Embedded Linux Conference, Fall 2022

# Improving IEEE 802.15.4 MAC management support in the Linux kernel

Miquel Raynal
*miquel.raynal@bootlin.com*

Corrections, suggestions, contributions and translations are welcome!

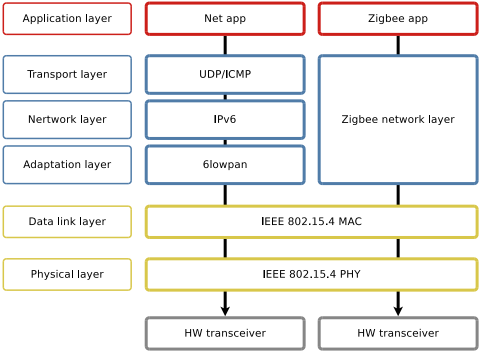
# Miquel Raynal

- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Open-source contributor
  - Maintainer of the NAND subsystem
  - Co-maintainer of the MTD subsystem
  - Kernel support for various ARM SoCs
  - **Active contributor to the WPAN subsystem** with Qorvo support
- Living in **Toulouse**, France

# A walk through the IEEE 802.15.4 specification

# Functional description

- Defines the PHY layer and the MAC sublayer
- Introduced to build Wireless Personal Area Networks (WPAN)
- Low power, low range (10m), low rate (up to 250kib/s)
- Easy connection between sensors and actuators
  - home automation
  - infrastructure monitoring
  - medical body area
  - RFID tags tracking
  - ...
- A base for Zigbee and 6LowPan



Application layer

Net app

Zigbee app

Transport layer

UDP/ICMP

Nertwork layer

IPv6

Zigbee network layer

Adaptation layer

6lowpan

Data link layer

IEEE 802.15.4 MAC

Physical layer

IEEE 802.15.4 PHY

HW transceiver

HW transceiver

IEEE 802.15.4 stack integration in the OSI model

# The PHY layer

The PHY layer manages:

- Channel switches across multiple frequency bands
- Energy Detection (ED)
- Medium Access
  CSMA-CA, TSCH-CCA, LECIM ALOHA...
- Transmitting/receiving packets
- Link Quality Indicators (LQI)
- Physical data encoding
  O-QPSK, BPSK, GFSK,...
- Ranging (UWB PHYs only)

- Encapsulating payloads into PHY Protocol Data Units (PPDU)

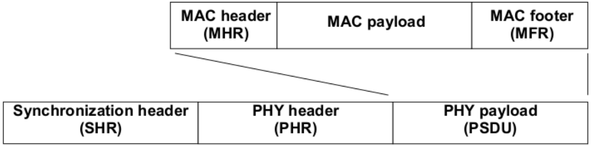


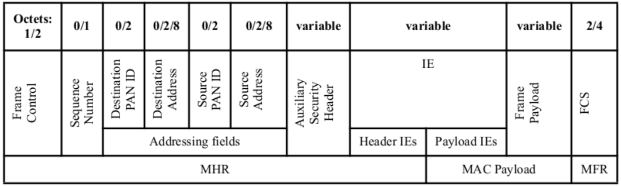
**Figure 5-7—Schematic view of the PPDU**

IEEE 802.15.4 specification screenshot

# The MAC sublayer

The MAC sublayer offers:

- MAC data services
  - Encapsulating payloads into MAC Protocol Data Units (MPDU)



<table>
<tr><td>Octets: 1/2</td><td>0/1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td colspan="2">variable</td><td>variable</td><td>2/4</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN ID</td><td>Destination Address</td><td>Source PAN ID</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td colspan="2">IE</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td><td>Header IEs</td><td>Payload IEs</td></tr>
<tr><td colspan="8">MHR</td><td colspan="2">MAC Payload</td><td>MFR</td></tr>
</table>

**Figure 7-1—General MAC frame format**

IEEE 802.15.4 specification screenshot

- MAC management services through the use of the MAC subLayer Management Entity (MLME):
  - Channel choice
  - Frames validation
  - Beacon and scan management
  - Associations/dis-associations
  - Security mechanisms
  - Acknowledgments

