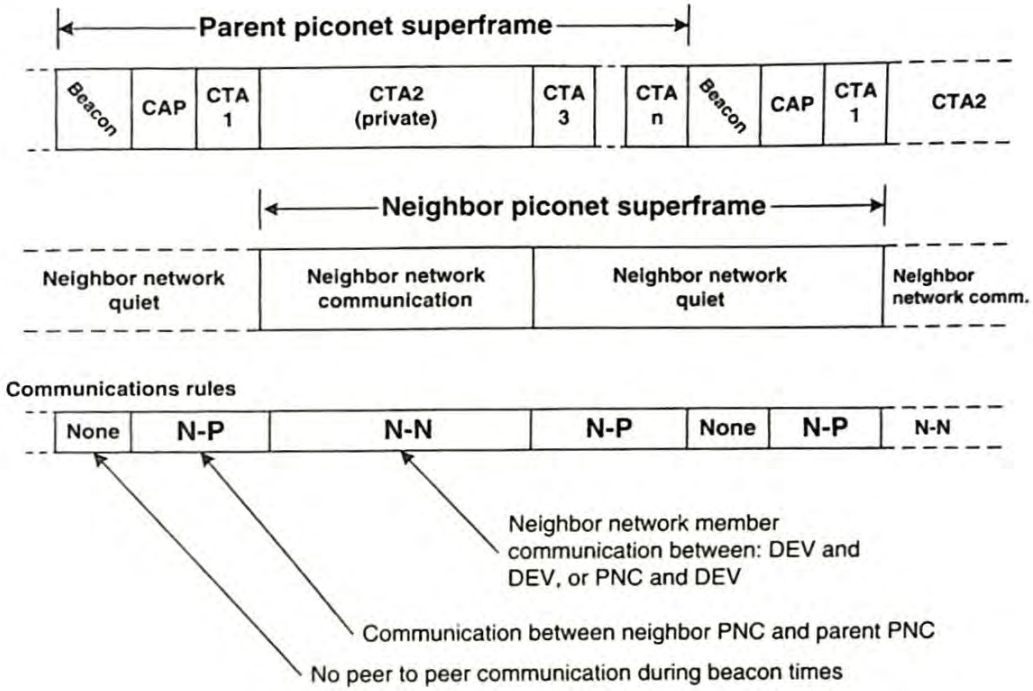


Figure 5–5: Relationship of the parent superframe to the neighbor superframe

continue operations. Rather than forcing the PNC to determine which dependent piconet is most deserving to continue operations, the standard requires the PNC to select the dependent piconet with the lowest DEVID to continue operations. It is unlikely that the PNC could determine the "best" piconet from its limited knowledge and in the limited amount of time allowed for the piconet shutdown procedure. Thus, an arbitrary choice is as good as an attempt to select the "best" one, and it requires very little overhead in an implementation. Using the lowest DEVID also gives preference to child piconets over neighbor piconets because the values of the NbrIDs are all greater than the highest allowed DEVID value.

All of the other dependent PNCs, if they do not see their DEVID in the beacon, will be required to either shut down their piconets, change PHY channels, or join another piconet as a dependent piconet before the parent PNC sends its last beacon. Once the parent PNC has sent its last beacon, all DEVs that are not members of the dependent piconet that was selected to continue operations will have to cease all transmissions. The dependent PNC that was selected to continue

INTEL-1010
U.S. Patent 9,265,043

operations will remove the Parent Piconet IE from its beacon to signify that it is no longer a dependent piconet and it can likewise remove the allocation in its superframe for the parent piconet's operations. DEVs that were a part of any of the piconets that shut down, either from the parent piconet or from another dependent piconet, can then join the remaining piconet either as DEVs or to start a dependent piconet.

