**IEEE P802.15**

**Wireless Personal Area Networks**

|  |  |  |
| --- | --- | --- |
| Project | IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs) | |
| Title | Document for Final Proposal Doc 15-13-380r2 | |
| Date Submitted | May 15, 2014 | |
| Source | [Qing Li, Chonggang Wang, Hongkun Li, Zhuo Chen, and Tao Han]  InterDigital Communication Cp. | E-Mail:  [Qing.Li@InterDigital.com]  Voice:  [610-878-5695] |
| Re: | Call for Final Proposals | |
| Abstract | This detailed document presents final proposals on the PHY/MAC system design for 802.15.8 (PAC) | |
| Purpose | To discuss technical feasibility of proposed system design for 802.15.8 (PAC) | |
| Notice | This document has been prepared to assist the IEEE P802.15. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. | |
| Release | The contributor acknowledges and accepts that this contribution becomes the property of IEEE and may be made publicly available by P802.15. | |
|  |  | |

Table of Contents

[1. Overview 3](#_Toc391476567)

[2. Definitions 3](#_Toc391476568)

[3. Abbreviations and acronyms 4](#_Toc391476569)

[4. General descriptions 5](#_Toc391476570)

[4.1. Concepts and architecture 5](#_Toc391476571)

[4.2. Topology 7](#_Toc391476572)

[4.3. Reference model 8](#_Toc391476573)

[5. MAC layer 8](#_Toc391476574)

[5.0. Overview 8](#_Toc391476575)

[5.1. 8](#_Toc391476576)

[5.2. Frame structure 8](#_Toc391476577)

[5.3. Synchronization 9](#_Toc391476578)

[5.4. Discovery 11](#_Toc391476579)

[5.5. Peering 12](#_Toc391476580)

[5.5.1. Single Hop Peering 13](#_Toc391476581)

[5.5.2. Multi-hop Peering 14](#_Toc391476582)

[5.6. Communications 14](#_Toc391476583)

[5.6.2. Multicast 14](#_Toc391476584)

[5.6.3. Multi-application Data Transmitting and Receiving 15](#_Toc391476585)

[5.7. MPDU structure 19](#_Toc391476586)

[5.7.1. General MAC Frame: 19](#_Toc391476587)

[5.7.2. Beacon Frame 20](#_Toc391476588)

[5.7.3. Peering Related Frame Structure 21](#_Toc391476589)

[5.8. Multiple access 25](#_Toc391476590)

[5.8.1. Fast Channel Accessing 25](#_Toc391476591)

[5.9. Synchronization procedure 34](#_Toc391476592)

[5.10. Discovery procedure 36](#_Toc391476593)

[5.11. QoS 37](#_Toc391476594)

[5.12. Interference management 37](#_Toc391476595)

[5.13. Power control 38](#_Toc391476596)

[5.14. Multi-hop 38](#_Toc391476597)

[5.15. Relative positioning 38](#_Toc391476598)

[5.16. Power management 38](#_Toc391476599)

[5.17. Security 45](#_Toc391476600)

[5.18. Coexistence 45](#_Toc391476601)

[5.19. Upper layer interaction 45](#_Toc391476602)

[5.19.2. Cross-Layer Context Management 50](#_Toc391476603)

[5.19.3. Cross-Layer Data Acknowledgement 50](#_Toc391476604)

# Overview

This standard defines PHY and MAC mechanism for Wireless Personal Area Networks (WPAN) Peer Aware Communications (PAC) optimized for Peer-to-Peer and infrastructure-less communications with fully distributed coordination. A PAC or Peer-to-Peer Network (P2PNW) is formed for a desired service or application within proximity. A peer can participate in multiple services or applications, i.e. multiple P2PNWs. Many P2PNWs may coexist in proximity.

PAC key features may include the following:

* Operational in selected globally available unlicensed/licensed bands below 11 GHz capable of supporting these requirements
* Scalable data transmission rates, typically up to 10 Mbps
* Discovery for peer information without Peering
* Discovery signaling rate typically greater than 100 kbps
* Discovery of the number of devices in the network
* Group communications with simultaneous membership in multiple groups, typically up to 10
* Multi-hop relay
* Relative positioning
* Security

# Definitions

**380r2 (beginning)**

|  |  |
| --- | --- |
| Channel Accessing | Procedure or action to physically connect to a physical communication channel to transmit or receive signals or data in a wireless communication system. |
| Channel Allocation | Procedure to define or assign the physical communication channel(s) to a terminal or multiple terminals for transmitting or receiving signals or data in a wireless communication system. A terminal accesses a channel allocated to it. |
| Context | Special situation information such as service, application, location, time, power, etc. |
| Context Manager | An MAC-layer function for local intra-PD context management or remote inter-PD context management. |
| Context Management | Context exchange interactions between two protocol layers within a PD or between two PDs on MAC layer. The interactions could include context information reporting, context information retrieval, context information deletion, context information subscription, and context notification. |
| Peer | A user or a device such as MS in 2G, a UE in 3GPP, an FFD or RFD in IEEE 802.15/WPAN; A peer can be a group of users or devices sharing a group ID. |
| P2P Proximity Communications | Infrastructure-based or infrastructure-less communications among peers within proximity |
| Peer Discovery | Procedure used for a peer to find another peer(s) before Peering to enable P2P proximity communications |
| Peering | Procedure used for a peer to establish logical relationship with another peer(s) before P2P data transmission can be started. It is also termed as peer attachment, peering, pairing, or link establishment. |
| Peering Identifier | A locally unique identifier to identify each established peering relationship among peers. It is assigned during Peering or Re-Peering and can be updated during Peering Update. |
| Peering Update | Procedure used for a peer to update Peering Identifier and/or Peering Context of an existing association relationship with other peer(s). |
| De-Peering | Procedures used for a peer to cancel an existing association relationship with other peer(s) |
| Re-Peering | Procedures used for a peer to Re-Peering a cancelled association relationship with other peer(s) |
| Virtual Leader | A virtual leader is a peer defined to represent, manage, and coordinate the P2P communications among a group of peers (i.e. the members of the group), sharing the same context-based service and/or application. A virtual leader may be dynamically determined and/or changed within the group. The virtual leader performs functions for the group such as context management, context-aware discovery broadcast, context-aware Peering, group membership management, synchronization, link management, channel accessing control, reliable data transmission, routing management, power control and interference management, and channel measurement coordination, etc. A peer can only be the virtual leader for one application, and one application can have only one virtual leader. Other alternative terms for virtual leader include group leader/header/controller/coordinator/master/manager, cluster leader/header/controller/coordinator/master/manager, and zone leader/header/controller/coordinator/master/manager, etc. |
| Sub Virtual Leader | A sub virtual leader is a peer defined to extend coverage through multi-hop based on the physical or logicalal topology. The roles of a sub virtual leader include: 1) as a virtual leader to manage a subgroup of peers with the same context-based service and/or application; 2) as a peer (i.e. a member) under the management of the virtual leader and/or a sub virtual leader of the same group. The sub virtual leader may perform a subset of functions of the virtual leader. |
| Super Virtual Leader | A super virtual leader is a peer defined to coordinate all virtual leaders for the purposes of synchronization, power control, interference management, and channel accessing control, etc. A super virtual leader may be dynamically determined and/or changed among the virtual leaders. The super virtual leader is the top leader of the virtual leaders’ hierarchical structure. |
|  |  |

**380r2 (end)**

# Abbreviations and acronyms

**380r2 (beginning)**

ACK Acknowledgement

APP Application

CAID Context Aware Identifier

CD Context Database

CDMF Context Dispatch Management Function

CEM Context Exchange Mode

CCDCH Common Control/Data Channel

CPCI Context and Power Control Information

CA Channel Allocation

CAc Channel Accessing

CM Context Manager

D2D Device-to-Device

DCDCH Dedicate Control/Data Channel

FFD Full-Function Device

ID Identifier

IE Information Element

IEEE Institute of Electrical and Electronics Engineers

MAC Medium Access ControlMCPS MAC Common Part Sublayer

MLME MAC subLayer Management Entity

MHR MAC HeaderPHY PhysicalPAC Peer Aware Communications

P2P Peer-to-Peer

P2PNW Peer-to-Peer Network

PD Peer Discovery

PI Peer Information

PeerReq Peer Requesting Channel Allocation or Channel Accessing

RFD Reduced-Function Device

SubVL Sub Virtual Leader

SuperVL Super Virtual Leader

VL Virtual Leader

WPAN Wireless Personal Area Network

TCP Transmission Control Protocol

VLreq VL Requesting Channel Allocation or Channel Accessing

**380r2(end)**

# General descriptions

## Concepts and architecture

****

A PAC system may contain the following functions at MAC and/or PHY.

**Higher Layer:** the layer above PHY/MAC, such as service or application layer for an infrastructure-less P2P wireless system.

**PD Management Entity:** manages PD information across PHY/MAC and higher layer for P2P communications.

**Synchronization Function:** performs initial and/or periodic time boundary synchronization with a peer at MAC; maintains frequency and/or clock phase synchronization at PHY.

