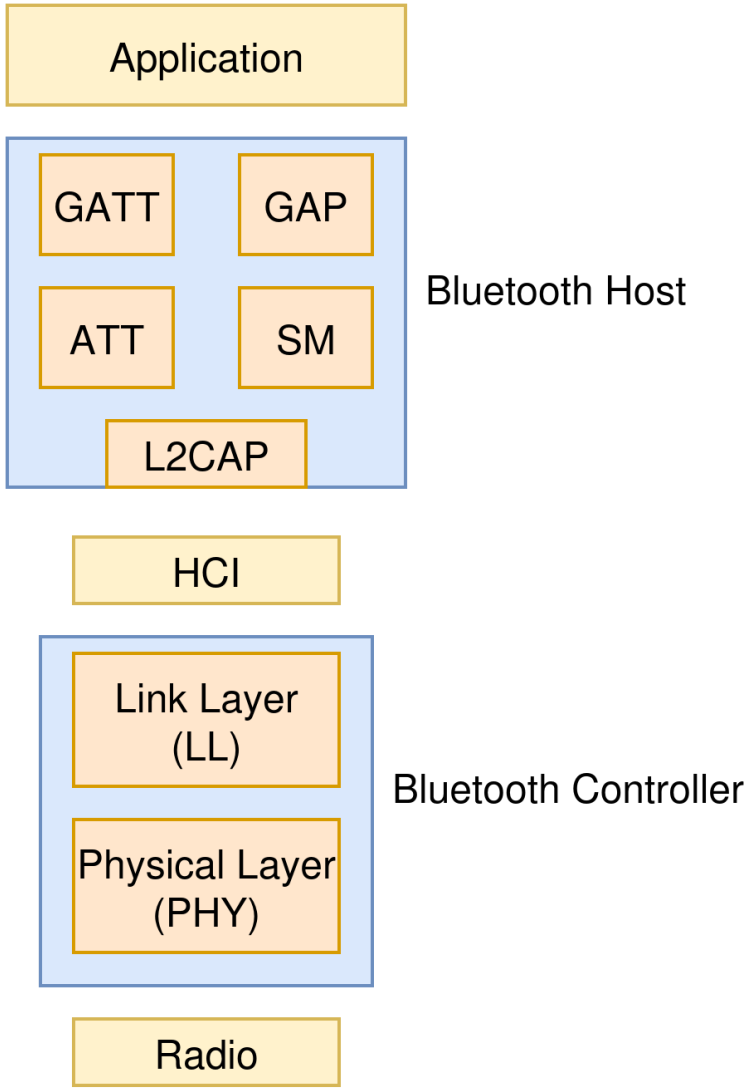
A Deep Dive into the Zephyr Bluetooth LE Controller

And how it all works underneath

Overview

- Bluetooth stack
- Bluetooth LE Controller
  - What is ULL/LLL
- Path Loss Monitoring Example
  - Host to Controller
  - Controller to Host, Execution contexts
- Power Control Example (LLCP)



Fast introduction to the Bluetooth LE stack

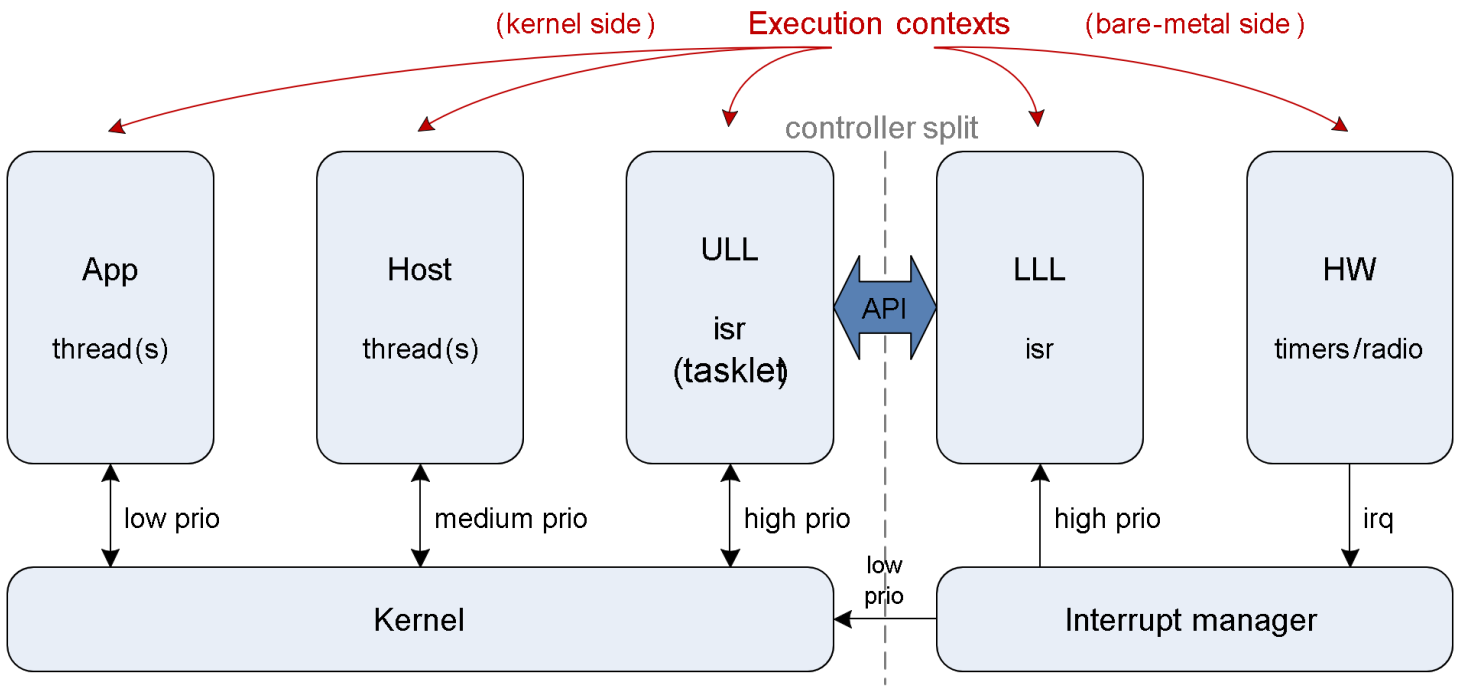
- Application
  - Ties together specific business logic
- Bluetooth Host
  - Bridge between app/controller
- Host Command Interface (HCI)
  - Commands
- Bluetooth Controller
  - Link Layer and Physical Layer
- Bluetooth Classic
  - Not pictured here

Zephyr Bluetooth Controller Features

Basic LE (Advertising, Scanning, Initiating, Connecting, Connection Events)	Connection Parameter Request Procedure	Isochronous Adaption Layer (ISOAL)	Extended Reject Indication	Feature exchange	Periodic Advertising Sync Transfer	LE Ping
Sleep Clock Accuracy Handling	LL Channel Map Update Procedure	LL Encryption Pause and Resume	LE Connection Role Switching	Support for multiple simultaneous connections	L2CAP Signaling over LE links	LE Privacy
LE Secure Connections	LE Link Layer PDU Length updates	LE 2M PHY	LE Coded PHY (S=2, S=8)	LE Extended Advertising	LE Periodic Advertising	LE Channel Selection Algorithm #2
LE PHY Update Procedure	LE Advertising Sets	LE Secure Connections	LE Isochronous Channels (CIS/BIS)	Encrypted Advertising Data	Advertising Coding Selection	LE Channel Classification by Peripheral

Upper Link Layer / Lower Link Layer – What is it?