PARENT PNC STOPPING A DEPENDENT PICONET

The parent PNC is allowed to force a dependent piconet to cease operations. The method that the parent PNC uses to do this depends on the type of dependent piconet that it wants to shut down. If the parent PNC is going to shut down a child piconet, it uses the stream termination procedure (see "Stream connections" on page 92) to remove the channel time that is allocated for the child piconet. The parent PNC sends the Channel Time Response command to the child PNC with a Reason Code that explains the reason that the parent PNC is removing the CTA. The child PNC remains a member of the parent piconet, and its other stream and asynchronous connections remain unchanged by the procedure.

If the dependent piconet is a neighbor piconet, the parent PNC uses the disassociation process (see "Disassociation" on page 78) to end the neighbor PNC's association with the parent piconet. The Disassociation Request command is used instead of the stream termination procedure because the neighbor PNC's only reason for associating with the parent PNC is to operate a neighbor piconet. If the parent PNC is not going to allow the neighbor piconet to operate, there is no reason for the neighbor PNC to remain associated in the parent piconet.

Once the dependent PNC has been informed, either via the Channel Time Response command or the Disassociation Request command, that it will no longer be able to operate a dependent piconet as a part of the parent piconet, the dependent PNC can then either shut down its piconet, change channels, or join another piconet as a dependent piconet. The parent PNC will listen for the dependent PNC's shut down or channel change procedure to determine when to remove the channel time for the dependent piconet from the beacon. However, the parent PNC is not required to wait for an indefinite period of time and it may choose a timeout for the dependent piconet that is shorter than the time indicated in the dependent PNC's shut down or channel change process. If the dependent

INTEL-1010
U.S. Patent 9,265,043

piconet is an neighbor piconet that is not 802.15.3-compliant, the parent PNC will select a time that is equivalent to what it would use to shut down its own piconet to allow the dependent piconet to either shut down or change to another channel.

Regardless of the length of time that the PNC allows for the dependent PNC to cease operations in the current PHY channel, when the CTA for the dependent piconet is removed from the beacon, all of the DEVs in the dependent piconet, including the dependent PNC, shall cease transmitting in that CTA. Because the CTA for a dependent piconet is a pseudo-static allocation, the parent PNC might want to allow mMaxLostBeacons number of superframes before it reallocates that time to another DEV. It can allocate it sooner, but the first few superframes may experience high frame loss if the DEVs in the dependent piconet have not yet realized that the allocation has been finally removed.

HANDING OVER PNC RESPONSIBILITIES IN A DEPENDENT PICONET

One of the key capabilities that IEEE Std 802.15.3 provides is the ability of the PNC to hand over control of the piconet to another DEV in the piconet. This capability is critical to the 802.15.3 protocol because the "best" PNC is not initially selected to be the PNC of the new piconet. Although early drafts of the standard included a PNC election process, this procedure was dropped in favor of simply letting the first PNC-capable DEV on the air become the PNC. This DEV will then hand over control to the "better" DEVs as they join the piconet. Eventually, the same result as a PNC election process will occur, in that the "best" DEV becomes the PNC of the piconet. PNC hand-over also allows the piconet to continue uninterrupted when the PNC needs to leave the piconet for any reason.

Thus, the PNC hand-over process is a key part of the operation of IEEE Std 802.15.3. However, up until the first sponsor ballot, the PNC hand-over process for dependent PNCs was not even supported in the standard. This capability was added after the first sponsor ballot and now is a part of the standard for DEVs that implement the dependent PNC capability.

Although the parent PNC can hand over PNC responsibilities to a DEV that is currently the PNC of a child piconet, this does not imply that the dependent PNC will then merge the two piconets. The standard does not provide either the commands or the procedures that are necessary to support merging two piconets.

INTEL-1010
U.S. Patent 9,265,043

This may be a future enhancement to the standard, but it is not currently supported. The PNC is never allowed to hand over control to a neighbor PNC because this DEV is not a full member of the parent piconet. If the parent PNC does hand over control of the piconet to a DEV that is currently a child PNC, that DEV would have to operate two distinct piconets simultaneously. If the child PNC is unable to support this mode of operation, it will refuse the hand-over request by sending the PNC Handover Response command to the PNC with the Reason Code "Handover refused, unable to act as PNC for more than one piconet."