**Discovery Function:** discovers peer(s) in proximity by using IDs such as Device ID, Device Group ID, Application type ID, Application-specific ID, Application-specific user ID, and Application-specific group ID, and/or peer context information; sends discovery request messages with IDs such as Device ID, Device Group ID, Application type ID, Application-specific ID, Application-specific user ID, and Application-specific group ID, and/or peer context information for to-be-discovered in proximity**.**

**Peering Function:** requests or responds to Peering (i.e. association), Peering Updates, De-peering, or Re-peering by using IDs such as Device ID, Device Group ID, Application type ID, Application-specific ID, Application-specific user ID, and Application-specific group ID, and/ or peer context information.

**Channel Management Function:** manages the radio resource or channel allocation among P2P networks based on the services or applications; manages channel allocation and/or accessing within a P2P network based on the peer information.

**Power Control Function:** performs transmitting power control and interference management based on power control information.

**Data Transceiving Function:** conducts reliable data transmitting and/or receiving based on the QoS required by service or application.

**Measurement and Report Function:** conducts measurements of channel, QoS, etc., and sends data reports from other logic functions to higher layer.

**Scheduler Function:** manages or controls the sequence of events in MAC and/or PHY layer, for example, transmission queuing, collision avoidance, etc.

**Encoder Function:** performs encoding and other data processing such as interleaving, scrambling, etc. at PHY to aid in reliable reception

**Modulator Function:** performs modulation and other data processing such as mapping, precoding, etc. at PHY

**Decoder Function:** performs decoding and other data processing, such as de-interleaving, de-scrambling, etc. at PHY to aid in reliable reception

**Demodulator Function:** performs demodulation and other data processing such as equalization, de-mapping, etc. at PHY

## Topology

**380r2 (beginning)**

Figure X: Typical P2P Network Topology

As required in TGD, a P2P network (P2PNW) is formed by a desired application. There may be multiple applications existing simultaneously, therefore multiple P2PNWs coexist at the same time in proximity. In addition, a peer may run multiple applications simultaneously, and therefore joins multiple P2PNWs at the same time. As shown in the figure X, there are 4 applications (P2PNWs), peer 6 and peer 9 join 2 P2PNWs for 2 applications, respectively. Application 3 is pair communication between two peers, while application 1 is multi-hop broadcast for commercial advertisement with a tree topology, and application 4 is gaming with a mesh topology. With many different P2PNWs co-existing in the proximity, the topology is more heterogeneous than homogenous.

**380r2(end)**

## Reference model

# MAC layer

* 1. Overview

This section defines MAC mechanism for Wireless Personal Area Networks (WPAN) Peer Aware Communications (PAC) optimized for peer-to-peer and infrastructure-less communications with fully distributed coordination.

## 

## Frame structure

**380r2 (beginning)**

#### Frame Structure for Single Hop



Figure 5-2-1: General Hierarchical Frame Structure

1. **Superframe Beacon:** start of a Superframe. It also indicates the superframe structure, such as the length of common channel, number of application frames, and length of reserved time duration. It may be used as synchronization reference also.
2. **Common Channel**: shared by all peers and applications in proximity - both public broadcasting / multicasting and private pair communications via contention based accessing.
3. **Application Frame:** dedicated to an application. There may be one or multiple Application Frames within a Superframe. An application frame consists of a dedicated channel and contention free period. Dedicated channel is shared among all peers within the application group, and is contention based for accessing. Contention free period is allocated to individual peers.
4. **Reserved Time:** reserved for the insertion of other application frames.
5. **Inactive Period:** optionally as the gap or guard time between Superframes.
6. **Hyperframe:** top level frame. It may include several Superframes. The Hyperframe structure is shown in Figure 5-2-2.



Figure 5-2-2: Hyperframe Structure

Figure 5-2-3 shows the an example of Superframe structure for TDMA.



Figure 5-2-3: Superframe structure for TDMA.

#### Frame Structure with Multi-hop Period

***Note*** *that the contribution of multi-hop frame structure is in final contribution document # 15-14-0258-00-0008.*

**380r2(end)**

## Synchronization

**380r2 (beginning)**

The synchronization procedure is to find the frame/slot boundary, and can be triggered by the request from other procedures such as Discovery, Peering, Data transceiving, higher layer etc. The synchronization procedure starts by scanning for the common synchronization reference signal, which may be carried on common beacon and or sent on common channel. Figure 5-3-1 illustrates an example of a superframe under virtually centralized control consists of the super beacon, application beacons and application frames. In the super beacon, which indicates the start of a superframe, there is a Frame Map (FM) field to indicate the application frame allocation for different applications. FM further contains “Application Offset List” (AOL) field indicating the time offset of the application frames according to the time reference of the super beacon. The time reference in the super beacon is the time information that can be used as the reference to calculate the starting time of each application frame. The structure of FM is shown inFigure 5-3-2.



Figure 5-3-1: Superframe Structure under Virtually Centralized Control



Figure 5-3-2: Structure of Frame Map in Super Beacon

In different scenarios, the peer may either find a super beacon or an application beacon first. Therefore, to collect the application information efficiently, a new field “Super Beacon Offset” (SBO) is contained in the application beacon to indicate where the Super Beacon is in the form of time offset. One possible example of SBO is illustrated in Figure 5-3-3, where the slot/symbol number between App 2 beacon and the next Super Frame is indicated in SBO.



Figure 5-3-3: Example of SBO

Based on the FM and SBO,

For hybrid control, there is no SuperVL acting as the virtually central controller to manage the applications in proximity. The individual application is managed by the VL. One of the VLs in proximity defines the time reference by sending the beacon to formulate a superframe. The beacon sent by the VL who maintains the time reference for the common channel is called Common Beacon (CB). The superframe consists of application frames for different applications with guard interval if the desired applications are found during CAIS, the peer shall synchronize with the VLs of desired P2PNWs for hybrid control. Otherwise, the peer shall synchronize with the Common Beacon

The Common Beacon contains two fields Common Beacon Offset (CBO) and Common Channel Offset (CCO) to search the common channel. For ease of searching the beginning of an application beacon, a new field Application End Offset (AEO) is included in the application beacon to indicate the end of the application frame as shown in Figure 5-3-4.



Figure 5-3-4: The Superframe under Hybrid Control with AEO example

**380r2(end)**

## Discovery

**380r2 (beginning)**

****

Figure 5-4-1: Discovery Procedure

The proposed discovery procedure consists of three main stages as shown in Figure 5-4-1:

1) Peer Discovery Scan;

2) Peer Discovery Evaluation;

3) Peer Discovery Request.

“To Discover” means that the peer wants to discover other peer(s) with a desired application in proximity; while “to be discovered” implies that the peer wants to be discovered by other peer(s) through broadcasting discovery message in proximity.

A peer enters Peer Discovery Scan first - “To Discover” a peer or peers for a desired service or application. Then, the peer enters Peer Discovery Evaluation to qualify the discovered peer(s). If no peer is discovered or qualified for the desired service or application based on certain criteria, the peer enters Peer Discovery Request - “To Be Discovered” for a desired service or application. During “To Discover” process, the peer scans the beacon or pilot/broadcast/paging channel carrying the desired context information across all available frequency channels if applicable. If any peer is detected, peer discovery evaluation stage is triggered for qualification check. The evaluation process is to check if the detected peer meets the discovery criteria, such as the signal strength. The qualified peer(s) is added to the candidate list. The last step in the evaluation stage is to determine if the peer will continue to discover more peers. If yes, the peer repeats the scan stage; otherwise the candidate list is passed to the Peering procedure. If no peer is detected during the scan stage for a configured period of time, the peer switches to “To Be Discovered” process for the desired service or application, and enters the peer discovery request stage. The peer periodically broadcasts the discovery message (e.g., beacon or discovery request frame) with the context information during the request stage. The peer continues broadcasting until either it is discovered by the required number of peers or exits this stage due to a configured timeout or policy.

The context-aware peer discovery procedure above requires scanning the context information during the peer discovery stage to determine if any peer with desired service or application is in proximity. However, the context information could contain lots of information, and scanning such information may take too much time. To reduce the overhead caused by checking each of the context components during discovery, a hierarchical Context-Aware Identifier (CAID), is used for fast context-aware peer discovery. For example, the peer checks the context category first during scan stage. If context category matches the context category of the desired service or application, then the process continues to check service ID, user ID, etc. in a sequential manner; otherwise, the peer immediately stops decoding the rest of CAID if any element does not match the desired service or application. Therefore, the peer does not need to go through all the elements of the context information for discovery.

**380r2(end)**

## Peering

**380r2 (beginning)**

### Single Hop Peering

Peering Identifier (PID) is the identity of each established link or peering relationship between two or more PDs. Considering different types of applications, link(s) may be established at different levels of peering:

* Device-based peering
* Service-based peering
* User-based peering

#### Peering Procedure



Figure 5-5-1: Peering Procedure

The peering procedure is initiated by sending a peering request message including requested peering information. Acknowledgement to peering request message may be optional for responder. Responder may send a peering response message to requestor for indicating if the peering request is accepted or not. The response message may include peering information as well as the peering ID if the request is accepted. Acknowledgement to response message may be optional as well.

#### Re-peering procedure:



Figure 5-5-2: Re-peering Procedure

Re-peering procedure is similar to peering procedure. The main difference is: 1) the peering ID should be included in request message since the re-peering is to re-establish a link or links among PDs that peered previously; 2) the PD(s) receives the request need to check the peering log before making decision if accepts the re-peering request.