- Split between high-level scheduling
- (Generally) ULL handles threading, LLL handles hard real-time
- (Generally) ULL is hardware agnostic while LLL allows handling proprietary, radio-hardware specifics



Source: https://docs.zephyrproject.org/latest/connectivity/bluetooth/bluetooth-ctlr-arch.html

ULL / LLL – Why?

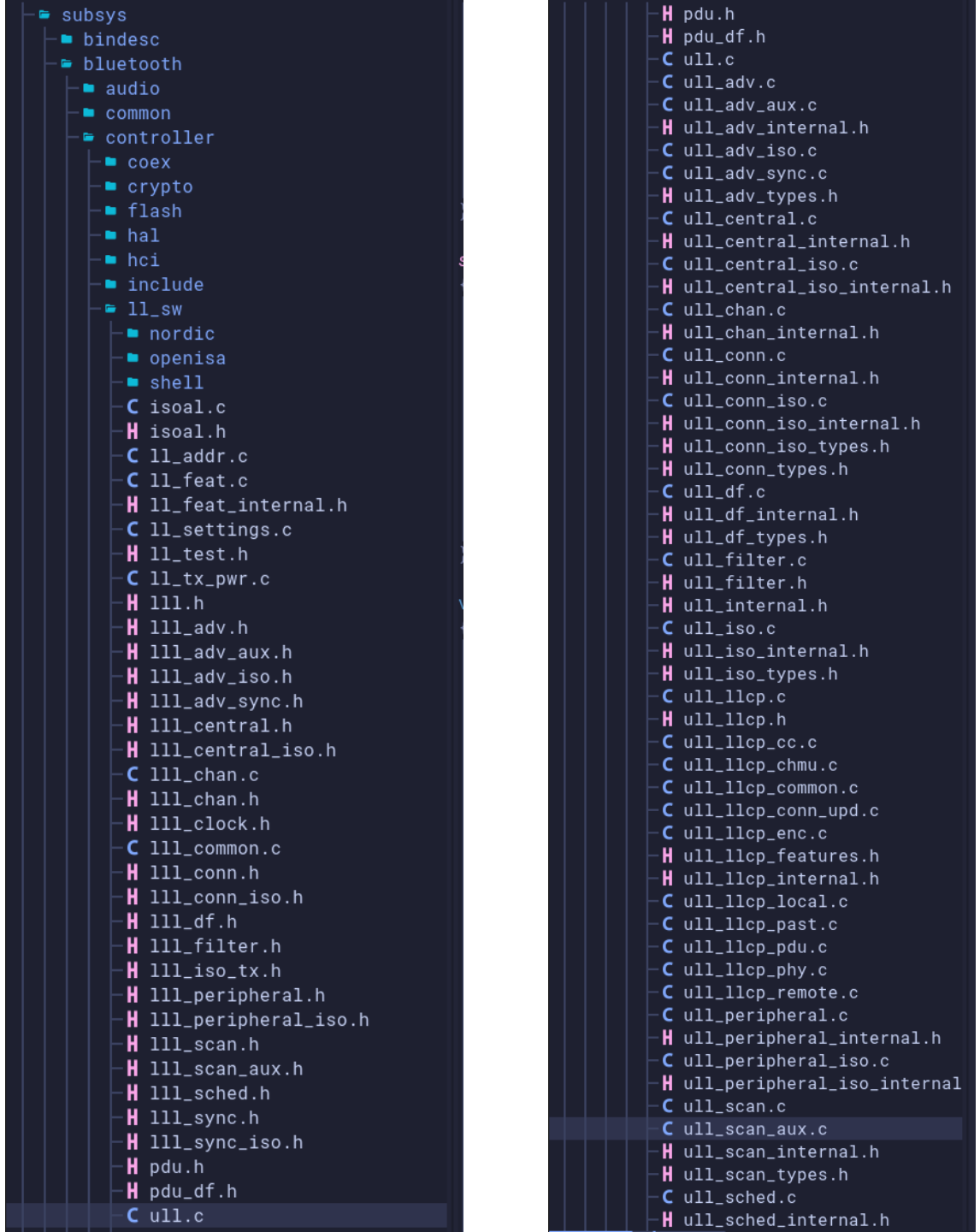
Shared logic in the ULL for different radios.