Handing over control of dependent piconet required the addition of two new capabilities to the standard. The first capability is to allow a DEV, under certain circumstances, to refuse a hand-over request. In almost every other case[28], the DEV that is selected by the PNC to become the new PNC is required to accept the nomination and will become the new PNC of the piconet. However, in the case of a dependent piconet, the new dependent PNC will need to become a member of the parent piconet before it can take over as the PNC of the dependent piconet. This may not be possible for any one of a number of reasons. For example, the DEV may be unable to enter into a secure relationship with the parent PNC, and so it cannot become a member of the parent piconet. This issue does not apply to neighbor piconets where the neighbor PNC is only associated and is not required to have a secure relationship with the parent PNC. The selected DEV may not be able to associate with the parent PNC because the parent PNC is unable to handle any new DEVs in its piconet. The selected DEV may not even be in range of the parent PNC; in which case, it will be impossible for it to operate as the dependent PNC. If the selected DEV is unable to join the parent piconet, the DEV will refuse the hand-over process by sending the PNC Handover Response command to the PNC with the Reason Code set to "Handover refused, unable to join parent piconet."

The other change that was necessary in the standard was to allow a DEV to hand over its CTA to another DEV. Assuming that the selected DEV is able to join the parent piconet, it will be assigned a DEVID that is different from the current dependent PNC. The CTA for the dependent piconet, however, will be associated with the old DEVID and the new PNC will be unable to modify or respond to any

[28] The other exception is that a DEV that is acting as a child PNC may refuse the handover due to its inability to run more than one piconet at the same time.

INTEL-1010
U.S. Patent 9,265,043

changes in the allocation. Even worse, if the current dependent PNC is handing over control because it is going to shut off, at some point, it will disassociate from the parent piconet and the parent PNC will then remove the CTA from the beacon and the dependent piconet will cease to exist. Thus, the parent PNC needs to be informed that a new DEV will be taking over as the "owner" of the time allocated to the dependent piconet.

The hand over of channel time is restricted to private pseudo-static allocations only, and only the current source of the allocation can request a change in the "ownership" of the CTA. To hand over the CTA to another DEV, the source DEV sends a Channel Time Request command to the PNC that contains the following information:

- The Num Targets field is set to one.
- The Target ID List field contains the DEVID of the target DEV that is to receive control of the CTA.
- The Stream Request ID field is set to zero.
- The Stream Index field set to the stream index of a CTA that has already been allocated to the dependent PNC as a private, pseudo-static CTA.
- All other fields set to the same values as in the last successful CTRq for this Stream Index.

The PNC will notice a change in the Target ID List and will know that this is a request to hand over control of the CTA to the DEV indicated in the Target ID list. The PNC will grant this request as long as the DEV that is to receive control of the CTA is a member of the piconet and its power management mode is ACTIVE. The PNC will respond with the Channel Time Response command indicating the success or failure of the request. If the request failed, the PNC will set the Reason Code field to indicate the reason why the request was refused. The frame exchange sequence for handing over control of a CTA is illustrated in Figure 5–6.



The standard does not explicitly state that the DEV that receives control of the CTA is in ACTIVE mode, but it is not possible for a DEV to be in a PS mode and have a super-rate allocation. Thus the DEV needs to be in ACTIVE mode when the control is passed.