#### De-peering procedure:



Figure 5-5-3: De-peering Procedure

De-peering procedure starts with a de-peering request, which is replied by a de-peering response message. Both acknowledgement and de-peering response may be optional. Accordingly, the peer may triggers peer discovery process with updated parameters after de-peering procedure.

### Multi-hop Peering

***Note*** *that the contribution of multi-hop Peering is in final contribution document # 15-14-0260-00-0008.*

**380r2(end)**

## Communications

* + 1. Unicast

### Multicast

**380r2 (beginning)**

***Note*** *that the contribution of reliable multicast is in final contribution document # 15-14-0262-00-0008*.

**380r2(end)**

### Multi-application Data Transmitting and Receiving

A peer may participate in multiple P2P services or applications. The proposed context-aware architecture enables multi-application capability for P2P communications in proximity. Examples of procedures and call flows for multi-application data transmitting and receiving are detailed in this section.

#### Multi-application Data Transmitting

A multi-application data transmitting is illustration in Figure 5-6-2, which may contain the following steps.

1. **Peer1 – Peer3 & Peer2 & Peer1: “ Start a P2P Session”**

Peer1, Peer2, and Peer3 follow the procedure of initiating a P2P session, described in Section 5.3.2.1 Initiation of P2P Communication, for Application1 and Application2 separately.

**Peer1 – Peer3: Application1**

A P2P session for Application1 is initiated with

0A. Peer1’s Upper Layer sending MLME-START-APP1.request

, and confirmed with

0B. Peer2’s Peering Function responding Upper Layer with MLME-Peering-APP1.confirm.

**Peer2 – Peer1: Application2**

A P2P session for Application2 is initiated with

0C. Peer2’s Upper Layer sending MLME-START-APP2.request

, and confirmed with

0D. Peer1’s Peering Function responding Upper Layer with MLME-Peering-APP2.confirm.

1. **Application1 Peer1: Data Transmission Request**
   1. Upper Layer sends trigger or request to the Data Transceiving Function with MCPS-DATA-APP1.request.
   2. Upper Layer down loads context related to Application1 via the Context Management Function.
2. **Application1 Peer1- Peer3: Data Transmitting**

The Data Transceiving Function sends Application1 Data1 to Peer3 via air interface.

1. **Application1 Peer3: Data Receiving** 
   1. Peer3 receives the data and notices the Upper Layer with MCPS-DATA-APP1.indication.
   2. Peer3 sends ACK for Application1 Data1 to Peer1 via air interface.
2. **Application1 Peer1: Data Transmission Confirmation**
   1. Peer1 receives the ACK and notices the Upper Layer with MCPS-DATA-APP1.confirm.
   2. Peer1 may update the context and upload the updated Application1 context to the Upper Layer via the Context Information Management.
3. **Application2 Peer1: Data Transmission Request**
   1. Upper Layer sends trigger or request to the Data Transceiving Function with MCPS-DATA-APP2.request.
   2. Upper Layer down loads context related to Application2 via the Context Management Function.
4. **Application2 Peer1- Peer2: Data Transmitting**

The Data Transceiving Function sends Application2 Data1 to Peer2 via air interface.

1. **Application2 Peer2: Data Receiving** 
   1. Peer2 receives the data and notices the Upper Layer with MCPS-DATA-APP2.indication.
   2. Peer2 sends ACK for Application2 Data1 to Peer1 via air interface.
2. **Application2 Peer1: Data Transmission Confirmation**
   1. Peer1 receives the ACK and notices the Upper Layer with MCPS-DATA-APP2.confirm.
   2. Peer1 may update the context and upload the updated Application2 context to the Upper Layer via the Context Information Management.



Figure 5-6-2 Multi-application Data Transmitting

#### Multi-application Data Receiving

A multi-application data transmitting and receiving is illustration in Figure 5-6-3, which may contain the following steps.

1. **Peer1 – Peer3 & Peer2 & Peer1: “ Start a P2P Session”**

Peer1, Peer2, and Peer3 follow the procedure of initiating a P2P session, described in Section 5.3.2.1 Initiation of P2P Communication, for Application1 and Application2 separately.

**Peer1 – Peer3: Application1**

A P2P session for Application1 is initiated with

0A. Peer1’s Upper Layer sending MLME-START-APP1.request

, and confirmed with

0B. Peer2’s Peering Function responding Upper Layer with MLME-Peering-APP1.confirm.

**Peer2 – Peer1: Application2**

A P2P session for Application2 is initiated with

0C. Peer2’s Upper Layer sending MLME-START-APP2.request

, and confirmed with

0D. Peer1’s Peering Function responding Upper Layer with MLME-Peering-APP2.confirm.

1. **Application1 Peer1: Data Transmission Request**

Upper Layer sends trigger or request to the Data Transceiving Function with MCPS-DATA-APP1.request for Application1 Data1.

1. **Application1 Peer1 - Peer3: Data Transmitting**

The Data Transceiving Function sends Application1 Data1 to Peer3 via air interface.

1. **Application2 Peer2: Data Transmission Request**

Upper Layer sends trigger or request to the Data Transceiving Function with MCPS-DATA-APP2.request for Application2 Data1.

1. **Application2 Peer2 - Peer1: Data Transmitting**

The Data Transceiving Function sends Application2 Data1 to Peer1 via air interface.

1. **Application1 Peer3: Data Receiving** 
   1. Peer3 receives the Application1 Data1from Peer1 and notices the Upper Layer with MCPS-DATA-APP1.indication.
   2. Peer3 sends ACK for Application1 Data1 to Peer1 via air interface.
2. **Application2 Peer1: Data Receiving** 
   1. Peer1 receives the Application2 data1 from Peer2 and notices the Upper Layer with MCPS-DATA-APP2.indication.
   2. Peer1 sends ACK for Application2 Data1 to Peer2 via air interface.
3. **Application1 Peer1: Data Transmission Confirmation**

Peer1 receives the ACK from Peer3 for Application1 Data1 and notices the Upper Layer with MCPS-DATA-APP1.confirm.

1. **Application2 Peer2: Data Transmission Confirmation**

Peer2 receives the ACK from Peer1 Application2 data1 and notices the Upper Layer with MCPS-DATA-APP2.confirm.



Figure 5-6-3 Multi-application Data Transmitting and Receiving

## MPDU structure

**380r2 (beginning)**

### General MAC Frame:

Figure 5-7-1 MPDU frame structure.

**Frame Type and Subtype** fields together indicate the type of a frame, i.e., the function of a frame.

**Required ACK Type** field in frame control specifies what type of acknowledge frame is expected. For example, no ACK or aggregated ACK.

**Addressing** field Indication field indicates the presence of the transmitting hop address and receiving hop address in the addressing fields.

Addressing fields consist of the following addresses: source address, destination address, transmitting hop address and receiving hop address. Transmitting hop address and receiving hop address are optional.

**P2PNW/APP ID** field shows the P2P network ID or application ID in the MHR. All the peers joining a P2PNW will have a locally unique P2PNW/APP ID. If P2PNW ID is not determined when a frame is sent, this field will carry application ID.

**Application Type** field indicates the application/service category, such as emergency service, social networking, smart office, etc.

**Hopper Indication** is used to indicate if the frame sender is willing to relay other frames for the multi-hop discovery.

### Beacon Frame

Beacon frame plays an important role in forming a P2PNW and enabling the P2P communication. It could be used to carry context information for the discovery procedure, to define a new superframe and/or application frame through channel management process, to determine the frame/slot boundary for synchronization and to facilitate the power control procedure.



Figure 5-7-2 Beacon frame structure

Frame information is a part of the beacon payload, and consists of two components:

* Superframe information: defines a new superframe
* Application frame information: defines the structure of an application frame.

### Peering Related Frame Structure

#### Peering Request Frame

The format of Peering Request frame is shown in Figure 5-7-3, where all the listed fields as part of MAC payload are mandatory. The optional fields are grouped in other MAC payload.



Figure 5-7-3 Peering Request frame structure.

Device capability could be the different types of capability of the peer that is sending the request. For example, the transmission data rate capability, battery/power consumption capability or security capability.

The Peering type field indicates what type of Peering is expected to establish. The Peering is classified as device-based, service-based and user-based Peering. A peer may maintain multiple applications, and therefore may maintain multiple different types of Peering connections.

Required duration field is set by the requestor to indicate how long the Peering connection is expected to be active.

VL indication explicitly shows if the sender of the request is VL or not.

Response type field is used to indicate what optional fields are required in other MAC payload as part of the corresponding Peering Response message.

Multi-hop indication indicates if the Peering Request is relayed for a peer outside one-hop range of receiver, i.e., multi-hop association.

#### Peering Response Frame

Figure 5-7-4 Peering Response frame structure

Peering ID is the identifier that identifies an association between two peers.

Peering decision indicates if the Peering Request is accepted or not**.**

Assigned duration indicates the lifetime of the Peering to establish. The responder makes the decision based on the required duration in Peering request. It could be different with the required duration.

Assigned short address contains the short address if short address required field is true in the request message.