Allows for open-source contributions in the ULL

Controller/radio specific and proprietary areas in the LLL

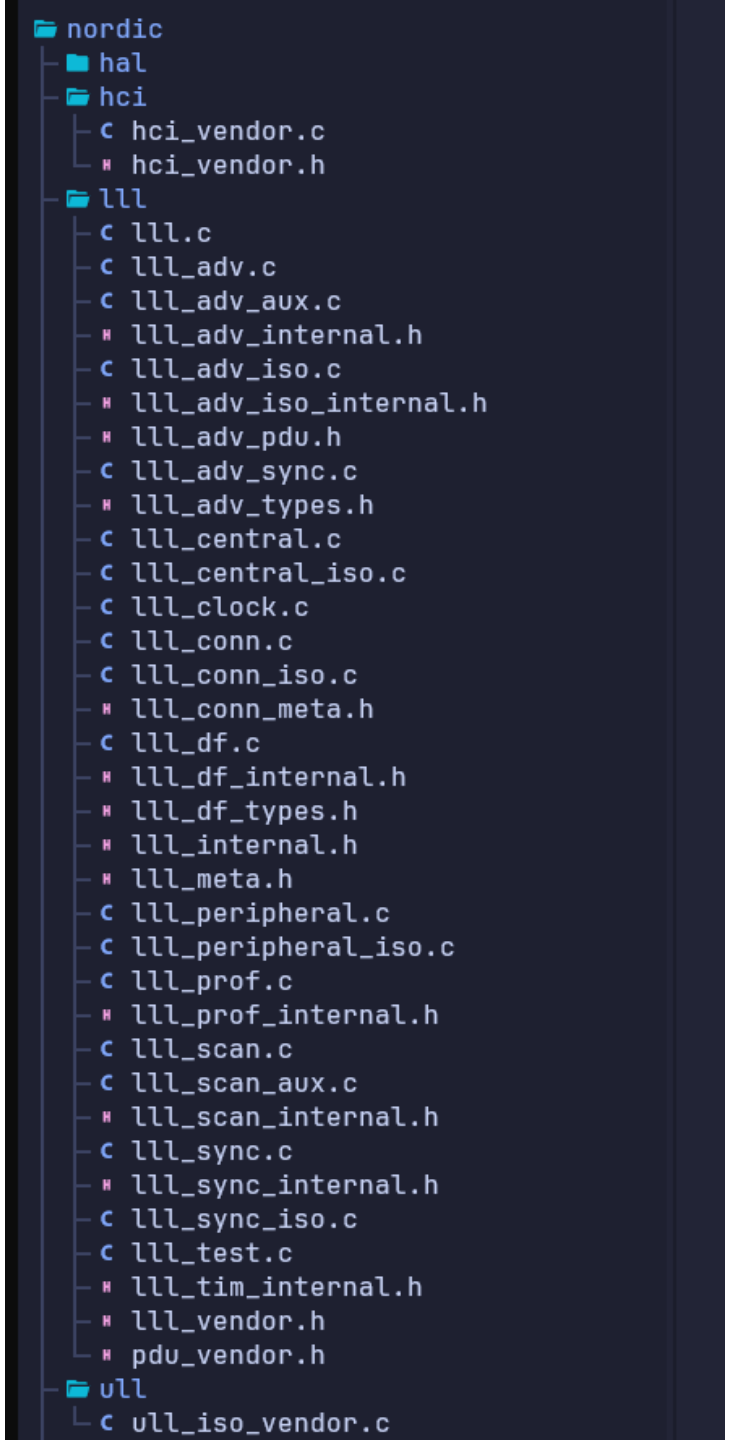
Key files, functions

- ll_*, ull_*, lll_*
  - Link Layer, Upper Link Layer, Lower Link Layer
- ULL contains both the implementation and the interface (*.h and *.c)
- LLL is interface only (*.h),
  - The implementation is vendor-specific, using HAL and radio.

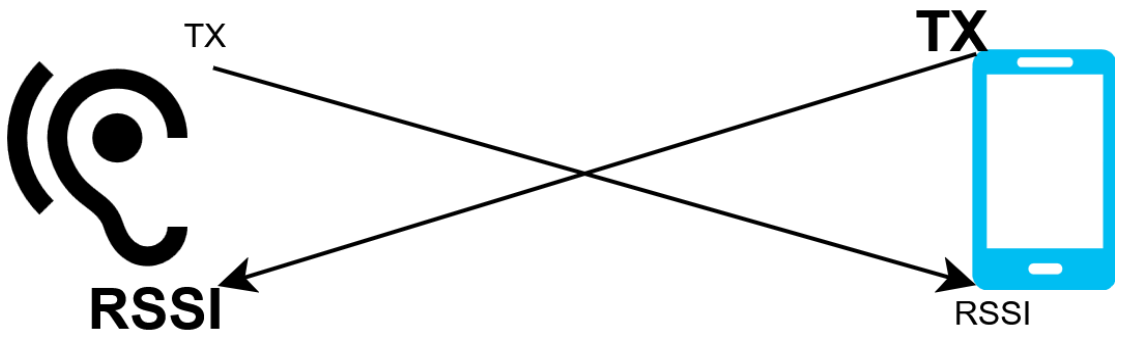


Key files, functions

- Radio implementation exists today
- Nordic has contributed a radio stack and an LLL implementation
- Their split is to share the radio with several chips, but **you can design the LLL however you want**.



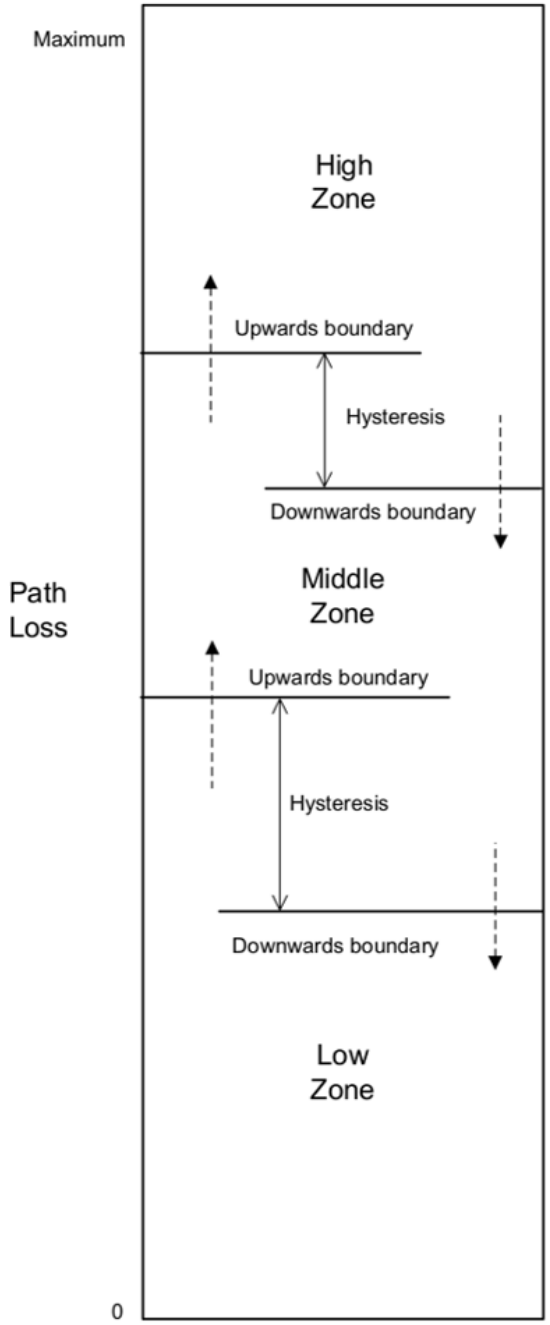
Example – Path Loss Monitoring



7.8.119 LE Set Path Loss Reporting Parameters command

Command	OCF	Command Parameters	Return Parameters
HCI_LE_Set_Path_-Loss_Reporting_-Parameters	0x0078	Connection_Handle, High_Threshold, High_Hysteresis, Low_Threshold, Low_Hysteresis, Min_Time_Spent	Status, Connection_Handle