INTEL-1010
U.S. Patent 9,265,043

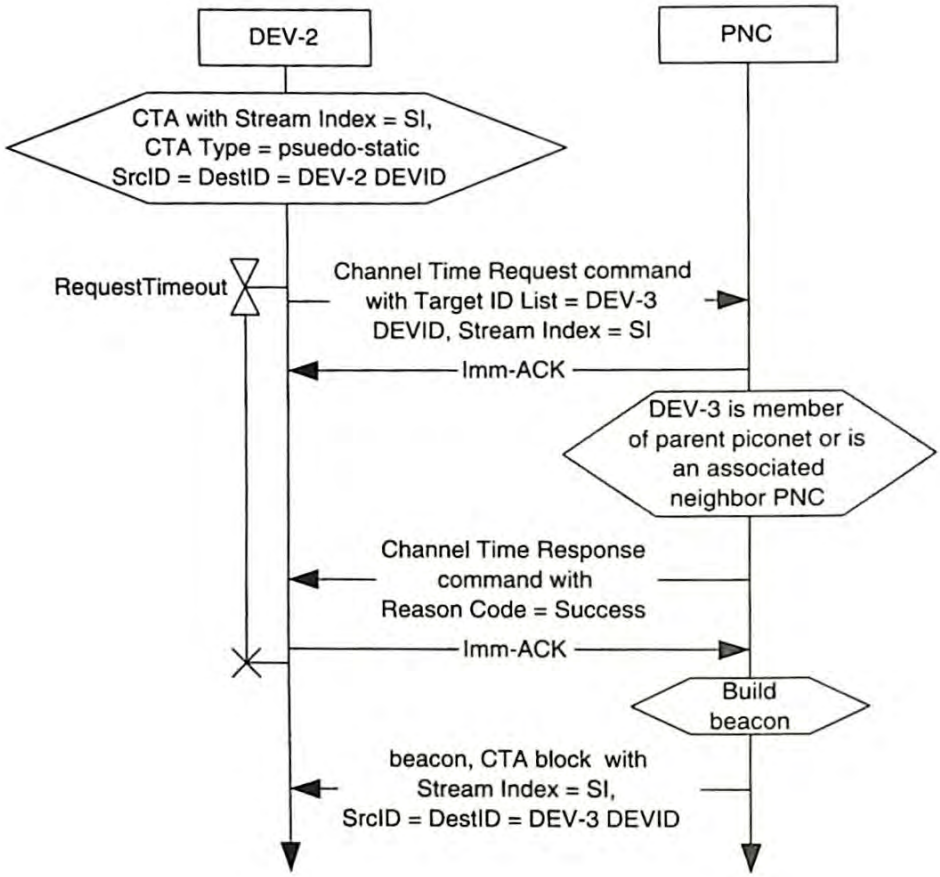


Figure 5–6: Dependent PNC handing over control of a CTA

With these two capabilities, the standard defines a procedure that will allow a dependent PNC to hand over control to another member of its piconet. Perhaps more important, it is possible to describe the potential failures in the dependent PNC handover process and how the protocol can handle them.

The dependent PNC hand-over process begins in the same manner as a regular PNC hand-over, with the current PNC sending the PNC Handover Request command to the DEV that it has selected to become the new PNC, as shown in Figure 5–7. In this and the two subsequent figures, the identities PNC, DEV-2, and DEV-3 are all relative to the dependent piconet and not the parent piconet. If DEV-2 is not a member of the parent piconet, then the DEV will begin the association process to join the parent piconet and, if required, establish a secure

INTEL-1010
U.S. Patent 9,265,043

relationship with the PNC. DEV-2 can join the parent piconet as either a neighbor or as a member of the piconet. While DEV-2 is attempting to join the parent piconet, the current dependent PNC will be sending DEV-2 the information about all of the DEVs with the PNC Information command, all of the current channel time requests with the PNC Handover Information command, and the power save information, if any, using the PS Set Information Response command. DEV-2 may also request the transfer of any security information at this point using the Security Information Request command. The dependent PNC transfers all of this data, anticipating that DEV-2 will be successful in joining the parent piconet. The transfer of this information does not interfere with either DEV-2's association and secure membership process or with the transfer of the dependent piconet data because the former occurs only during time reserved for the parent piconet, whereas the latter only occurs during time reserved for the dependent piconet.