# RFD vs. FFD

- Reduced Function Device (RFD)
  - Limited devices, usually just able to send a few bytes of data and return to sleep
  - Typically battery powered
  - Possibly only working in Rx or only in Tx
  - Leaf nodes in a network
- Full Function Device (FFD)
  - Expected to support much more features
  - Typically mains powered
  - Either coordinators or leaf nodes
    - Coordinators provide synchronization services

# Personal Area Networks (PAN)

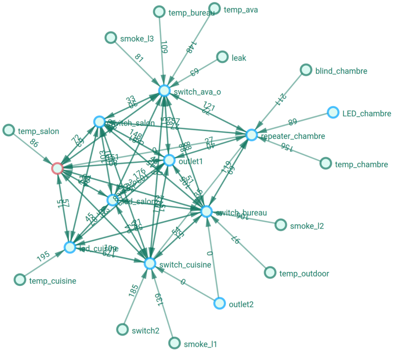
Several networks can live together on the same channel thanks to PAN IDs.

- If a coordinator does not detect any surrounding PAN or decides to create its own, it may start a PAN and act as PAN coordinator:
  - Picking a PAN ID (16-bit), apparently unused
  - Picking a short address (16-bit)
- Maintaining the PAN
  - Advertising the PAN to the surrounding devices
  - Allowing devices to associate
    - Possibly allocating short addresses
    - Can handover the "PAN coordinator" role

# Discovery

- Coordinators and PAN coordinators shall advertise their PAN by sending beacons
  - Either upon reception of a `BEACON REQUEST`
  - Or “passively” at a given rate, in beacon-enabled PANs
- Beacons are short frames with information about the emitting device and its PAN

- Devices can scan the various channels they support
  - Energy Detection (ED) scans to know which channels are used
  - Passive scans to detect surrounding beacon enabled PANs with their LQI
  - Active scans to detect surrounding beacon and non-beacon enabled PANs with their LQI



Alexandre Belloni’s IEEE 802.15.4 based home network

# Beacon enabled PANs

Beacons enabled PANs are more energy efficient, because they allow battery powered devices to synchronize:

- Beacon are sent at a fixed rate, the beacon interval, based on the beacon order
- A beacon transmission starts a superframe
  - Superframe duration depend on the superframe order, advertised in the beacon
  - The superframe is the active portion between each beacon
  - The remaining part is the inactive portion, when radios can be turned off

Superframes are divided into time sections, themselves divided into timeslots

- The Contention Aware Period (CAP): devices compete for the medium access
- The Contention Free Period (CFP): Guaranteed Time Slots (GTS) for critical and low-latency devices

ACKs are unslotted (answered immediately)

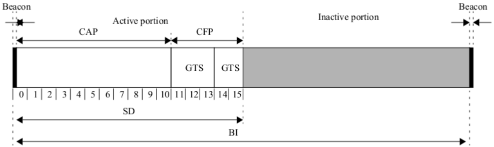


**Figure 6-1—An example of the superframe structure**

IEEE 802.15.4 specification screenshot

# Hardware filtering

- A device can be addressed either with:
  - its extended address (8 bytes)
  - its short address (2 bytes) if part of a PAN
  - the broadcast address
- Most transceivers are capable of different hardware filtering levels defined by the specification:
  - no filtering
  - promiscuous mode (checks the frame integrity only)
  - address filters (checks the frame validity)
- These address filters must be kept in sync with the device association state

Caution: the linux-wpan community used the word “promiscuous” differently: to define the total absence of filtering, with sniffing interfaces in mind

# MLME operations

- MAC management commands
  - Discovering surrounding devices
  - Enlarging/shrinking the network
  - Keeping all devices synchronized
  - Handling faulty situations (loss of contact, conflicts, etc)

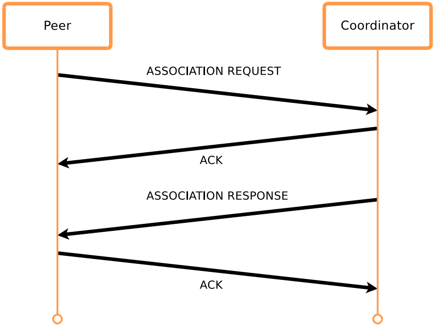
# Scanning

- Any device may enter scan mode upon MLME request
  - Transmissions shall be paused
  - Use of a specific filter mode
    - Only beacon frames are accepted
    - Beacons have no destination field
  - The MLME request specifies the channels to scan
  - Scanning a channel involves waiting for a known period of time
- Beacon frames sent by other coordinators with their information must be parsed and forwarded to the upper layers
  - Devices build this way a list of the surrounding coordinators
  - Parameters like the LQI during beacon reception may be used to pick the right coordinator to associate with