#### Re-Peering Request Frame

Re-Peering Request frame has the very similar structure with the Peering Request.

#### Re-Peering Response Frame

Re-Peering Response frame shares the same structure and fields with Peering Response frame.

#### De-Peering Request Frame



Figure 5-7-5 De-Peering Response frame structure

De-Peering Request is sent to notify the other side of the association that the association will be shut off soon. The required ACK type in frame control will indicate if the De-Peering Response message is required or not.

De-Peering reason field indicates why the Peering is going to be shut off. The possible reason includes: link failure, application termination or resource limitation.

De-Peering duration field indicates the time duration of the De-Peering. This means that after the time duration, the association will be active.

#### De-Peering Response Frame

De-Peering Response frame is to confirm that the association is disconnected after receiving the De-Peering Request.



Figure 5-7-6 De-Peering Response frame structure

De-Peering status indicates the association is disconnected permanently or just for a time period, which means the association will be activated then.

#### Peering Update Notification Frame

Peering Update Notification frame is to notify the other side that some attributes of an existing Peering needs to update.



Figure 5-7-7 Peering Update Notification frame structure

Updated Peering information field may include one or multiple information fields about an existing Peering that needs to update. Any field either mandatory or optional in Peering Request and Response frames could be included in the updated Peering information field.

#### Peering Update Response Frame



Figure 5-7-8 Peering Update Response frame structure

Peering Update Response is used to confirm the update of some Peering attributed noted by Peering Update Notification frame.

Updated status field indicates if all of the requested updated Peering information is updated or not. It could be fully updated, partially updated or rejected at all.

Updated Peering information field includes two parts of Peering information:

* the request updated Peering information in Peering Update Notification frame but not updated
* the Peering information that is required to update by the sender of Peering Update Response.

#### Channel Management Frame

* + - * 1. Inter-P2PNWs Channel Allocation Request

Inter-P2PNWs channel allocation request frame as shown in Figure 5-7-9 is used to broadcast a request on the CCDCH in the proximity for radio resource allocation.



Figure 5-7-9 Inter-P2PNWs Channel Allocation Request Frame

VL indication field shows whether the sender is VL or not. In distributed control, this field is always false.

Desired application frame length indicates the desired time duration of the application frame that the sender is trying to construct.

SuperVL willingness indicates if the sender is willing to act as SuperVL. This field is mandatory

Desired application beacon location is an optional field, which indicates when the application beacon is broadcast. This field is present only when the sender has the knowledge of the superframe structure and already synchronizes with the P2PNW.

* + - * 1. Inter-P2PNWs Channel Allocation Response



Figure 5-7-10 Inter-P2PNWs Channel Allocation Response Frame

Inter-P2PNWs channel allocation response comes after with Inter-P2PNWs channel allocation request frame on CCDCH shown in Figure 5-7-10.

SuperVL indication shows if the response is sent from a superVL or not. In hybrid and distributed control, this field is always false.

Response decision explicitly indicates if the corresponding Inter-P2PNWs channel allocation request is accepted or not.

Reject reason is to indicate the reason why the request is rejected. For example, the requested time period is totally or partially overlapped with a time period that has been allocated to an application frame.

Adjustment suggestion is an optional field, and could include the suggestion on where the available time period is. The adjustment suggestion will be given higher priority if the response is from the SuperVL.

* + - * 1. Intra-P2PNWs Channel Allocation Request

#### 

Figure 5-7-11 Inter-P2PNWs Channel Allocation Request Frame

Intra-P2PNWs channel allocation request frame sent over DCDCH is used to request one or more time slots in an application frame, as shown in Figure 5-7-11.

It is assumed that the sender knows the application frame specification when broadcasting the Intra-P2PNWs channel allocation request.

SubVL indication field shows if the requestor is the SubVL or a peer. In distributed control, this field is always setup as a peer.

Desired number of slots indicated how many time slots the sender requests. The slot size is fixed during the whole application frame, a peer could only request for different number of slots for transmission.

Desired slot location is an optional field, which indicates the position of the desired time slots in the application frame.

* + - * 1. Intra-P2PNWs Channel Allocation Request



Figure 5-7-12 Inter-P2PNWs Channel Allocation Response Frame

Intra-P2PNWs channel allocation response message shown in Figure 5-7-12 is sent as reply to the Intra-P2PNWs channel allocation request.

VL indication shows if the response message is sent by the VL or not. In distributed control, this field is always false.

Response decision, reject reason and adjustment suggestion fields have the same usage as those in Inter-P2PNWs channel allocation response frame.

**380r2(end)**

## Multiple access

**380r2 (beginning)**

**NOTE**: *this is the procedure for channel management is considered as part of Multiple Access* .

### Fast Channel Accessing

As illustrated in Figure 5-8-1, the fast CCDCH accessing scheme may be conducted in the following steps as an example.

1. Fast CCDCH accessing is triggered either by the application from a higher layer or by other logical functions such as Peer Discovery (PD), Peering, Synchronization Request (SR), Channel Accessing (CAc), Power Control (PC) and Interference Management (IntM), etc.
2. Scan if CCDCH is occupied for a predefined time window tScanCCDCH (across the superframe boundary).
3. If CCDCH is occupied, wait for tCCDCH and then check if predefined timer, tOutCCDCH, has timed out. If not timed out, go back to step 2 and check if CCDCH is occupied again; if timer has expired, abort and notify the high layer or logical function(s) which triggered the fast CCDCH accessing.
4. If CCDCH is not occupied
   * If the SuperVL requested the fast CCDCH accessing for a centralized inter-P2PNWs control in proximity, the SuperVL can access the CCDCH immediately, as shown in dotted lines in Figure 5-8-1, with SuperVLinfo, its CCDCH usage, and control/data messages etc. – highest priority.
   * If the VLi of P2PNWi requested the fast CCDCH accessing for a centralized or hybrid inter-P2PNWs control, VLi shall wait for tVLi (tVLi > 0, defined for VLi based on channel accessing priority, QoS of application i, etc.) and then scan if CCDCH is available for accessing.
     + If available, the requesting VLi accesses the CCDCH with VLi info, its CCDCH usage, and control/data messages etc.
     + If not available, checks if predefined timer has timed out, defined by tOutVLi for VLi.
       - If timer has not expired, continues scanning the CCDCH until either CCDCH is available or timed out;
       - if timer has expired, the VLi has the option of broadcasting on its DCDCHi (following the DCDCH accessing procedure depicted in Figure 5-8-2) if necessary, aborts CCDCH accessing and notifies the higher layer or logical function(s) which triggered the fast CCDCH accessing.
   * If the SubVLik of P2PNWi requested the fast CCDCH accessing for a centralized or hybrid inter-P2PNWs control, SubVLik shall wait for tSubVLik (tSubVLik > tVLi, defined for SubVLik based on channel accessing priority, QoS of application i, etc.) and then scans if CCDCH is available.
     + If available, the requesting SubVLik accesses the CCDCH with SubVLik info, its CCDCH usage and control/data messages etc.
     + If not available, checks if predefined timer has timed out, defined by tOutSubVLik for SubVLik.
       - If timer has not expired, continues scanning the CCDCH until either CCDCH is available or timed out;
       - If timer has expired, the SubVLik has the option of broadcasting on the DCDCHi of P2PNWi (following the DCDCH accessing procedure depicted in Figure 5-8-2) to request VLi to access CCDCH with higher priority, aborts CCDCH accessing and notifies the higher layer or logical function(s) which triggered the fast CCDCH accessing.
   * If the Peerip of P2PNWi requested the fast CCDCH accessing for a centralized, hybrid or distributed inter-P2PNWs control, Peerip shall wait for tPeerip (tPeerip >tSubVLik > tVLi, defined for Peerip based on channel accessing priority, QoS of application i, etc.) and then scans if CCDCH is available.
     + If available, the requesting Peerip accesses the CCDCH with Peerip info, its CCDCH usage, and control/data messages etc.
     + If not available, checks if predefined timer has timed out, defined by tOutPeerip for Peerip.
       - If timer has not expired, continues scanning the CCDCH until either CCDCH is available or timed out;
       - If timer has expired, the Peerip has the option of broadcasting on DCDCHi of P2PNWi (following the DCDCH accessing procedure depicted in Figure 5-8-2) to request VLi to access CCDCH with higher priority, aborts CCDCH accessing and notifies the higher layer or logical function(s) which triggered the fast CCDCH accessing.

 **Figure 5-8-1 Fast Channel Accessing for Inter-P2PNWs Communications through CCDCH**

As illustrated in Figure 5-8-2, the fast DCDCH accessing scheme may be conducted in the following steps as an example.

