NimBLE - portable Bluetooth stack from Apache Mynewt

Szymon Janc
szymon.janc@codecoup.pl

RIOT Summit, Amsterdam, 2018
Agenda

- About
- Short Bluetooth Low Energy introduction
- Apache Mynewt
- NimBLE
- Supported BLE features
- NimBLE Bluetooth LE stack architecture
  - controller
  - host
- GAP (Scanning, Advertising, Pairing etc)
- GATT
- NimBLE Ports
- Future work
About

● Me
  ○ Embedded software engineer
  ○ Works with embedded Linux and Android platforms since 2007
  ○ Since 2015 involved in couple RTOSes development
  ○ Focused on Local Connectivity (Bluetooth, NFC)
  ○ Open Source contributor (BlueZ, Linux, Zephyr, Apache Mynewt)

● Codecoup
  ○ Founded in 2015
  ○ Support in Bluetooth, Linux, Android, RTOS, Open Source, embedded systems
  ○ Internet of Things projects
  ○ [www.codecoup.pl](http://www.codecoup.pl)
History of Bluetooth Low Energy

- Introduced with Bluetooth 4.0 released in June 2010
- Bluetooth 4.1 released in December 2013
  - Link Layer Topology
  - LE L2CAP Connection Oriented Channels
- Bluetooth 4.2 released in December 2014
  - LE Secure Connection
  - Link Layer Privacy
  - Data Length Extensions (up to 2.5x speed increase)
  - IP Support Profile released
- Bluetooth 5 released in December 2016
  - 2M PHY
  - Coded PHY (LE Long Range)
  - Advertising Extensions, Periodic Advertising
- Bluetooth Mesh released in July 2017
Technology overview

- Much simpler comparing to Bluetooth Classic
- Short range wireless technology (typically up to 100 meters, possible 1km+)
- Operates at 2.4 GHz (IMS band)
- Designed for low power usage (coin battery)
- Fast connection establishment
- Simple modulation to minimize transceiver complexity
- Multiple PHYs supported (Bluetooth 5)
  - 1 Mbps bandwidth over the air (~0.27 Mbps max throughput for applications)
  - 2 Mbps bandwidth over the air
  - Coded PHY - S2 and S8 coding
- Frequency hopping to combat interferences
- Use both FDMA and TDMA
Technology overview (II)

- Short data packets (27 bytes, 255 with Data Length Extension)
- 40 physical channels
  - 3 for advertising, 37 for data (Bluetooth 4.2 and older)
  - All channels used for advertising with Advertising Extensions (Bluetooth 5)
- Channel selection algorithms #1 and #2
  - #2 introduced in Bluetooth 5 improves spectrum usage and WiFi coexistence
Technology overview (III)

- **Connectionless roles**
  - observer and broadcaster

- **Connection oriented roles**
  - central and peripheral

- **Security**
  - Encryption and authentication

- **Generic Attribute Profile (GATT)**
  - Server exposes list of attributes (characteristics grouped as services) called database
  - Client can discover services, read, write and enable notifications

- **L2CAP Connection Oriented Channels**
The Bluetooth Special Interest Group (SIG) is the body that oversees the development of Bluetooth standard (Core specification, profile specifications)

- Over 30000 companies are part of Bluetooth SIG as of now
- Two levels of membership available:
  - Adopter (free) → license to build products and use trademarks
  - Associate (paid) → access to working groups, i.e. can actively develop specification

Adopted Core and Profile specifications and corresponding test specifications are available publicly on Bluetooth SIG website


Qualification Process is available to ensure newly developed devices are compliant with specification

- PTS test tool (free for members) provides semi-automation for testing/qualification process
Apache Mynewt

- An Open Source RTOS for 32-bit MCUs
- Permissive Apache 2.0 license
- Community driven under Apache Foundation
- Small memory requirements (<16KB RAM, <64KB Flash)
- Rich in features
  - Preemptive RTOS, multitasking, mutexes, semaphores, timers etc.
  - Unified buffer management (mbuf)
  - Event driven model (timers, IO etc.)
  - Flash filesystem (NFFS)
  - Console - UART and RTT
  - Networking (Bluetooth 5, Bluetooth Mesh, LoRa, COAP and more)
  - Secure bootloader and image update

https://mynewt.apache.org
Apache Mynewt (II)

- Modular
  - Most components come as packages
  - Application enables only packages it uses
- Highly configurable
- Release every 3-4 months
  - 1.4.1 released July 2018
  - 1.5 expected October 2018
- Comes with own build and packages management system (newt)
  - Used for configuring, building, installing and debugging
- Support for Linux, MacOS and Windows
- Portable
  - ARM Cortex-M (M0/M3/M4/M7), MIPS, RISC-V, ARC
  - nRF52DK, nRF52840, nRF51DK, RuuviTag, BLE Nano/Nano2, STM32F4DISCOVERY, Arduino Zero, NUCLEO-F401RE/F767ZI, FRDM-K64F, BBC micro:bit, Adafruit FeatherPortable, HiFive1 DevKit
Apache NimBLE