Once the transfer of the information is complete and DEV-2 has joined the parent piconet, it will send a PNC Handover Response command to the dependent PNC with a Reason Code set to the DEVID that was assigned to it by the parent PNC. This lets the current dependent PNC know that the new dependent PNC is ready to take over control of the piconet. The Reason Code in the PNC Handover Response command is interpreted as follows:

- *0x00:* Success. (This code is used only for a non-dependent PNC handover.)
- *0x01–0xEC:* Success, the DEV that is to become the new dependent PNC is a member of the parent piconet with a DEVID equal to the value of the Reason Code. (These values cover all possible DEVIDs for members of a piconet.)
- *0xED–0xF6:* Reserved. (These values are reserved DEVIDs and so they are reserved for this command as well.)
- *0xF7–0xFC:* Success, the DEV that is to become the new dependent PNC has associated in the parent piconet as a neighbor PNC with a DEVID equal to the value of the Reason Code. (These values are all NbrIDs.)
- *0xFE:* Handover refused, unable to join the parent piconet. (This value is the same as the unassociated ID, which is appropriate for this error code.)
- *0xFF:* Handover refused, unable to act as a PNC for more than one piconet. (This value is the same as the BcstID and so it is never assigned to a DEV as its DEVID).

INTEL-1010
U.S. Patent 9,265,043

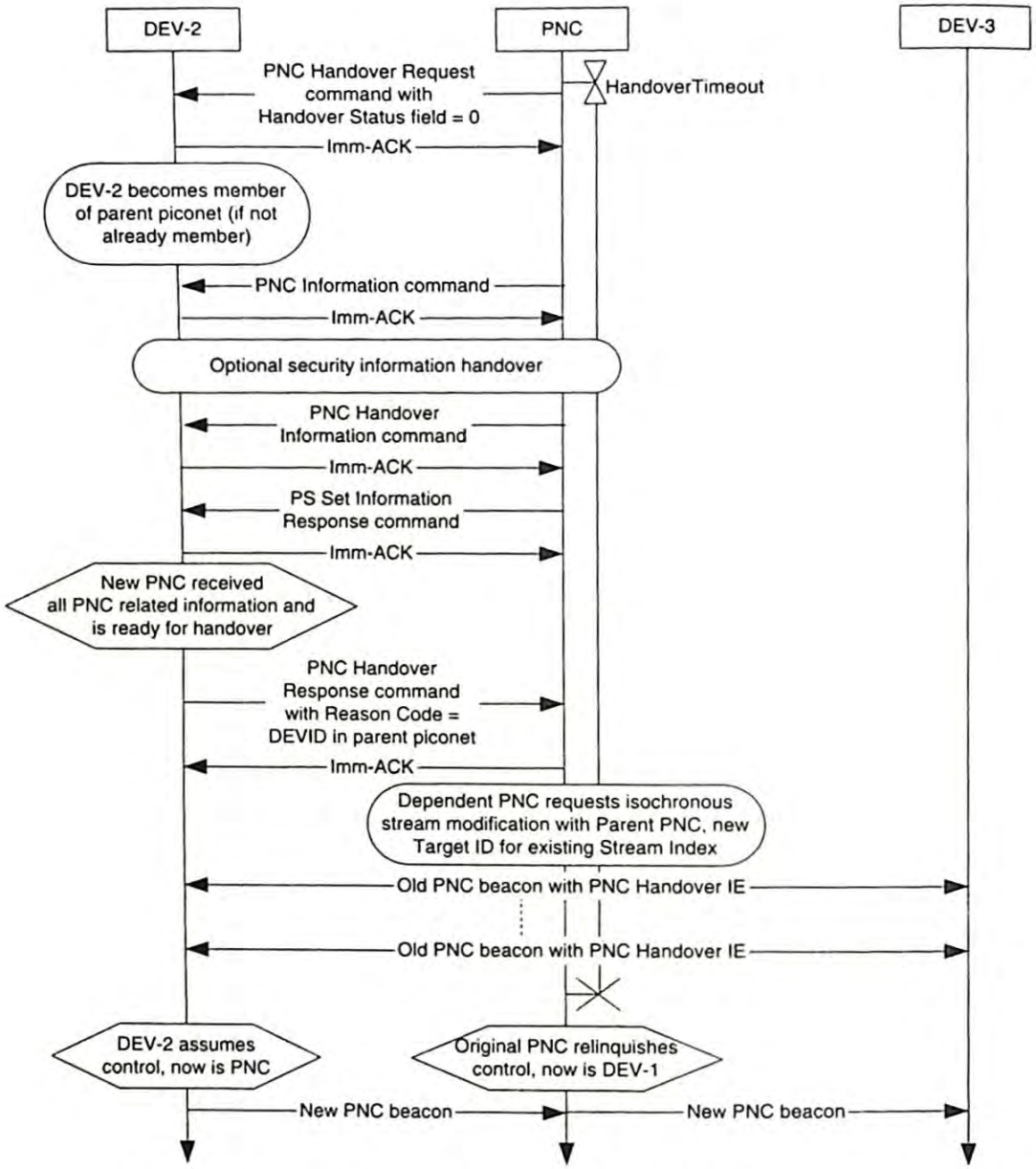


Figure 5–7: Successful hand over of dependent PNC

INTEL-1010
U.S. Patent 9,265,043

At this point, the dependent PNC uses the Channel Time Request command to hand over the control of the dependent piconet CTA to the new dependent PNC. Once the parent PNC changes the SrcID and DestID of the dependent piconet CTA, the current dependent PNC must complete the hand-over process because it will not be able to regain control of the CTA.

After the control of the dependent piconet CTA has been transferred to the new dependent PNC, the current dependent PNC will begin placing the PNC Handover IE in its beacon with the Handover Beacon Number field set to the beacon number of the first beacon that will be sent by the new PNC. The last superframe controlled by the current dependent PNC will be the one in which the beacon number is one less than the value of the Handover Beacon Number field. The following superframe will begin when the new dependent PNC sends its first beacon.

There are multiple points in the hand-over process where it can fail. The current dependent PNC can cancel the hand-over process up until the time that it requests that the parent PNC hand over control of the dependent piconet CTA to the new dependent PNC. The dependent PNC cancels the process by sending a PNC Handover Request command to DEV-2 with the Handover Status field set to 1 to indicate that the process has been cancelled.