Source: Bluetooth Core Specification v5.4



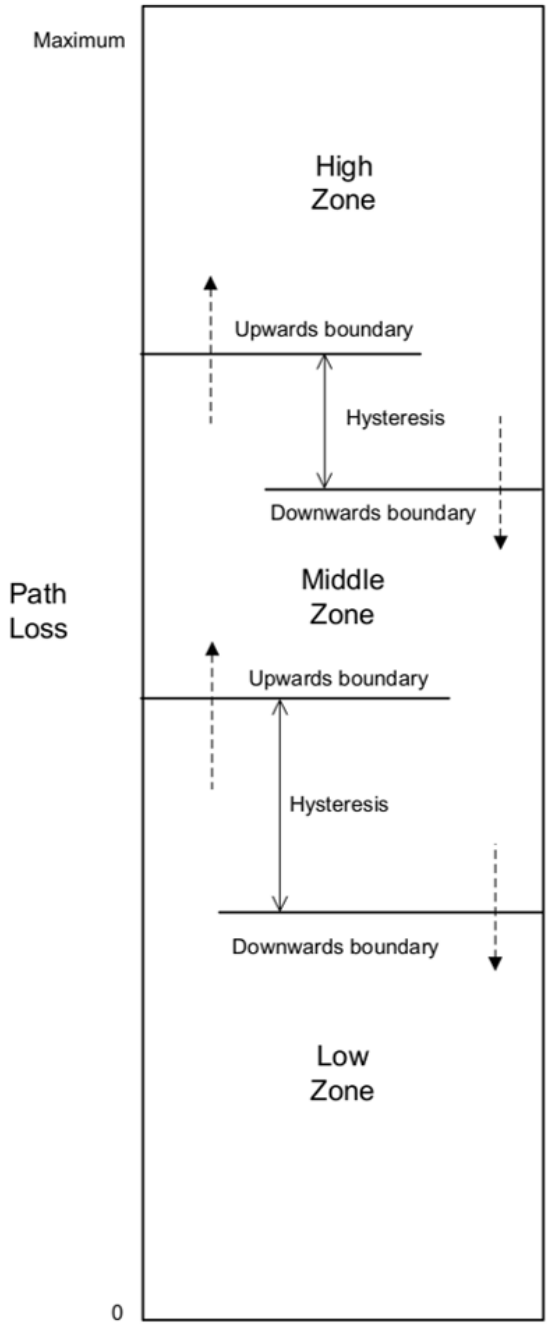
Calculating Path Loss Factor

- Terms
  - PL = Path Loss Factor
  - TX Power = Transmission power on the **peer** device
  - RSSI = Received Signal Strength Indication (measured on the **local** device)
- PL = TX Power – RSSI
- Assume +10 dBm if power control is not supported.

Path Loss Factor	TX Power	RSSI	Description
40	10 dBm	-30 dBm	As good of a signal as possible; unrealistic
70	10 dBm	-60 dBm	Great signal
80	10 dBm	-70 dBm	Okay signal
90	10 dBm	-90 dBm	Not good signal

Calculating Path Loss Factor

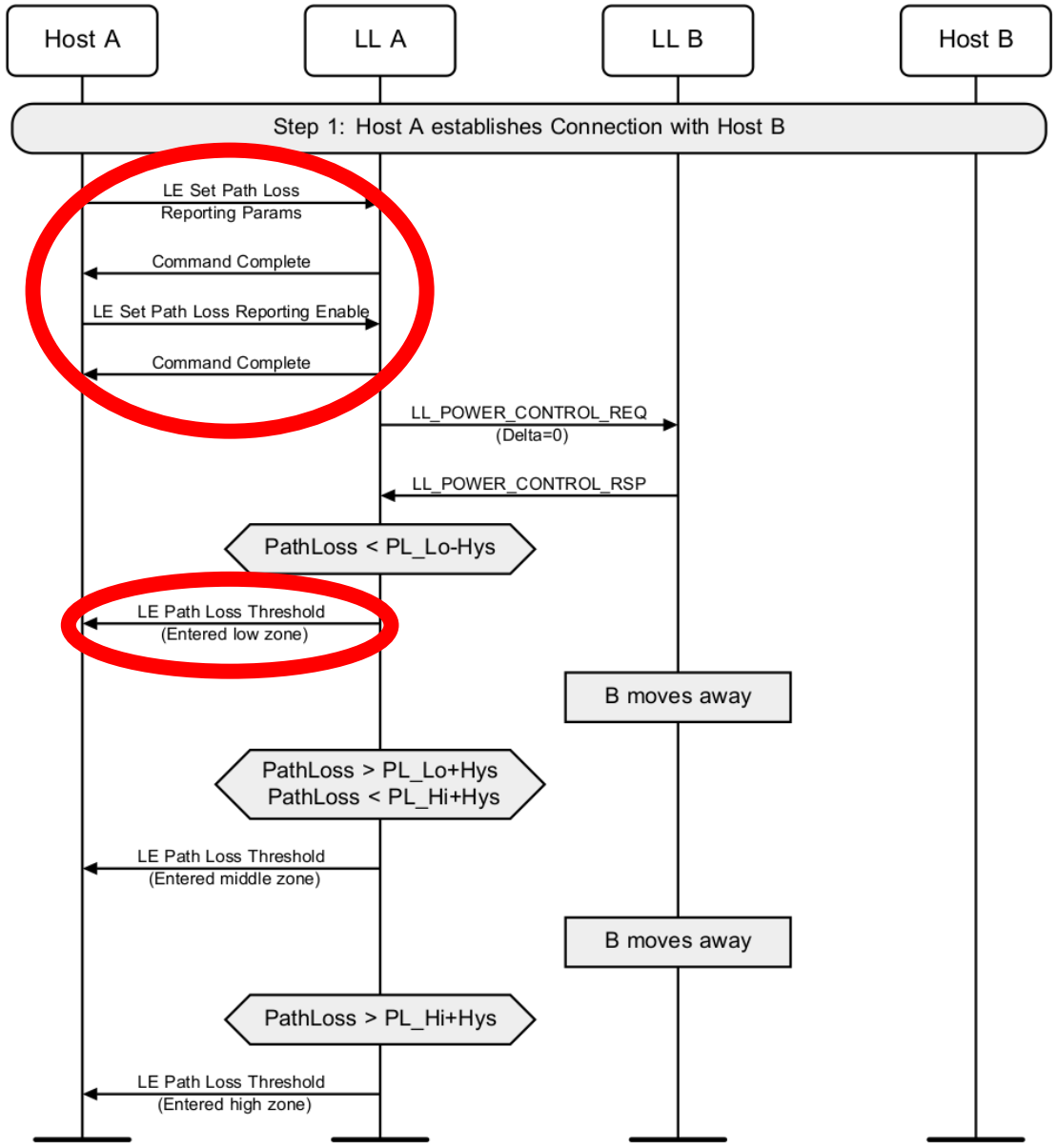
Path Loss Factor	TX Power	RSSI	Description
40	10 dBm	-30 dBm	As good of a signal as possible; unrealistic
70	10 dBm	-60 dBm	Great signal
80	10 dBm	-70 dBm	Okay signal
90	10 dBm	-90 dBm	Not good signal



Source: Bluetooth Core Specification v5.4

Example – Path Loss

- Host sets the **path loss parameters** and **enables** it
- Using the **transmit power** from the peer, perform **path loss factor** calculations.
- Once threshold is reached, report the **path loss zone** back to the host.