- Originated as part of Apache Mynewt project
- Permissive Apache 2.0 license
- Community driven under Apache Foundation
- [https://github.com/apache/mynewt-nimble](https://github.com/apache/mynewt-nimble)
- 1.0.0 released on June 2018
  - Future releases planned every few months
  - Released independently of Apache Mynewt
- BT SIG Qualifiable (host)
NimBLE Bluetooth Low Energy features

- Core Specification 5.0
  - 1M, 2M and Coded PHY
  - Advertising Extensions
- Low Energy only
- Generic Access Profile (GAP)
  - central, peripheral, observer, broadcaster
  - privacy
  - multiple concurrent roles
- Security Manager
  - Legacy Pairing, Secure Connections
- Generic Attribute Profile (GATT)
- L2CAP Connection Oriented Channels
- Bluetooth Mesh
NimBLE architecture

- Split between host and controller allows for different builds
  - combined host + controller
  - host-only works with external controller
  - controller-only works with external host
- Fully configurable using syscfg parameters
  - number of supported connections, max packet lengths etc. can be configured
  - features can be enabled/disabled to adjust RAM/flash usage
- Decoupled to separate repository (mynewt-nimble) for easier reuse
NimBLE controller

- Complete Link Layer implementation
- Uses standard HCI interface
- RF drivers available for various SoCs from Nordic Semiconductor
  - nRF51xxx (without 2M and Coded PHY)
  - nRF52xxx (without Coded PHY)
  - nRF52840
- Support for other RF/SoC possible
- Can be used without NimBLE host (blehci)
  - interfaces with any Bluetooth host stack using UART H4 transport
  - works with BlueZ
  - 28 kB ROM / 2 kB RAM (default)
  - 39 kB ROM / 3 kB RAM (all features)
NimBLE HCI transport

- Combined (host + controller) build uses shared memory for HCI transport (nimble/transport/ram)
- External controllers can be used with other transports
  - UART H4 (nimble/transport/uart) standard UART H4 interface
  - Socket transport (nimble/transport/socket) IPC socket for interfacing controller on Linux host (native simulator build)
  - SPI (nimble/transport/emspi) proprietary HCI transport for interfacing controllers made by EM Microelectronics
NimBLE host

- Highly configurable
  - complete features can be disabled to reduce memory usage
  - even single GATT procedures can be disabled, if necessary
  - 50 kB ROM / 3 kB RAM (default)
  - 65 kB ROM / 4 kB RAM (L2CAP CoC, Advertising Extensions, SM SC and Mesh)
NimBLE host API

- Extensive API to use all host features
- API allows for detailed control over all Bluetooth parameters
  - unlike e.g. Linux or iOS which provide more high-level APIs thus hiding some details

- GAP API available in ble_gap.h

  ```c
  int ble_gap_adv_start(uint8_t own_addr_type, const ble_addr_t *direct_addr, int32_t duration_ms,
                        const struct ble_gap_adv_params *adv_params, ble_gap_event_fn *cb, void *cb_arg);
  int ble_gap_disc(uint8_t own_addr_type, int32_t duration_ms, const struct ble_gap_disc_params *disc_params,
                   ble_gap_event_fn *cb, void *cb_arg);
  int ble_gap_connect(uint8_t own_addr_type, const ble_addr_t *peer_addr, int32_t duration_ms,
                      const struct ble_gap_conn_params *params, ble_gap_event_fn *cb, void *cb_arg);
  ```
NimBLE host API (2)

- **GATT client and server API available in `ble_gatt.h`**
- **All GATT client procedures available as function calls**

```c
int ble_gattc_disc_all_svcs(uint16_t conn_handle, ...);
int ble_gattc_disc_all_chrs(uint16_t conn_handle, uint16_t start_handle, uint16_t end_handle, ...);
int ble_gattc_read(uint16_t conn_handle, uint16_t attr_handle, ...);
int ble_gattc_read_long(uint16_t conn_handle, uint16_t handle, uint16_t offset, ...);
int ble_gattc_write_no_rsp(uint16_t conn_handle, uint16_t attr_handle, ...);
int ble_gattc_write(uint16_t conn_handle, uint16_t attr_handle, ...);
int ble_gattc_notify(uint16_t conn_handle, uint16_t chr_val_handle, ...);
```

- **...and more!**
NimBLE host API (3)