1. Fast DCDCHi accessing is triggered either by the application from a higher layer or by other logical functions such as PD, PA, SR, CAc, PC, IM, etc.
2. Scan if DCDCHi is occupied within P2PNWi for a predefined time window tScanDCDCHi (across framei boundary).
3. If DCDCHi is occupied, wait for tDCDCHi and check if predefined timer tOutDCDCHi has timed out. If timer has not expired, go back to step 2 and check if DCDCHi is occupied again; if timer has expired, abort and notify the high layer or logical function(s) which triggered the fast DCDCHi accessing.
4. If DCDCHi is not occupied

* If the VLi requested the fast DCDCHi accessing for a centralized intra-P2PNW control, as shown in dotted lines, the VLi can access DCDCHi immediately with VLi info, its DCDCHi usage, and control/data messages etc. – highest priority.
* If SubVLik of P2PNWi requested the fast DCDCHi accessing for a centralized intra-P2PNW control with multi-hops, as shown in dotted lines, SubVLik shall wait for tDSubVLik (tDSubVLik > 0, defined for SubVLik based on channel accessing priority, QoS of application i, etc.) and then scan if DCDCHi is available.
  + If available, the requesting SubVLik accesses DCDCHi with SubVLik info, its DCDCHi usage and control/data messages etc.
  + If not available, checks if predefined timer has timed out, defined by tDOutSubVLik for SubVLik.
    - If timer has not expired, continues scanning the DCDCHi until either DCDCHi is available or timed out;
    - If timer has expired, aborts DCDCHi accessing and notifies the higher layer or logical function(s) which triggered the fast DCDCHi accessing.
* If the Peerip of P2PNWi requested the fast CCDCH accessing for a centralized or distributed intra-P2PNW control,, Peerip shall wait for tDPeerip (tDPeerip >tDSubVLik, defined for Peerip based on channel accessing priority, QoS of application i, etc.) and then scan if DCDCHi is available.
  + If available, accesses DCDCHi with Peerip info, its DCDCHi usage, and control/data messages etc.
  + if not available, checks if predefined timer has timed out, defined by tDOutPeerip for Peerip.
    - If timer has not expired, continues scanning DCDCHi until either DCDCHi is available or timed out;
    - If timer has expired, aborts DCDCHi accessing and notifies the higher layer or logical function(s) which triggered the fast DCDCHi accessing.



Figure 5-8-2 Fast Channel Accessing for Intra-P2PNW Communications through DCDCH

As illustrated in Figure 5-8-3, CA with P2PNW detection for a centralized control scheme (centralized inter-P2PNWs and intra-P2PNW as shown in Figure 4) may be conducted in the following steps as an example.

1. CA with P2PNW detection is triggered either by the application from a higher layer or by other logical functions such as Peer Discovery (PD), Peering, Channel Accessing (CAc) request, etc.
2. **SuperVL Detection:** the VL requesting CA (VLreq) scans the beacon/paging/broadcast for an existing SuperVL, for centralized P2PNWs control in proximity, until the SuperVL is found or a predefined timer tOutScanSpVL has expired, .
3. **CA by SuperLV:** if the SuperVL is detected in proximity, the SuperVL will manage the CA – centralized inter-P2PNWs control.

* The VLreq defines its proposed superframe/frame, slot/code/subcarriers etc. based on its application, QoS etc., and then broadcasts the request on the CCDCH (following the fast CCDCH accessing scheme defined in section 5.2.1.1) to the SuperVL. The VLreq then waits for the SuperVL’s response on the CCDCH or for the predefined timer tOutChReq to expire.
  + If the predefined timer, tOutChReq, expires, the VLreq aborts the CA request and reports to the higher layer or function(s) that triggered the CA request.
  + If the VLreq receives an “accepted” response by the SuperVL, the VLreq accesses the channel allocated by the SuperVL, and broadcasts with superframe/frame, slot/code/subcarriers etc., on the beacon/paging/broadcasting - indicating the newly formed P2PNW in the proximity logically led by VLreq as centralized intra-P2PNW control.
  + If the VLreq receives a “rejected” response by the SuperVL, the VLreq may adjusts the requested superframe/frame, slot/code/subcarriers etc., based on SuperVL’s rejection response, and re-broadcasts the request to SuperVL on CCDCH and then wait for the response from SuperVL if the predefined timer, tOutChReq, has not expired.

1. **CA for SuperVL or First VL with Collision Avoidance:** if no SuperVL is detected in proximity, VLreq may assign itself as the SuperVL (the first VL in proximity as the default SuperVL) and insert the CCDCH. But there may be a collision scenario, if there are other VL(s) assuming the SuperVL claim and starts to insert the CCDCH at the same time. To avoid this possible collision, the VLreq shall conduct a Collision Avoidance procedure outlined below.
   * **Collision Avoidance:** the VLreq defines the CCDCH and its superframe/frame, slot/code/subcarriers etc. based on its application, QoS etc., and then scans the CA request on CCDCH for a time window of tScanCCDCH.
     + If CCDCH is detected, VLreq updates its CCDCH based on the detection, and check if it’s from the SuperVL.

* If the SuperVL sent the CCDCH (just formed while the VLreq is planning to do so), VLreq conducts CA by SuperLV as described in step 3.
* If a VL sent the CCDCH (the first VL chose not to be the default SuperVL), the VLreq responds with “accept” to the first VL’s request, updates its VL list with the first VL detected, adjusts its superframe/frame, slot/code/subcarriers etc. based on the detection, assigns itself as the SuperVL if it wishes, and broadcasts the updated CA request on CCDCH until the predefined time tOutChReq has expired or receives a responses from other VL(s).
* If no CCDCH is detected, the VLreq is the first VL in proximity, and may assign itself as the default SuperVL if it wishes and then inserts the CCDCH with the CA request.
* If no response (there is no other VL(s) or P2PNW(s)) or “accept” response(s) from other VL(s) are received, the VLreq is granted with CA as the SuperVL as centralized inter-P2PNWs control.
* If the response is “reject” from other VL(s), VLreq updates its VL list and its superframe/frame, slot/code/subcarriers etc. based on the rejection(s) and continues the CA request until timed out by tOutChReq or receives “accept” from all the VLs on its VL list.
* If granted the CA, the VLreq accesses the channel allocated, and broadcasts the superframe/frame, slot/code/subcarriers etc. on the beacon/paging/broadcasting - indicating the newly formed P2PNW in the proximity logically led by VLreq, as well as itself as the SuperVL if accepted by all VLs in proximity.
* If the predefined time tOutChReq expires, the VLreq aborts the CA request and reports to the higher layer or function(s) that triggered the CA request.



Figure 5-8-3 Inter-P2PNWs CA with P2PNW Detection for Centralized Control

The CA with P2PNW cooperation for distributed control is illustrated in Figure 5-8-4 and the following steps may be conducted as an example.

1. CA with P2PNW cooperation is triggered either by the application from higher layer or by other logical functions such as PD, PA, CAc request, etc.
2. **P2PNW Peer Detection:** the Peer Requesting CA (PeerReq) scans peer(s) with a fully distributed P2PNW control in proximity until the Peer(s) are found or a predefined timer, tOutScanP2P, has expired.
3. **CA for the First Peer with Collision Avoidance:** if no peer is detected in proximity, PeerReq may set itself as the first peer in proximity and insert the CCDCH. But there may be a collision scenario, if there are other peers conducting the CCDCH insertion at the same time. To avoid the possible collision, the PeerReq conducts a Collision Avoidance procedure as outlined below.
   * **Collision Avoidance: the PeerReq defines the C**CDCH and its superframe/frame, slot/code/subcarriers etc. based on its application, QoS etc., and then scans the CA request on CCDCH for a time window of tScanCCDCH.

* **P2PNW Cooperation:** If CCDCH is detected, PeerReq updates it’s CCDCH based on the detection, responds to the request with “accept” to the first peer’s request, update**s** its peer list with the first peer detected, adjusts superframe/frame, slot/code/subcarriers etc. for peer(s) to make adjustments based on the detection, and broadcasts the updated CA request on CCDCH until timed out or receives “accept” from all the peers on its peer list.
* If no CCDCH is detected, the PeerReq is the first peer in proximity and inserts the CCDCH with its CA request.
* If no response (there is no other peer(s) or P2PNW(s)) or “accept” response(s) are received from other peer(s), the PeerReq is granted with CA.
* **P2PNW Cooperation:** If the response is “reject” from other peer(s), PeerReq updates the peer list and superframe/frame, slot/code/subcarriers etc. for peer(s) to make adjustments based on the rejection(s) and broadcasts the updated CA request until timed out by tOutChReq or receives “accept” from all the peers on its peer list.
* If granted CA, the PeerReq accesses the channel allocated, and broadcasts the superframe/frame, slot/code/subcarriers etc. on the beacon/paging/broadcasting – indicating itself in proximity. The other peers also make adjustment with the superframe/frame, slot/code/subcarriers etc. accordingly
* If timed out with tOutChReq, the PeerReq aborts the CA request and reports to the higher layer or function(s) that triggered the CA request.

1. **CA for a New Peer:** if a peer or peer(s) are detected, the PeerReq may conduct the CA request as a new peer in the proximity.

* The PeerReq defines superframe/frame, slot/code/subcarriers etc. for peer(s) based on its application, QoS etc., and detected usage of them. Then PeerReq broadcasts the request on the CCDCH (following the fast CCDCH accessing scheme defined in section 5.2.1.1) and waits for other peer’s response on CCDCH or until a predefined timer tOutChReq has expired.
  + If timed out by tOutChReq, the PeerReq aborts the CA request and reports to the higher layer or function(s) that triggered the CA request.
  + If responded with “accepted” by all peers on its peer list, the PeerReq accesses the channel allocated and broadcasts with superframe/frame, slot/code/subcarriers etc. on beacon/paging/broadcasting - indicating itself in proximity. The other peers also make adjustment with the superframe/frame, slot/code/subcarriers etc. accordingly
  + **P2PNW Cooperation:** If responded with “rejected” by peer(s), the PeerReq adjusts the requested frame, slot/code/subcarriers for peer(s) to make adjustments based on peer’s rejection response, and re-broadcasts the request on CCDCH to peer(s) if not timed out by tOutChReq.



