

Synopsis: focus on MAC / PHY rework, including security aspects (if time: security also beyond lower layers)

Moderator: José. Location: Room B in Gather.town

see slides from José: [https://github.com/jia200x/docs/blob/master/bs\\_ll.pdf](https://github.com/jia200x/docs/blob/master/bs_ll.pdf)

Status:

RDM: The 802.15.4 Radio HAL #13943  
<https://github.com/RIOT-OS/RIOT/pull/13943>

ieee802154\_submac: add initial support for common MAC sub layer  
<https://github.com/RIOT-OS/RIOT/pull/14950>

Discussion:

- \* function pointer vs. switch case:
  - > Security concerns about function pointers
  - > Focus on the API instead of optimizations
- \* Hannes suggest to compare network device driver APIs among different network stack implementations and try to harmonize
  - > The Radio HAL design was revised against the following Radio APIs:
    - OpenWSN
    - OpenThread
    - Linux (ieee802154\_ops)
    - Contiki (radio\_driver)
    - Mbed (device\_driver\_s)
    - Zephyr-OS
  - > Should we follow this discussion on Github? Mailing list?
- \* Status of other Link Layers:
  - > BLE probably doesn't require such a rework because most internal stuff are handled by the stack (Nimble)
- \* IEEE 802.15.4 MAC
  - > Options: implement custom IEEE 802.15.4 MAC (with L2 security, indirect transmission, etc).
  - > There seems to be consensus about focusing on existing implementations of IEEE 802.15.4 (OpenWSN, OpenThread)
- \* How to send L2 data?
  - > It might be interesting to have a mechanism to abstract sending L2 data.
    - > E.g it's not nice to build an ethernet or IEEE 802.15.4 frame each time sometimes wants to send
- \* Unify network stack integration code (IRQ handling, init code)
  - > The OS shouldn't hardcode the desired mechanism for handling IRQ (e.g ``event_t``, ``msg_t``)
  - > However, these mechanisms could be unified and reused by different network stacks
    - > E.g processing IRQ could be implemented once (one for ``msg_t``, one for ``event_t``, process from ISR) and then configure the network stack to use one mechanism
- \* GNRC: Use only one stack for all network interfaces?
  - > Rough consensus for NO. GNRC was designed for being flexible. Probably memory consumption is not the focus here.
- \* Frame-buffers and zero-copy
  - > We ran out of time. Will open an issue and/or post something in the mailing list.