The hand-over process can also fail if DEV-2 fails to join the parent piconet. If DEV-2 attempts to join as a neighbor PNC and the parent PNC does not support neighbor PNCs or does not wish to allow any more neighbor PNCs, then the association request by the new dependent PNC will be rejected. In that case, the DEV can also try to join as a regular DEV; in which case, the dependent piconet would become a child piconet after the hand-over process. Although it may seem strange that the hand-over process can change a piconet from being a neighbor piconet to being a child piconet, it is important to remember that the difference between a child and a neighbor piconet lies only in the characteristics of the dependent PNC. If the dependent piconet is currently a child piconet and DEV-2 attempts to join as a neighbor, the parent PNC will be able to decide if it wants to support the piconet as a neighbor piconet. If the parent PNC does not want to allow a neighbor piconet, then it will reject the association request of the DEV as a neighbor PNC. Likewise, if the dependent piconet is currently a neighbor piconet and DEV-2 either is a member or becomes a member of the parent piconet, then the parent PNC should allow the piconet to become a child piconet.

INTEL-1010
U.S. Patent 9,265,043

The PNC has already allowed DEV-2 to become a member of the parent piconet, so it should not have any problem with DEV-2 taking over control of the dependent piconet's CTA. Once the CTA has been allocated, there is minimal additional effort required by the PNC to change the SrcID and DestID for the CTA.

If DEV-2 fails to join the parent piconet as either a regular DEV or as a neighbor PNC, it will send the PNC Handover Response command to the dependent PNC with the Reason Code set to 0xFE, which is an error code that means "Handover refused, unable to join parent piconet." Note that the error code is equal to the UnassocID, which is DEVID that the DEV-2 will receive if it is unable to associate with the parent PNC. The frame exchange leading up to the failure of DEV-2 to join the parent piconet is illustrated in Figure 5–8. As shown in the figure, DEV-2 is allowed to refuse the handover at any time while the dependent PNC is sending over the information about the piconet. DEV-2 does not have to wait until all of the information is transferred before it sends the PNC Handover Response command if it is going to reject the handover.