Figure 5-8-4 Inter-P2PNWs CA with P2PNW Cooperation for Distributed Control

**380r2(end)**

## Synchronization procedure

**380r2(beginning)**

Figure 5-9-1 illustrates the Context Aware Initial Synchronization (CAIS) under virtually centralized control. The following are some more details about the CAIS:

* If a beacon is received successfully, it will be processed according to the beacon type. If the beacon is a super beacon, the superframe boundaries and the AOL in Frame Map are extracted and passed to the triggering entity. If the beacon is an application beacon, the SBO is extracted to search the super beacon. If a super beacon is not found, the scan will repeat maximum times and then scan the application beacons without AOL. Otherwise, a super beacon is processed as described above. If a received beacon is neither a super beacon nor an application beacon, it shall continue to scan if the maximum scan number is not reached.
* The Application Indicator (AI) is to indicate the criterion/result of selecting applications. For example, if CAIS is triggered by PD, PD can indicate the desired application or application ID to CAIS. The AI can be passed from the triggering entity as a part of the context information at the beginning of CAIS, or received through the interaction between CAIS and the triggering entity. The AI can also indicate whether the applications are desired or not based on AOL. The part in blue is optional if AI is a parameter accessible by CAIS.
* If the AOL is extracted from the super beacon, the scan with AOL is performed. Based on the AI, the time offsets of the desired applications are extracted and the scan function is triggered to find the desired application beacons. The frame/slot boundaries of these applications are extracted passed to the triggering entity.
* The CAIS can be terminated by the triggering entity through the synchronization acknowledgement if the desired application beacon has been found.
* If the desired application is not found during scan, the peer gets synchronized with a beacon according to some criteria. For example, the peer can synchronize with the super beacon or the strongest beacon detected. The criteria can be per-determined or a result of the acknowledgement received from the triggering entity.

Note: the CAIS procedure also applies to the multi-hop scenarios.

****

Figure 5-9-1: Context Aware Initial Synchronization (CAIS) under Virtually Centralized Control

CAIS procedure under hybrid control is presented in Figure 5-9-2.



Figure 5-9-2: CAIS Procedure for Hybrid Control

**380r2(end)**

## Discovery procedure

**380r2(beginning)**

To further speed up the PD Scan in the PD process, a hierarchical CAID scanning approach for service-based is detailed in Figure 5-10-1. For user-based, or device-based, the corresponding ID of each level will be used in Figure 5-10-1.



Figure 5-10-1: Fast scan procedure for Discovery

**380r2 (end)**

## QoS

## Interference management

## Power control

**380r2 (beginning)**

***Note*** *that the contribution of power control is in final contribution document # 15-14-0266-00-0008*.

**380r2(end)**

## Multi-hop

## Relative positioning

## Power management

**380r2 (beginning)**

A typical power management for P2P communications may contain the following operational states as shown in Figure 5-16-1.



Figure 5-16-1: The state machine of the power management

1. **Idle**: The Peer waits for data transceiving request at this state after a successful Peering, Peering Update, Re-Peering, or data transceiving.

Exit the “Idle” State

* 1. The Peer exits the “Idle” state and transitions to the “Data Transceiving” state after receiving data transmitting or receiving signal from Higher Layer.
  2. The Peer exits the “Idle” state and transitions periodically (i.e. either provisional or defined at “Peering” state ) to the “Peering Update” state for maintaining the association with current peer while not transceiving data.
  3. The Peer exits the “Idle” state and transitions to the “Sleep” state for saving power as a result of Higher Layer’s “Sleep” command.
  4. The Peer exits the “Idle” state and transitions to the “De-Peering” state as a results of a “De-Peering” Request from either Higher Layer or Air Interface due to the peer’s mobility, channel condition, etc.

1. **Data Transceiving**: The Peer enters “Data Transceiving” state from the “Idle” state. The Peer conducts data transmitting or receiving with the other peer via Air Interface at this state.

Exit the “Data Transceiving” State

* 1. The Peer exits the “Data Transceiving” state and transitions back to the “Idle” state after a successful data transmitting or receiving;
  2. The Peer exits the “Data Transceiving” state and transitions to the ‘De-Peering” state after low QoS or a failure of data transceiving due to the peer’s mobility, channel condition, etc.

1. **RE-PEERINGPeering Update**: The Peer enters the “Peering Update” state from the “Idle” state or the “Sleep” state. The Peer updates the current association with the Peering Update request and/or response with the current peer via the Air Interface.

Exit the “Peering Update” State

* 1. The Peer exits the “Peering Update” state and transitions back to the “Idle” state after a successful “Peering Update” requested from the “Idle” state to wait for next data transceiving.
  2. The Peer exits the “Peering Update” state and transitions to the “Idle” state after a successful “Peering Update” requested from the “Sleep” state as a result of a Higher Layer’s “Wake up” request.
  3. The Peer exits the “Peering Update” state and transitions back to the “Sleep” state after a successful “Peering Update” requested from the “Sleep” state as a result of timed “Wake up”.
  4. The Peer exits the “Peering Update” state and transitions to the “Re-Peering” state to establish a new link after an unsuccessful “Peering Update” with the current link.

1. **Sleep**: The Peer enters the “Sleep” state from the “Idle” state either due to a timed out while in the “Idle” state (i.e. after a predefined time interval at the “Idle” state), or as directed by a Higher Layer’s “Sleep” command.
   1. The Peer may periodically enter the “Peering Update” state defined by a wake up timer, or enter the “Peering Update” state as a result of a Higher Layer’s “Wake up” command.
   2. The Peer may transition to the “De-Peering” state after a predefined time interval without any data transceiving activity (i.e. after the sleep mode is expired), or as a result of a Higher Layer’s “De-Peering” command.
2. **De-Peering**: The Peer conducts De-Peering request and/or response with the current peer via Air Interface at the “De-Peering” state, and requests “Channel De-allocation” to release the link resources through the “Channel Management” state.

The Peer enters “De-Peering” state

1. from the “Data Transceiving” state due to low QoS or a failure of data transceiving;
2. from the “Idle” state due to a De-Peering request from Higher Layer or Air Interface;
3. from the “Sleep” state due to an expiration of sleep mode or a Higher Layer’s De-Peering request.

The Peer exits the “De-Peering” state

1. to the “Re-Peering” state as a result of a Higher Layer’s “Resume” (i.e. Re-Peering) command;
2. to the “To Discover” state as a result of a Higher Layer’s “Discover New Peer” command;
3. to “End” Application i as a result of a Higher Layer’s “End Application i” command.
4. **Channel Management**: de-allocates or releases the channel per the request from “De-Peering” state.
5. **Re-Peering**: The Peer conducts Re-Peering request and/or response with the current peer via Air Interface. The Peer may request “Channel Allocation” for sending Re-Peering messages or re-establishing the link or channel with the current peer for data transceiving through the “Channel Management” state if needed.

The Peer enters the “Re-Peering” state

1. from the “Peering Update” state due to a failure of updating the current link, i.e. the current association;
2. from the “De-Peering” state as a result of a “Resume” command from Higher Layer.

Channel for Re-Peering Request

* 1. *Designated Channel*: The Peer may request a designated channel to send the “Re-Peering” message through the “Channel Management” state.
  2. *Common, Dedicated or Public Channel*: The Peer may send “Re-Peering” message on a known or predefined common, dedicated, or public channel and may skip the “Channel Management” state for channel allocation.

Channel for Data Transceiving

1. The Peer may request a radio link or channel for the P2P data transceiving during the Re-Peering through the “Channel Management” state for intra-P2PNW channel accessing. The previously allocated channel needs to be de-allocated before a new channel is assigned for the data transceiving.
2. The Peer may also use predefined or previously used radio link or channel for the P2P data transceiving and skip the “Channel Management”.

The Peer exits the “Re-Peering” state

1. to the “Idle” state after a successful Re-Peering;
2. to the “To Discover” state to find a new peer after an unsuccessful Re-Peering with the current peer.

As shown in Figure 5-16-2, the power management call follow for updating connections with idle mode are exampled in the following call flow.

1. **Peer1 & Peer2: Idle**

Peer1 and Peer2 have a P2P session, i.e. Application 1, established and are in the Idle mode after a successful data transceiving.

1. **Peer1: Peering Update Request**

An Peering Update is inserted either by a predefined Peering Update timer or a High Layer’s Peering Update Request as shown in Figure 6.

1. **Peer1 – Peer2: Peering Update Request**

Peer1 sends an Peering Update request to Peer2 via Air Interface.

1. **Peer2: Peering Update Indication**

Peer2’s Peering Function indicates to its Higher Layer that an Peering Update request is received.

1. **Peer2: Peering Update Response**

Peer2’s Higher Layer returns an Peering Function Response to its Peering Function that the Peering Update request is acknowledged.