# Associations

In order to create networks, devices shall associate with each other

- Associations can be attempted by RFD and FFD devices



- `ASSOCIATION RESPONSE` payload:
  - Status: Success, PAN coordinator at capacity, PAN coordinator rejected the request
  - Short address (0xFFFE if not requested)

# Dis-associations

- Usable by both the parent and the child
  - The coordinator can inform a child that it has been kicked out

  - A child may inform the coordinator that it leaves

- In both cases, address filters must be updated

# PAN ID conflicts

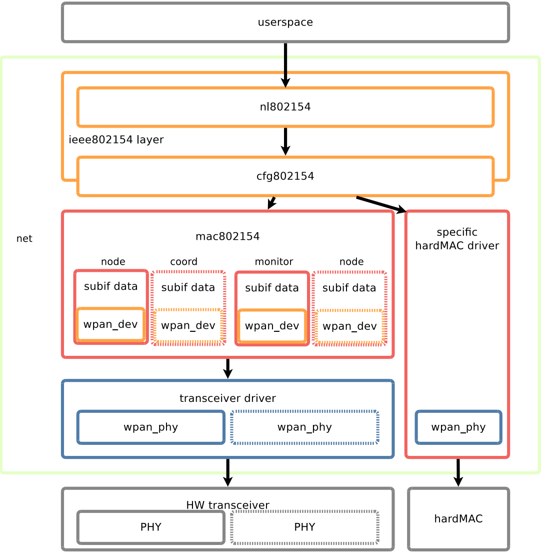
- Two situations may lead to a PAN ID conflict
  - The PAN coordinator receives a beacon from another coordinator
  - The other coordinator says it is the PAN coordinator
- or
  - Any node in the network receiving beacons with the PAN coordinator bit set from a node that is not its known PAN coordinator
  - A `CONFLICT NOTIFICATION` command must be sent
- Upon detection of a conflict, the concerned PAN controller should resolve the situation by:
  - Scanning for an available channel/PAN ID
  - Send a `COORDINATOR REALIGNMENT` command

# Internal realignments

- Devices receiving a `COORDINATOR REALIGNMENT` command shall:
  - Change their internal parameters (PAN Information Base, PIB)
  - Follow the channel/PAN ID change
  - Except TSCH devices
    - Channel hoping is part of their common actions
    - They can safely ignore the realignment commands
- Devices loosing the sync with their coordinator (no more `ACK`s?) shall
  - Iterate over all their supported channels
  - Generate `ORPHAN NOTIFICATIONS`
  - Expect a `COORDINATOR REALIGNMENT` from the coordinator which recognized the orphaned device in return
  - Otherwise look for a new coordinator

# The Linux kernel IEEE 802.15.4 stack

# Architecture



# Hardware offloading

Most supported devices in Linux are bare transceivers driven by the softMAC layer (single exception). These transceivers may usually perform some of the MAC operations by hardware:

- Frame validation
  - Depend on the configuration of the address filters
    - short address, extended address, PAN ID
  - A promiscuous mode is usually available to bypass the filters upon request
  - Other filtering levels are described by the specification
- Acknowledgment
  - Time critical operation
  - Done only if:
    - The frame passes the filters
    - The frame has the `Acknowledgment Request` (AR) bit enabled
    - Not disabled by the user

# Current state of the support

- Defining networks and addresses can only be done statically:
  - PAN ID and short address can be manually set
  - No discovery/no dynamic network management
  - The administrator shall provide a static list of devices and their addresses
  - The PAN ID shall be picked-up in advance
- But...
  - Devices can move, appear, disappear
  - Static descriptions are no longer relevant
- Need for a dynamic way to discover the peers around and possibly associate with them dynamically as well
  - There are MAC commands for that!