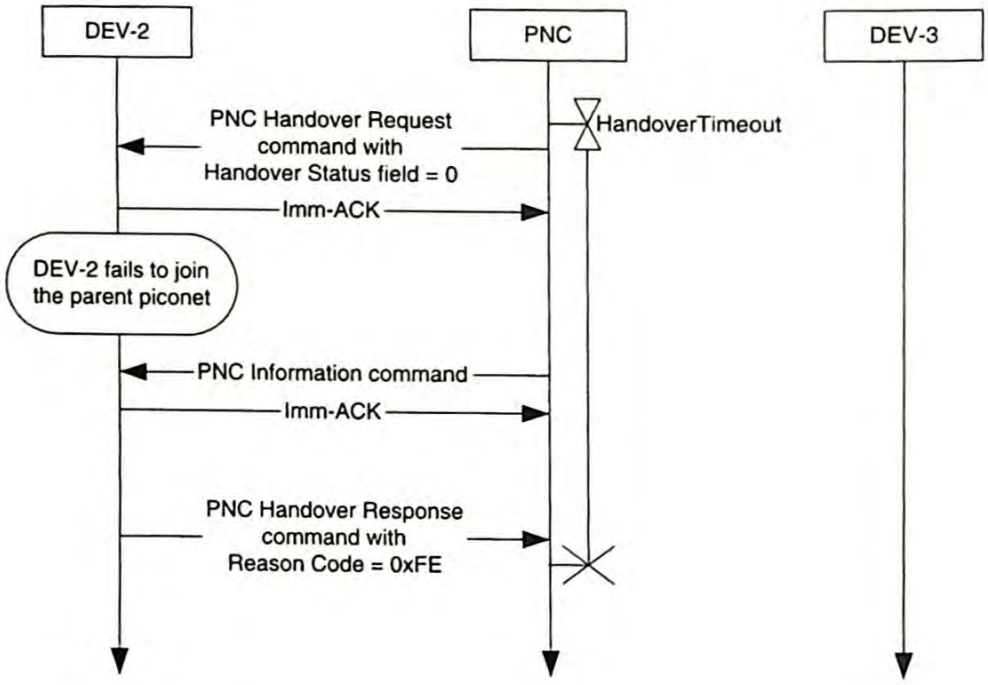


Figure 5–8: Hand-over of dependent PNC fails because new PNC fails to join the parent piconet

INTEL-1010
U.S. Patent 9,265,043

If DEV-2 is successful in joining the parent piconet, then the next problem that is unique to dependent PNC hand-over is handing over control of the CTA to the new dependent PNC. Because DEV-2 has passed its new DEVID to the current dependent PNC in the PNC Handover Response command, the current dependent PNC should not make a mistake in selecting the correct DEVID to send in the Channel Time Request command. However, the PNC may reject the handover of the CTA control if any of the following are true:

- The DEVID specified in the target ID list does not correspond to an associated DEV.
- The DEVID specified in the target ID list does not correspond either to a secure member of the piconet or a NbrID and secure membership is required in the parent piconet.
- The DEVID specified in the target ID list is in power-save mode.
- The PNC is terminating the stream.

The PNC should have time available to allocate the CTA because it is already allocating the same amount of time for the dependent piconet. If the parent PNC rejects the request to hand over control of the CTA to the new DEV-2 PNC, the dependent PNC will send a PNC Handover Request command to DEV-2 with the Handover Status field set to 1 to indicate that the hand-over process is being cancelled, as illustrated in Figure 5–9.

If the dependent PNC cancels the hand-over process, DEV-2 may choose to disassociate from the parent piconet because there is no longer a need for it to be a part of the parent piconet. If DEV-2 joined as a neighbor PNC, it should disassociate from the parent piconet when the hand-over process is cancelled to free up that resource for other DEVs that may want to form a neighbor piconet.

INTEL-1010
U.S. Patent 9,265,043