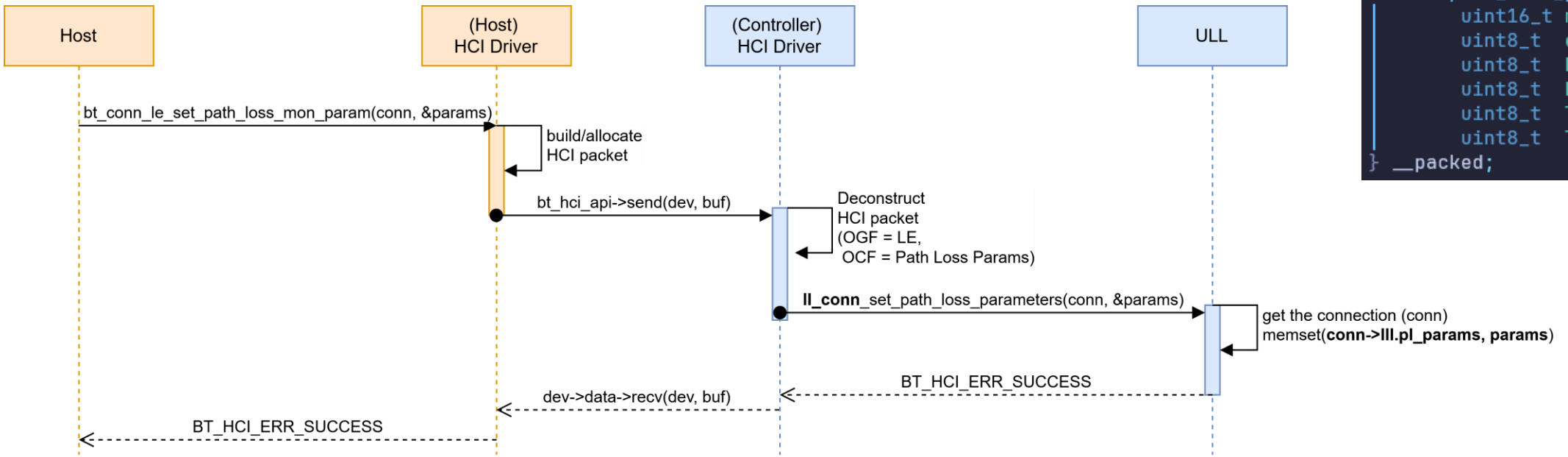


Source: Bluetooth Core Specification v5.4

Example – Path Loss

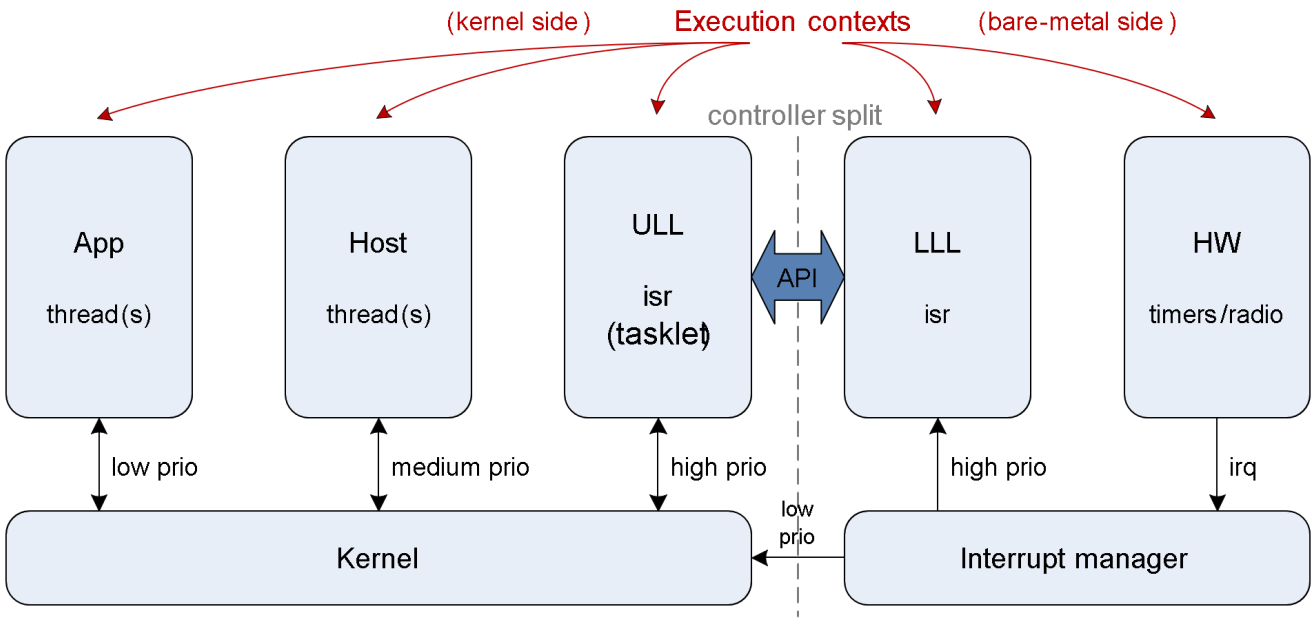
```
struct lll_conn {
        struct lll_hdr hdr;
#if defined(CONFIG_BT_CTLR_LE_PATH_LOSS_MONITORING)
        struct path_loss_params pl_params;
        struct path_loss_state pl_state;
        uint8_t pl_current_zone;
#endif /* CONFIG_BT_CTLR_LE_PATH_LOSS_MONITORING */

};
```

```
struct path_loss_params {
        uint16_t min_time_spent;
        uint8_t  enabled;
        uint8_t  high_threshold;
        uint8_t  high_hysteresis;
        uint8_t  low_threshold;
        uint8_t  low_hysteresis;
} __packed;
```