- GATT database can be registered using full service description

```c
static const struct ble_gatt_svc_def gatt_svr_svcs[] = {
    {
        /* Service: Heart-rate */
        .type = BLE_GATT_SVC_TYPE_PRIMARY,
        .uuid = BLE_UUID16_DECLARE(GATT_HRS_UUID),
        .characteristics = (struct ble_gatt_chr_def[]) { {
            /* Characteristic: Heart-rate measurement */
            .uuid = BLE_UUID16_DECLARE(GATT_HRS_MEASUREMENT_UUID),
            .access_cb = gatt_svr_chr_access_heart_rate,
            .val_handle = &hrs_hrm_handle,
            .flags = BLE_GATT_CHR_F_NOTIFY,
        }, {
            /* Characteristic: Body sensor location */
            .uuid = BLE_UUID16_DECLARE(GATT_HRS_BODY_SENSOR_LOC_UUID),
            .access_cb = gatt_svr_chr_access_heart_rate,
            .flags = BLE_GATT_CHR_F_READ,
        }, 0, /* No more characteristics in this service */
    },
    },
};

/* ... */
ble_gatts_add_svcs(gatt_svr_svcs);
```
Applications samples

- Located in apps/ folder
- Blinky - ‘Hello World’ sample
- Provide code reference for API usage for most subsystems
  - blehci, blehr, bleprh, btshell - BLE samples
  - blemesh - Bluetooth Mesh
- btshell - application that enables all BLE features and provide console shell for user to control every aspect of NimBLE stack
NimBLE Ports

- Support for multiple OSes
  - Apache Mynewt
  - FreeRTOS
  - RIOT
  - RIOT
  - Linux
NimBLE updates for porting

- Move NimBLE to separate repository (was as subdir in Mynewt core)
- Make sure only OS API calls are used to interact with OS
  - Refactored code which accessed various OS structures directly (not portable)
  - Missing OS API calls added
- Fix build outside Mynewt tree
  - Unused dependencies (#include) to Mynewt components removed
  - Build with stubbed Mynewt-specific subsystems
  - Add conditional compilation for Mynewt-specific code
- Port necessary Mynewt structures to external builds
  - Memory pools and memory buffers (mbufs) are inherent structures for data buffers in Mynewt and thus also NimBLE
- Change OS API calls to portable versions (os_* → ble_npl_*)
NimBLE 1.0+ (with NPL)

- **OS abstraction layer defined** (nimble/include/nimble/nimble_npl.h)
  - Events and event queues
  - Mutexes
  - Semaphores
  - Callouts (timers)
  - Ticks time handling
  - Few auxiliary calls (mostly to make controller integration easier)

- **Common code required to build NimBLE** (porting/nimble)
  - Components “extracted” from Mynewt (os_mbuf, os_mempool, os_cputime)
  - Stubbed headers for subsystems specific to Mynewt (logs, stats, trace)
  - NimBLE initialization code

- **NPL implementation for supported OS-es** (porting/npl)
  - Mynewt also implements NPL - it’s just a shim to call original os_* APIs
NimBLE in RIOT OS

- NPL for RIOT OS merged to NimBLE repository (porting/npl/riot)
- NimBLE provided in RIOT OS as package (pkg/nimble)
  - Built from NimBLE repository *without* extra patches required (minor workarounds required by RIOT OS port are merged to NimBLE repository)
  - Minimal extra setup required - see sample app in examples/nimble_gatt
- Works:
  - All Bluetooth Core features implemented in NimBLE
  - nRF52xxx MCU (combined controller + host build with HCI over shared memory)
    (note: controller needs exclusive access to TIMER0, RTC0, RADIO, CCM and AAR)
- Does not (yet) work, but possible:
  - Host on other MCUs (possible with specific HCI transport implementation)
  - Bluetooth Mesh (not ported in NimBLE 1.0, will be available in next NimBLE release)
- Implemented during RIOT hackathon on May 2018 in Paris
  - Credits: Andrzej Kaczmarek and Hauke Petersen
Future Work

- BT SIG qualification of controller
- Support for periodic advertising
- More samples implementing Bluetooth Profiles
- Improvements to Mesh (sync with Zephyr)
Community and Contributing

- Found a bug or work on new feature? Contribute!
- Mynewt NimBLE uses GitHub for development
  - [https://github.com/apache/mynewt-nimble](https://github.com/apache/mynewt-nimble)
  - Pull requests should be sent against master branch
- Discussions
  - Mailing list: [dev@mynewt.apache.org](mailto:dev@mynewt.apache.org)
  - Slack channel: [https://mynewt.slack.com/messages](https://mynewt.slack.com/messages)
- More information on [https://mynewt.apache.org](https://mynewt.apache.org)
NimBLE - portable Bluetooth stack from Apache Mynewt

Szymon Janc
szyzon.janc@codecoup.pl

RIOT Summit, Amsterdam, 2018