1. **Peer2 – Peer1: Peering Update Response**

Peer2 sends an Peering Update response to Peer1 via Air Interface to acknowledge the Peering Update.

1. **Peer1: Peering Update Confirmation**

Peer1’s Peering Function confirms to its Higher Layer that the Peering Update is successful.

1. **Peer1 & Peer2: Idle**

Peer1 and Peer2 return back to Idle.



Figure 5-16-2: The power management call follow for updating connections with idle mode.

As shown in Figure 5-16-3, the power management call follow for updating connections with sleep mode are exampled in the following call flow.



Figure 5-16-3: The power management call follow for updating connections with sleep mode.

1. **Peer1: Sleep Request**

A Sleep is enabled either by a predefined Idle monitoring timer (i.e. the Idle is expired), or a High Layer’s Sleep Request as shown in Figure 6.

1. **Peer1 – Peer2: Sleep Request**

Peer1 sends Sleep request to Peer2 via Air Interface.

1. **Peer2: Sleep Indication**

Peer2’s Higher Layer is notified that a Sleep request is received.

1. **Peer2: Sleep Response**

Peer2’s Higher Layer returns Sleep Response to acknowledge it.

1. **Peer2 – Peer1: Sleep Response**

Peer2 sends Sleep response to Peer1 via Air Interface to acknowledge the Sleep request.

1. **Peer1: Sleep Confirmation**

Peer1’s Higher Layer is confirmed about the Sleep request.

1. **Peer1 & Peer2: Sleep**

Peer1 and Peer2 set their sleep timers accordingly and enter into Sleep for some time.

1. **Peer1 & Peer2: Timed Wake Up**

Peer1 and Peer2 wake up by the sleep timers’ expiration.

1. **Peer1 & Peer2: Peering Update**

Peer1 and Peer2 performs Peering Update as described above in step 1 ~ 6 .

1. **Peer1 & Peer2: Sleep**

Peer1 and Peer2 return back to Sleep.

**380r2(end)**

## Security

## Coexistence

## Upper layer interaction

**380r2 (beginning)**

Context Management

#### One-Hop Remote Context Exchange



Figure 5-19-1 One-hop remote context exchange architecture

Figure 5-19-1 illustrates the proposed basic context management architecture for proximity communications, where:

* Context Manager (CM) is a MAC-layer function and resides in each peer.
* The CM in a peer maintains a context database which may contain context information about this peer and other peers. The CM can issue context management related requests to other CMs. The CM can also receives requests from other CMs and make responses. The CM can also receives requests from local higher layer, local MAC functions, and local PHY layer, and make responses.
* CM can be directly accessed by other MAC functions such as dicoverty, Peering, relaying, etc.
* CM can also directly interract to higher layer (e.g. applications) and/or PHY layer through inter-layer primitives
* The CM in one peer can communication to the CM in another peer through MAC layer frames over the interface “Icm” as indicated in Figure 5‑1. When two CMs talk to each other, client/server model may be used.Basically, one CM acts a Client and the other CM acts as a Server The client CM sends request to and waits for response from the server CM.
  + One CM (e.g. a CM Server) can simultaneously talk multiple other CMs (e.g. CM Clients) on other peers via multicast/broadcast.
* If higher layer, PHY layer, or MAC-layer functions can not find the required context information from the local CM (i.e. the CM on the same peer), the local CM can contact the remote CM (i.e. the CM on another peer) by sending a request for the required context information.
* The CM is able to perform context analytics or operations such as context filtering, context summarization, context aggregation, etc.



Figure 5-19-2 One-hop remote context exchange procedure

The procedures for One-hop remote context exchange procedure is illustrated in Figure 5-19-2, where,

* Step 1: CM 1 sends Context Request frame to CM 2. This message may contain the following fields/parameters:
  + List of Context Operations: to indicate the operations to be performed in CM 2. CM 1 may contain multiple operations in one Context Request frame.
  + List of Context ID: to indicate the context IDs which the operations will be operated on.
  + Response Indication: to indicate if the responses should be sent back in one MAC frame or can be in separate MAC frames.
* Step 2: CM 2 sends Context Response frame back to CM 1. Note that the response for CM 1 may be too long to be included in one MAC frame. Instread, multiple MAC fames are needed. In this message, the following fields/parameter may be included:
  + Number of Remained Responses: to indidate how many responses (i.e. Context Response frame) are left to be transmitted after the response in Step 2.
  + ACK Indication: to indicate if an ACK is required for the Context Response frame. This ACK indication can also indicate if the required ACK is for each left responses or one ACK for certain number of responses.
  + List of Operations: to indidate the operation(s) which this response is for.
  + List of Context ID: to indicate the context IDs which this response is for.
  + Context Values: to contain the values of related context.
* Step 3: CM 1 sends Context Response ACK frame to CM 2 if Step 2 requires such an acknowledgement.
* Step 4: CM 2 sends the left Context Response to CM 1, similar to Step 2.
* Step 5: CM 1 sends Context Response ACK to CM 2, similar to Step 4.
* More steps may be repeated until all responses for Step 1 are transmited from CM 2 to CM 1.

#### Multi-Hop Remote Context Exchange



Figure 5-19-3 Multi-hop remote context exchange architecture

Figure 5-19-3 illustrates proxy-based context management architecture where,

* Context Manager Client (CMC) indirectly communicates to Context Manager Server (CMS) via Context Manager Proxy (CMP).
* CMC, CMS, and CMP are MAC-layer function. The CM in one peer may act as any combination of CMC, CMP, and CMS.
* CMC issues requests to CMP and receives response from CMP
* CMP receives the request from CMC and it may perform:
  + Translate the request to the format which CMS can understand if needed, and forward the translated request to CMS.
  + Optionally, CMP may send response to CMC directly without contacting CMS.
* CMS receives translated request from CMP and sends response back to CMC.
* For example, one game player in a group of players (peers) may act as CMP
  + to manage and control context information exchanging among other players, or

to collect context information about some peers which can be shared and discovered by other peers.

 Figure 5-19-4 Multi-hop proxy-based remote context exchange procedure

Figure 5-19-4 illustrates the procedures for proxy-based context exchange, where:

* Step 1: The CMC on peer 1 sends Context Request frame to the CMP on peer 2.
* Step 2: The CMP looks up its local context database. If there are enough context information to answer the Context Request in Step 1, it sends Context Response Frame to CMC and the following steps may not be required; otherwise, it may translate the Context Request into the format which CMS can understand.
* Step 3: The CMP sends Context Request frame to CMS on the peer 3.
* Step 4: The CMS on peer 3 sends Context Response frame to the CMP.
* Step 5: The CMP sends Context Request frame to CMS on the peer 4.
* Step 6: The CMS on peer 4 sends Context Response frame to the CMP.
* Step 7: The CMP aggregates the responses received from both CMSs and sends Context Response frame to the CMC.

#### Local Context Exchange



(a)



(b)

Figure 5-19-5 Local context operations

Figure 5-19-5 illustrates the procedures for local context operations. In Figure 5-19-5 (a)

* Step 1: Higher layer, PHY layer, and/or MAC functions issues Context Request Primitive to CM 1 on the same peer. This primitive may contain the following information:
  + Primitive ID: to indicate the type of this primitive.
  + List of Context Operations: to indicate the operations to be performed in CM 2. CM 1 may contain multiple operations in one Context Request frame.
  + List of Context IDs: to indicate the context IDs which the operation will be operated on.
* Step 2: CM 1 sends Context Confirmaiton Primitive to higher layer, PHY layer, and/or MAC functions. This primitive may contain the following information:
  + Primitive ID: to indicate the type of this primitive
  + Context Values: to indicate the values of requested context.

Figure 5-19-5 (b),

* Step 1: CM 1 sends Context Indicaiton Primitive to higher layer, PHY layer, and/or MAC functions. This primitive may contain the following information:
  + Primitive ID: to indicate the type of this primitive
  + List of Context Operations: to indicate the operations to be performed in CM 2. CM 1 may contain multiple operations in one Context Request frame.
  + List of Context IDs: to indicate the context IDs which the operation will be operated on.
* Step 2: Higher layer, PHY layer, and/or MAC functions issues Context Response Primitive to CM 1 on the same peer.
  + Primitive ID: to indicate the type of this primitive
  + Context Values: to indicate the values of requested context.

#### Session-based Context Operations



Figure 5-19-6 Session-based context operations

Figure 5-19-6 illustrates the procedures for Session-based Context Operations, where:

* Step 1: CM 1 sends Context Begin frame the CM 2 to request the beginning of a context exchange session.
* Step 2: CM 2 sends Context Begin Approval frame to CM 1 to approve CM 1’s request for the context exchange session.
* Step 3: CM 1 sends Contex Request frame to CM 2.
* Step 4: CM 2 sends Context Response frame to CM 1.
  + The following Step 5 and Step 6 just repeat Step 3 and Step 4.
* Step 5: CM 1 sends Contex Request frame to CM 2.
* Step 6: CM 2 sends Context Response frame to CM 1.
* Step 7: CM 2 sends Contex Request frame to CM 1.
  + Note that CM 2 can piggyback Context Request in Step 6 when it sends Context Response to CM 1.
  + Optionally, CM 2 can also send Context Request before CM 1 finishes its turn (i.e. request context information from CM 2).
* Step 8: CM 1 sends Context Response frame to CM 2.
* Step 9: CM 1 sends Context End frame to CM 2 to request to stop the current context exchange session.
* Step 10: CM 2 sends Context End ACK frame to CM 1 as an acknowledgement.