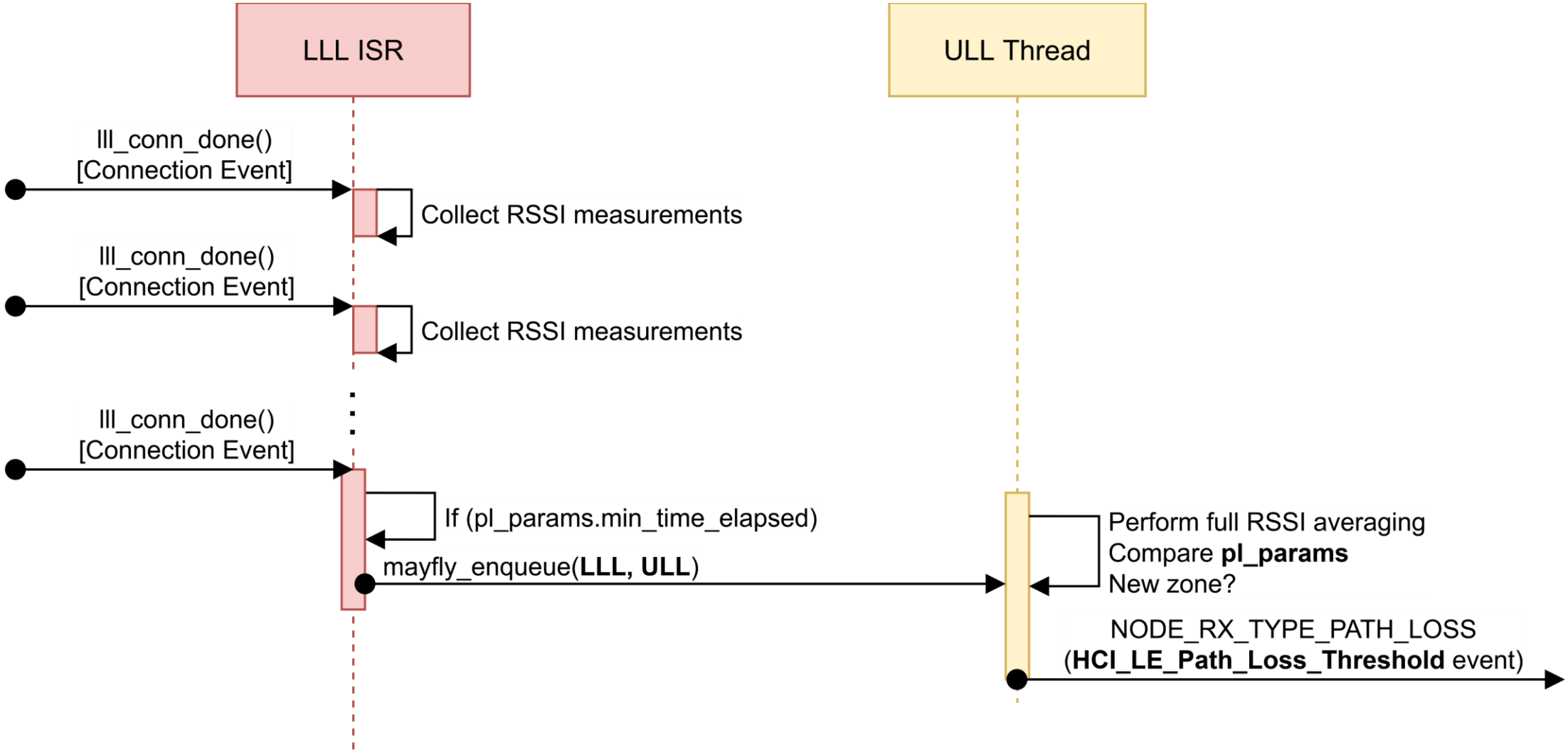
Execution contexts

- There are several interrupt contexts
  - Hard, real-time radio interrupts
  - Soft, real-item interrupts
  - Thread-based scheduling
- Running connection event (**lll_conn_done()**)
  - Performed in LLL
  - Push RSSI calculation up to ULL
  - If Path Loss Threshold update needed, push to HCI driver



Source: https://docs.zephyrproject.org/latest/connectivity/bluetooth/bluetooth-ctlr-arch.html

Example – Path Loss

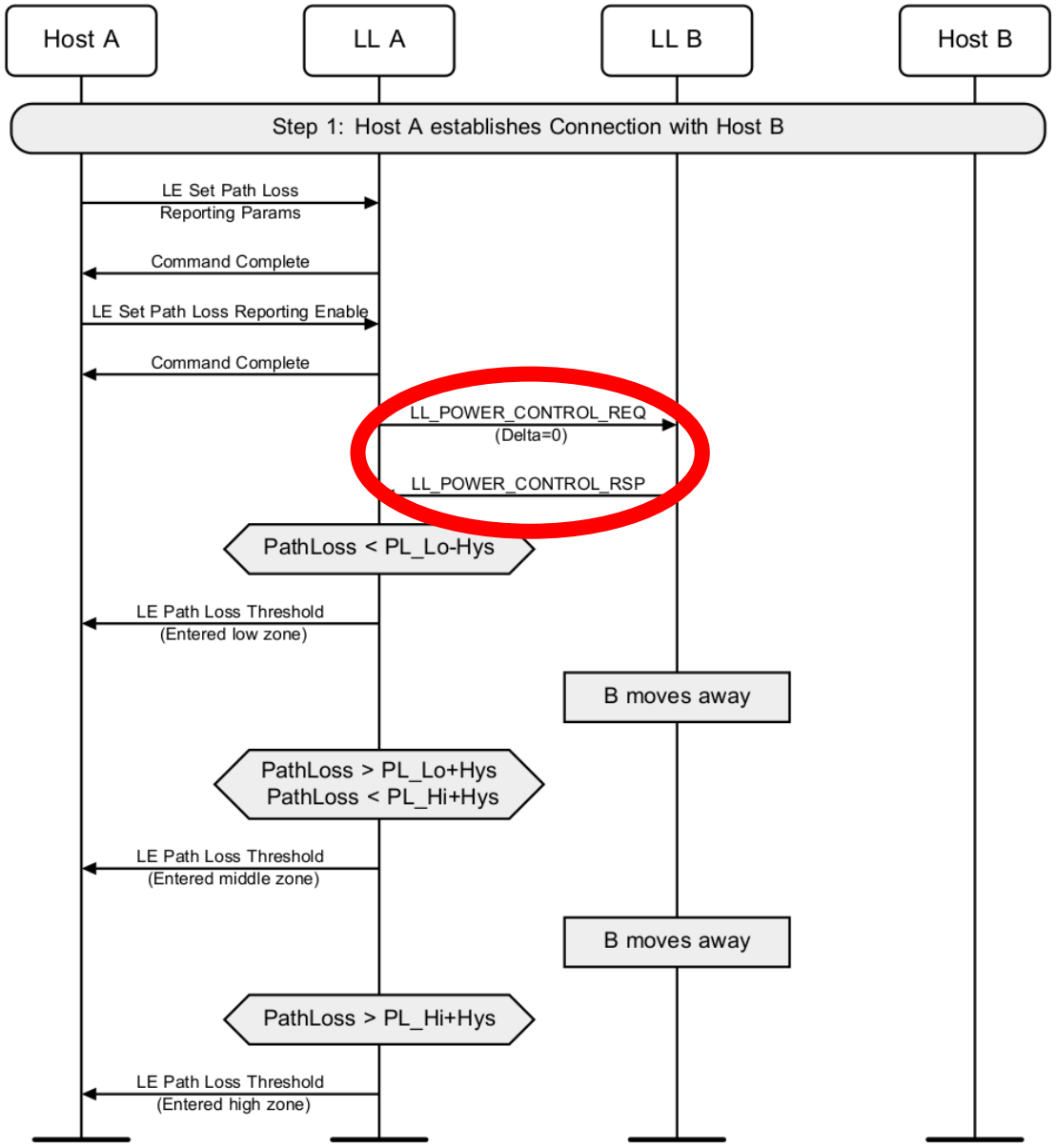


Example – Path Loss

```
uint8_t zone_entered = get_path_loss(lll_conn->pl_current_zone, TX_POWER - -lll_conn->rssi_latest, &lll_conn->pl_params);
if (zone_entered != lll_conn->pl_current_zone)
{
    lll_conn->pl_current_zone = zone_entered;
    path_loss_event_t event = {
        .curr_path_loss = TX_POWER - -lll_conn->rssi_latest,
        .zone_entered   = lll_conn->pl_current_zone
    };

    struct node_rx_pdu *rx_node = ull_pdu_rx_alloc();
    LL_ASSERT(rx_node);

    rx_node->hdr.type = NODE_RX_TYPE_PATH_LOSS;
    memcpy(rx_node->pdu, &event, sizeof(path_loss_event_t));

    ull_rx_put(rx_node->hdr.link, rx_node);
    ull_rx_sched();
}
```