# The scanning interface 1/2

Trigger: netlink user request with:

- The type of scan: passive? active? ED?
- The range of channels to scan, possibly the page
- The duration (Beacon Interval, BI)

The request will be forwarded to the MAC layer:

- The Tx traffic on the interface is stopped
- The Tx queue gets flushed
- The hardware filters must be configured for the scan
  - Any non-beacon frame gets dropped
- A background thread is started

# The scanning interface 2/2

- The background thread shall:
  - Iterate over all the requested channels
  - Send `BEACON REQUEST`s during active scans (only)
  - Wait for incoming beacons
- Upon reception, the Rx logic will:
  - Check the beacon validity
  - Forward its content to the upper layers
- Scans can be aborted at any moment
- An end of scan information is sent back to the user
- The interface is set back in its original state

# The beaconing interface

Trigger: netlink user request with:

- The duration (Beacon Interval)
  - The maximum duration is 15
  - 15 means beacons are only sent upon `BEACON REQUEST`s
  - Otherwise the sending rate is not impacted by `BEACON REQUEST`s
- A delayed background job is started
- Can be modified or stopped at any moment

# iwpan -h

```
iwpan dev <devname> scan type <type> [page <page>]
                                     [channels <bitfield>]
                                     [duration <duration-order>]
iwpan dev <devname> scan trigger type <type> [page <page>]
                                             [channels <bitfield>]
                                             [duration <duration-order>]
iwpan dev <devname> scan abort

iwpan dev <devname> beacons send [interval <interval-order>]
iwpan dev <devname> beacons stop

iwpan monitor [-t|-r] [-f]
```

# Associating

- The list of devices in our range is now known
  - Their LQI as well
- Upper layers may device to associate with an available coordinator
  - User to provide a PAN ID and coordinator address to connect to
- The request is translated to the MAC layer which will:
  - Change the address filters to accept the new PAN ID
  - Send an `ASSOCIATION REQUEST` and wait for it to be `ACK`ed
- The peer coordinator shall answer with an `ASSOCIATION RESPONSE`
- Upon reception the frame is parsed:
  - In the payload, a status indicates whether the association is successful
  - If yes and if requested, a short address to use is also provided
- Address filters shall again be updated to match the new short address
- The parent device is saved for future reference

# Processing associations requests

- Associations can be refused for two reasons:
  - The maximum number of devices is reached
    - Return a `PAN AT CAPACITY` status
    - Configurable with a netlink command
  - We do not want this device in our network
    - Not highly time critical, the question can be asked to userspace
    - API not implemented yet, currently we allow all associations
  - A list of associated devices must be maintained

# Disassociation notifications

- Both ends can notify a disassociation
- The user must provide the peer address (short or long)
- An `ACK` is expected from the remote device
  - Choice in the code: assume the device disassociated anyway

# iwpan -h

```
iwpan dev <devname> associate pan_id <pan_id> coord <coord>
iwpan dev <devname> disassociate short_addr|ext_addr <addr>
iwpan dev <devname> list_associations
iwpan dev <devname> set max_associations <max_associations>
```

# Upstream proposals, discussions ongoing

- Kernel patches:
  - v2 https://lore.kernel.org/all/20220826144049.256134-1-miquel.raynal@bootlin.com/
  - v3/only filtering https://lore.kernel.org/all/20220905203412.1322947-1-miquel.raynal@bootlin.com/
  - Latest version https://github.com/miquelraynal/linux/tree/wpan-next/scan
- wpan-tools patches:
  - Last patches https://lore.kernel.org/all/20220701143434.1267864-1-miquel.raynal@bootlin.com/
  - Latest version https://github.com/miquelraynal/wpan-tools/tree/wpan-master/scan
- Zephyr changes https://github.com/zephyrproject-rtos/zephyr/pull/49947

No support for orphan notifications/coordinator realignments yet

# Live wild: demo time!

Hardware setup:

- One ATUSB device acting like a PAN coordinator (wpan0/coord0)
- One ATUSB device acting like a node (wpan1/coord1)
- One ATUSB device monitoring (wpan2/mon2)
- One Arduino Nano 33 BLE running Zephyr being a leaf node

# Questions? Suggestions? Comments?

Miquel Raynal

*miquel.raynal@bootlin.com*