### Cross-Layer Context Management

***Note*** *that the contribution of cross layer context management is in final contribution document # 15-14-0264-00-0008.*

### Cross-Layer Data Acknowledgement

**380r2(beginning)**

ACK-based retransmission mechanisms exist in most MAC protocols such as IEEE 802.15 and IEEE 802.11 series to provide reliable transmission in MAC layer. In the meantime, higher layer protocols (e.g. TCP) also provide ACK-based reliable transmission based. However, the MAC-layer re-transmission and higher-layer retransmission have been treated independent of each other, which introduce extra overhead and latency. For PAC applications, it turns out that the independently-treated MAC-layer retransmission and higher-layer retransmission are inefficient and could be redundant.

Cross-layer data acknowledgement aims to reduce the number of messages by allowing “application data” to be piggybacked in the MAC-layer ACK; here, “application data” could be an application ACK. The problem is how to coordinate and optimize MAC-layer (re-)transmission and higher-layer (re-)transmission mechanisms for PAC applications.

#### Basic Cross-Layer Data Acknowledgement

**Figure 5-19-9** illustrates the scenario where the sender and receiver are within one-hop. In addition, application response will be piggybacked in application ACK; in other words, the application ACK contains both ACK information and application-related response.

**Figure 5-19-9 (a)** depicts existing case without using cross-layer ACK, where two MAC data frames contain application data (i.e. App Data) and application ACK (i.e. App ACK) respectively. As a result, two MAC ACK frames are required for those two MAC data frames. In total, there are four messages between the sender and receiver. Note that the application data could be an application request and application ACK could be an application response.

The New MAC ACK in  **Figure 5-19-7 (b)** (i.e. Integrated MAC ACK) contains not only the normal MAC ACK but also the application ACK. In other words, the Integrated MAC ACK serves two purposes simultaneously: 1) To acknowledge the previous MAC data frame from the sender to the receiver; 2) To acknowledge the application data contained in the previous MAC data frame. With cross-layer ACK, the number of total messages is greatly reduced with the following benefits:

* It reduces the overall power consumption and make the system more energy-efficient.
* The reduction of required messages also reduces the number of bits to be transmitted over the air since each message needs MAC header, MAC footer, PHY header, and PHY footer.
* The reduction of required messages can mitigate and reduce potential channel collision over the air and in turn improve performance in terms of latency, throughput, and energy-consumption, etc.



(a). Without Cross-Layer ACK



(b). With Cross-Layer ACK

Figure 5-19-7. Cross-Layer ACK for Piggybacked Application Response

**Figure 5-19-8** illustrates the flow chart of the Sender for “Cross-Layer ACK” operation:

* When receiving an application message from higher layer, if the message needs ACK and is not fragmented, “Cross-Layer ACK” should be enabled with following new operations to perform (see **Figure 21** (a)):
  + Disable MAC fragmentation and mark a flag in the MAC frame to request “integrated ACK from the receiver”, or enable MAC fragmentation but only mark the flag for “integrated ACK” for the last fragment.
  + Maintain mapping relationship between MAC frame and application message.
* When receiving an “Integrated MAC ACK” from PHY layer (i.e. from the receiver), “Cross-Layer ACK” should be enabled with following new operations to perform (see  **Figure 5-19-8** (b)):
  + Map “Integrated ACK” to MAC data frame.
  + Map “Integrated ACK” to application data.
  + Re-construct normal App ACK.
  + Forward the re-constructed App ACK to application layer.
* If the Sender does not receive an expected Integrated MAC ACK or a normal MAC ACK from the Receiver for a previously sent MAC frame, it shall re-transmit the MAC frame when the retransmission time gets expired.

**Figure 5-19-9** illustrates the flow chart of the Receiver for “Cross-Layer ACK” operation:

* When receiving an MAC data frame from the PHY layer, if “Integrated ACK” flag is set, “Cross-Layer ACK” shall be enabled with following new operations to perform:
  + Hold MAC-layer ACK
    - If the receiver finds that it needs to wait for long time to get application ACK, the receiver may disable “Cross-Layer ACK” and just use normal MAC ACK procedures.
  + Pass application data to high layer and get application ACK/response back from high layer.
  + Generate “Integrated MAC ACK” MAC frame: The Integrated ACK frame should contain the following new information/fields/parameters:
    - Integrated ACK Flag: to indicate this ACK is an Integrated ACK,
    - Application ACK Information Element (IE): to contain the corresponding application ACK. Optionally, this IE can contain and carry application data especially when the size of the application data is small.
* Optionally, the receiver can independently determine to use Normal ACK or Cross-Layer ACK.

**Figure 5-19-10** shows an example of Cross-Layer ACK operations including both the Sender and the Receiver.

Overall, the following changes (See Table 3) should be introduced to MAC data frame and MAC ACK frame to support “Cross-Layer ACK”

* Changes to MAC Data Frame:
  + Add one “Cross-Layer ACK” bit in MAC data frame header to indicate that this MAC data frame expects or requests an “Integrated MAC ACK”.
* Changes to MAC ACK Frame:
  + Add one “Cross-Layer ACK” bit in MAC ACK frame header to indicate that this ACK is an Integrated MAC ACK frame,
  + Embed or contain “App ACK” (or even App Data) in MAC ACK frame to formulate Integrated ACK. A new MAC-layer IE (i.e. App IE) is introduced to contain the App ACK (or even App Data).
  + Note that if there is no segmentation, the application ACK may not be needed via the proposed cross-layer ACK mechanism.

Table 3. New Fields/Parameters to MAC Data Frame and MAC ACK Frame

|  |  |
| --- | --- |
| **New Field of MAC Data Frame** | **New Field of MAC ACK Frame** |
| **Cross-Layer ACK Flag**: to indicate if this MAC data frame expects or requests an “Integrated MAC ACK” or not. | **Cross-Layer ACK Flag**: to indicate if this ACK is an Integrated MAC ACK frame or not.  **App IE**: to contain the App ACK or App Data. |



(a). When Receiving Messages from Higher Layer



(b). When Receiving MAC ACK Frames from PHY Layer

Figure 5-19-8. Flow Chart of Cross-Layer ACK at the Sender



Figure 5-19-9. Flow Chart of Cross-Layer ACK at the Receiver



Figure 5-19-10. An Example of Cross-Layer ACK Operations

#### Streamlined Cross-Layer Data Acknowledgement

In **Figure 5-19-7 (b)**, it may take the Receiver certain time to calculate App ACK and in turn the Integrated MAC ACK cannot be issued until the App ACK becomes available. Thus, the App ACK is not necessarily paired with the MAC ACK that is associated with the MAC frame that carried Application Data. This constraint can be alleviated by using the “**Streamlined Cross-Layer ACK**” approach as illustrated in **Figure 5-19-11**, where:

* Step 1: The Sender sends MAC Data Frame 1 to the Receiver.
* Step 2: The Receiver sends normal MAC ACK 1 back to the Sender to acknowledge the receiving of MAC Data Frame 1.
* Step 3: The Sender sends MAC Data Frame 2 to the Receiver.
* Step 4: The Receiver sends Integrated MAC ACK 1 to the Sender. This Integrated MAC ACK also contains an App ACK for App Data 1 received from Step 1. As a result, this integrated ACK acknowledges the receiving of both MAC Data Frame 2 and App Data 1.
  + Note that the Receiver may already calculate the App ACK for App Data 1 and in turn it can issue this Integrated ACK immediately after receiving MAC Data Frame 2 in Step 3.
* Step 5: The Sender sends MAC Data Frame 3 to the Receiver.
* Step 6: The Receiver sends Integrated MAC ACK 2 to the Sender. This Integrated MAC ACK also contains an App ACK for App Data 2 received from Step 3. As a result, this integrated ACK acknowledges the receiving of both MAC Data Frame 3 and App Data 2.
  + Note that the Receiver may already calculate the App ACK for App Data 2 and in turn it can issue this Integrated ACK immediately after receiving MAC Data Frame 3 in Step 5.
* Step 7: The Sender sends MAC Data Frame 4 to the Receiver.
* Step 8: The Receiver sends Integrated MAC ACK 3 to the Sender. This Integrated MAC ACK also contains an App ACK for App Data 3 received from Step 5. As a result, this integrated ACK acknowledges the receiving of both MAC Data Frame 4 and App Data 3.
  + Note that the Receiver may already calculate the App ACK for App Data 3 and in turn it can issue this Integrated ACK immediately after receiving MAC Data Frame 4 in Step 7.
* Notes:
  + In **Figure 5-19-13**, optionally, the Receiver can hold multiple App ACKs and piggyback them together in one immediate MAC ACK.
  + Step 7&8 can be repeated if there are more App Data and MAC Data Frames to be transmitted from the Sender to the Receiver.



Figure 5-19-11. Streamlined Cross-Layer ACK for Piggybacked Application Response

**380r2(end)**