Example – Power Control

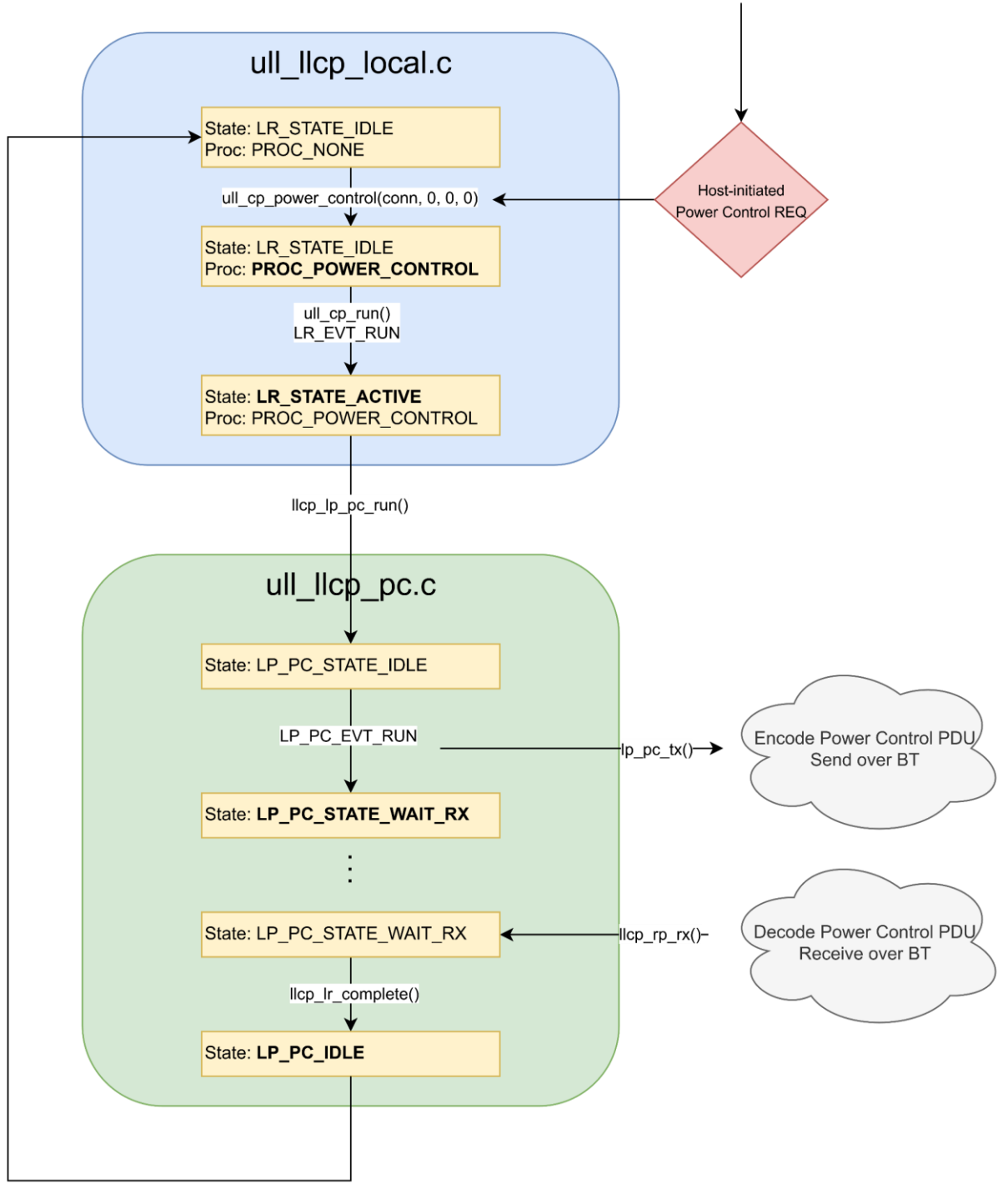
- Performs local and remote procedures
- Three kick-off points
  - Requests from Host
  - Controller-initiated requests
  - Responses to peer
- Driven by the LLCP state machine



Source: Bluetooth Core Specification v5.4

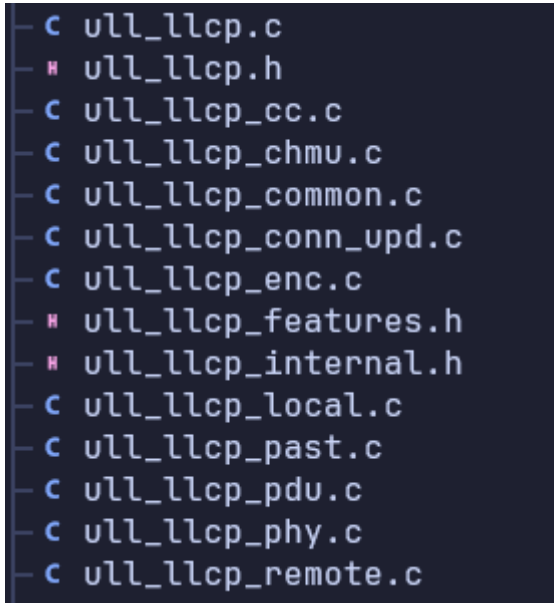
Example – Power Control

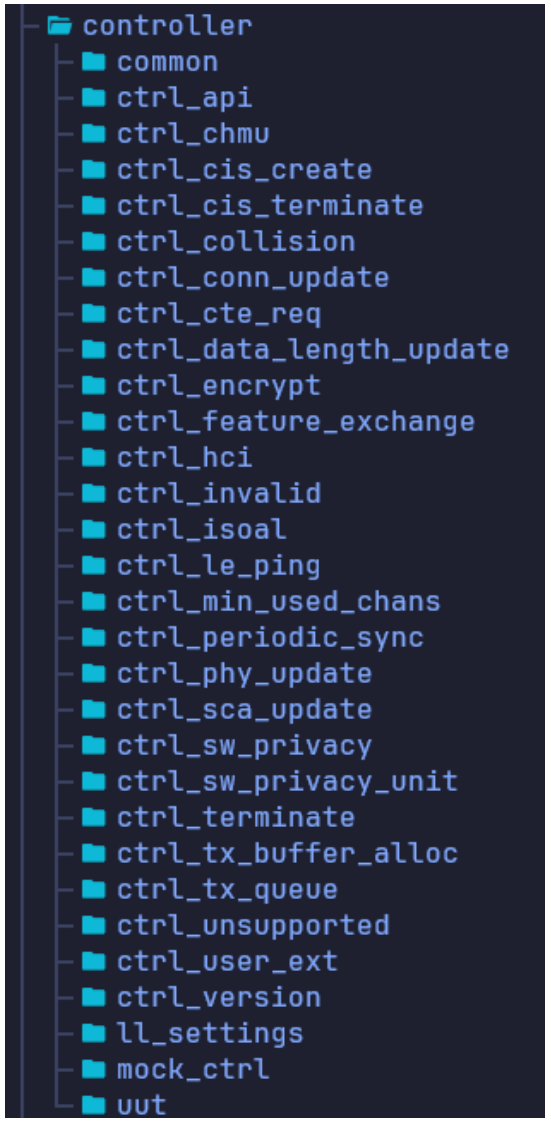
- Host-initiated LL_POWER_CONTROL_REQ
- Starts the local request state machine
- Forwards to the LLCP Power Control state machine
- Sends/receives PDUs over the air
- Cleans up once LL_POWER_CONTROL_RSP is received



Are there other LLCP state machines? Definitely!

- Lots of different LLCP state machines
- Twister tests used to facilitate the design
- Only one “remote request” and one “local request” by design





Getting involved

- Lots of examples for the nRF5x, just enable **CONFIG_BT_LL_SW_SPLIT=y**
  - Support for all projects under samples/bluetooth/*
- Unittests under tests/bluetooth/controller/ctrl_*
  - Mostly for LLCP
  - gdb, step debugging, control flows
- Babblesim tests under tests/bsim/bluetooth/ll/*

Try it yourself!

- samples/bluetooth/peripheral/prj.conf

```
# Increased stack due to settings API usage
CONFIG_SYSTEM_WORKQUEUE_STACK_SIZE=2048

CONFIG_BT=y
CONFIG_LOG=y
CONFIG_BT_SMP=y
CONFIG_BT_SIGNING=y
CONFIG_BT_PERIPHERAL=y
CONFIG_BT_DIS=y
CONFIG_BT_ATT_PREPARE_COUNT=5
CONFIG_BT_BAS=y
CONFIG_BT_HRS=y
CONFIG_BT_IAS=y
CONFIG_BT_CTS=y
CONFIG_BT_CTS_HELPER_API=y
CONFIG_BT_PRIVACY=y
CONFIG_BT_DEVICE_NAME="Zephyr Peripheral Sample Long Name"
CONFIG_BT_DEVICE_APPEARANCE=833
CONFIG_BT_DEVICE_NAME_DYNAMIC=y
CONFIG_BT_DEVICE_NAME_MAX=65

CONFIG_BT_KEYS_OVERWRITE_OLDEST=y
CONFIG_BT_SETTINGS=y
CONFIG_FLASH=y
CONFIG_FLASH_MAP=y
CONFIG_NVS=y
CONFIG_SETTINGS=y
CONFIG_BT_HCI=y
CONFIG_BT_LL_SW_SPLIT=y
```

```
[nix-shell:/scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral]$ west build -b nrf52dk/nrf52832 .
-- west build: generating a build system
Loading Zephyr default modules (Zephyr base).
-- Application: /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral
-- CMake version: 3.25.1
-- Found Python3: /scratch/tyhu/zephyrproject/venv/bin/python3 (found suitable version "3.12.11", minimum required is "3.10") found components: Interpreter
-- Cache files will be written to: /home/tyhu/.cache/zephyr
-- Zephyr version: 4.2.99 (/scratch/tyhu/zephyrproject/zephyr)
-- Found west (found suitable version "1.5.0", minimum required is "0.14.0")
-- Board: nrf52dk, qualifiers: nrf52832
-- ZEPHYR_TOOLCHAIN_VARIANT not set, trying to locate Zephyr SDK
-- Found host-tools: zephyr 0.17.2 (/home/tyhu/zephyr-sdk-0.17.2)
-- Found toolchain: zephyr 0.17.2 (/home/tyhu/zephyr-sdk-0.17.2)
-- Found Dtc: /home/tyhu/zephyr-sdk-0.17.2/sysroots/x86_64-pokysdk-linux/usr/bin/dtc (found suitable version "1.7.0", minimum required is "1.4.6")
-- Found BOARD.dts: /scratch/tyhu/zephyrproject/zephyr/boards/nordic/nrf52dk/nrf52dk_nrf52832.dts
-- Generated zephyr.dts: /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/zephyr.dts
-- Generated pickled edt: /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/edt.pickle
-- Generated devicetree_generated.h: /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/include/generated/zephyr/devicetree_generated.h
Parsing /scratch/tyhu/zephyrproject/zephyr/Kconfig
Loaded configuration '/scratch/tyhu/zephyrproject/zephyr/boards/nordic/nrf52dk/nrf52dk_nrf52832_defconfig'
Merged configuration '/scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/prj.conf'
Configuration saved to '/scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/.config'
Kconfig header saved to '/scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/include/generated/zephyr/autoconf.h'
-- Found GnuLd: /home/tyhu/zephyr-sdk-0.17.2/arm-zephyr-eabi/arm-zephyr-eabi/bin/ld.bfd (found version "2.38")
-- The C compiler identification is GNU 12.2.0
-- The CXX compiler identification is GNU 12.2.0
-- The ASM compiler identification is GNU
-- Found assembler: /home/tyhu/zephyr-sdk-0.17.2/arm-zephyr-eabi/bin/arm-zephyr-eabi-gcc
-- Using ccache: /usr/bin/ccache
-- Found gen_kobject_list: /scratch/tyhu/zephyrproject/zephyr/scripts/build/gen_kobject_list.py
-- Configuring done
-- Generating done
-- Build files have been written to: /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build
-- west build: building application
[1/374] Preparing syscall dependency handling

[3/374] Generating include/generated/zephyr/version.h
-- Zephyr version: 4.2.99 (/scratch/tyhu/zephyrproject/zephyr), build: v4.2.0-5354-gc8d91030f8a4
[374/374] Linking C executable zephyr/zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:      198144 B       512 KB     37.79%
             RAM:       30341 B        64 KB     46.30%
        IDT_LIST:          0 GB        32 KB      0.00%
Generating files from /scratch/tyhu/zephyrproject/zephyr/samples/bluetooth/peripheral/build/zephyr/zephyr.elf for board: nrf52dk
((venv) )
```

Features missing

- LE Connection Subrating
- LE Periodic Advertising with Responses
- LE Frame Space Update
- Channel Sounding
- Bluetooth v6.0/v6.1 features

Thank you