LoRa Communications in Tock

TockWorld 6 - 2023
Branden Ghena

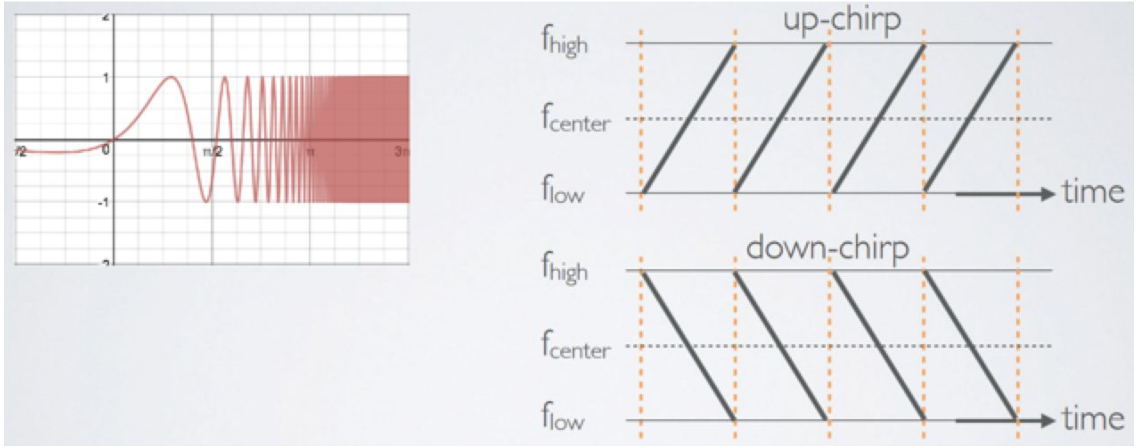
What is LoRaWAN?

- Open communication standard built with proprietary LoRa physical layer
  - LoRa owned by Semtech who makes transceiver chips, but some collaborations: STM32WL
- Low rate (1-20 kbps) and long range (~5 km)
  - With relatively low energy costs
- Most popular low-power wide-area network (LPWAN)
  - Target of academic research
  - Industry involvement in hardware and deployments

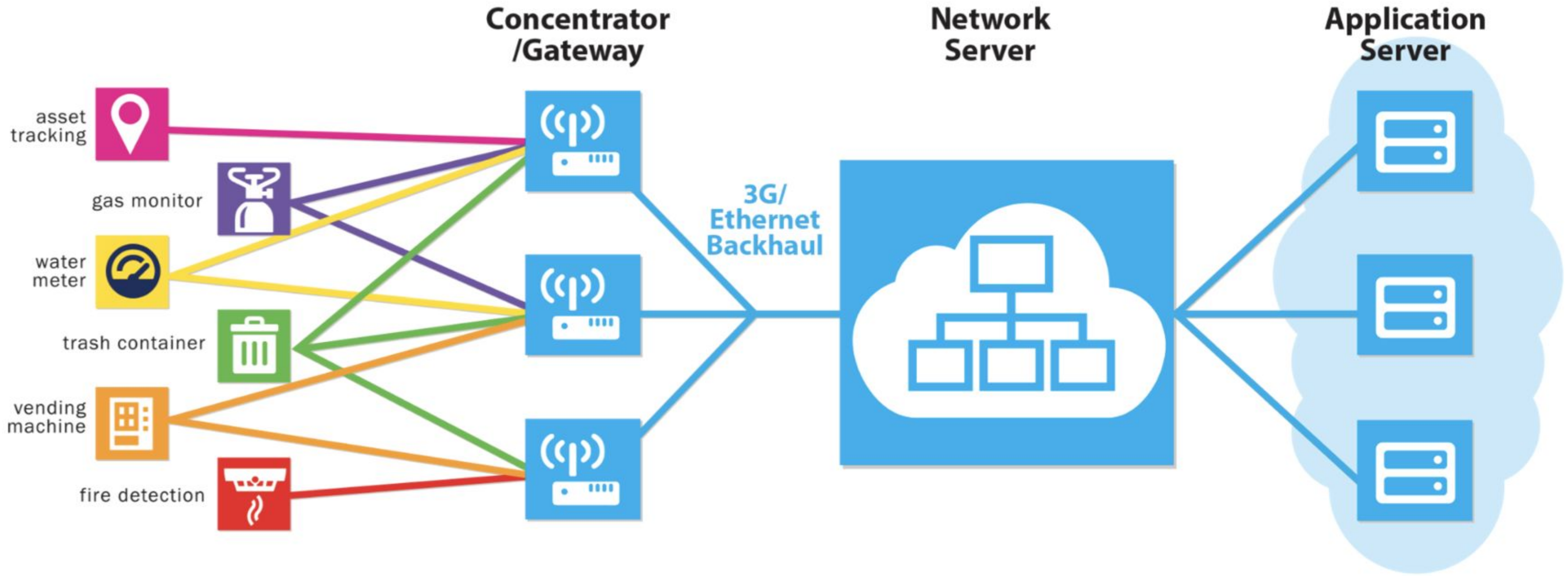
What is LoRa?

- LoRa is a proprietary long-range physical layer communication standard
  - Specifies how to send bits wirelessly
  - Uses Chirp Spread Spectrum (CSS) modulation: technique where frequency is varied linearly from lowest to highest in a channel

- Proprietary: owned by Semtech who makes radio chips for it
  - Some collaboration recently STM32WL



LoRaWAN network details



Tock Support

- Tracking issue: https://github.com/tock/tock/issues/2344

- Focus is on communicating with external transceivers, not built-in radios as they are much less common
    - Usual design is a microcontroller + Semtech SX1262 or SX127x over SPI

Existing support in Tock

- Work by Alistair that provides direct SPI access to userspace
  - Along with Libtock-C driver to control the LoRa transceiver over it
  - Raw LoRa packets

- Tock board support for SparkFun LoRa Thing Plus “expLoRaBLE”
  - Apollo3 microcontroller
  - SX1262 LoRa transceiver

- Work completed in Tock in June 2023
  - https://github.com/tock/tock/pull/3330
  - https://github.com/tock/tock/pull/3360
  - https://github.com/tock/libtock-c/pull/317

Two false-starts for LoRa capsules

- April 2020
  - SPI: https://github.com/tock/tock/pull/1760
  - Work by Nitish Kulshrestha UCSD
  - Raw LoRa packets
- June 2021
  - I2C: https://github.com/tock/tock/pull/2615
  - Work by Olivia Weng UCSD
  - Specific for LoRa-MAC-in-C (LMIC) library
- Both ended up stale after student progress stopped

Takeaways

- Non-trivial to make something real (hence the false starts)

- Raw LoRa packets is relatively simple

- LoRaWAN protocol is more complex
  - Relying on userland library to do most of the lifting seems plausible
  - Generally not very time sensitive, which makes this realistic

- Deciding on a hardware board to support that is available would be useful
  - SparkFun expLoRaBLE board is a reasonable place to start

Bluetooth Low Energy (BLE) in Tock

TockWorld6 – Pat Pannuto

Basics of BLE

- Direct device-to-device communication
  - Usually: Computer to Thing
  - Smartphone to device, Laptop to device, etc.

- Focus on making the “Thing” really low energy
  - Push energy-intensive requirements onto “Computer”

- Devices (Computer or Thing) are servers with accessible fields
  - Not the traditional send-explicit-packets interface you might be expecting
  - Lower layers are still exchanging packets to make it work

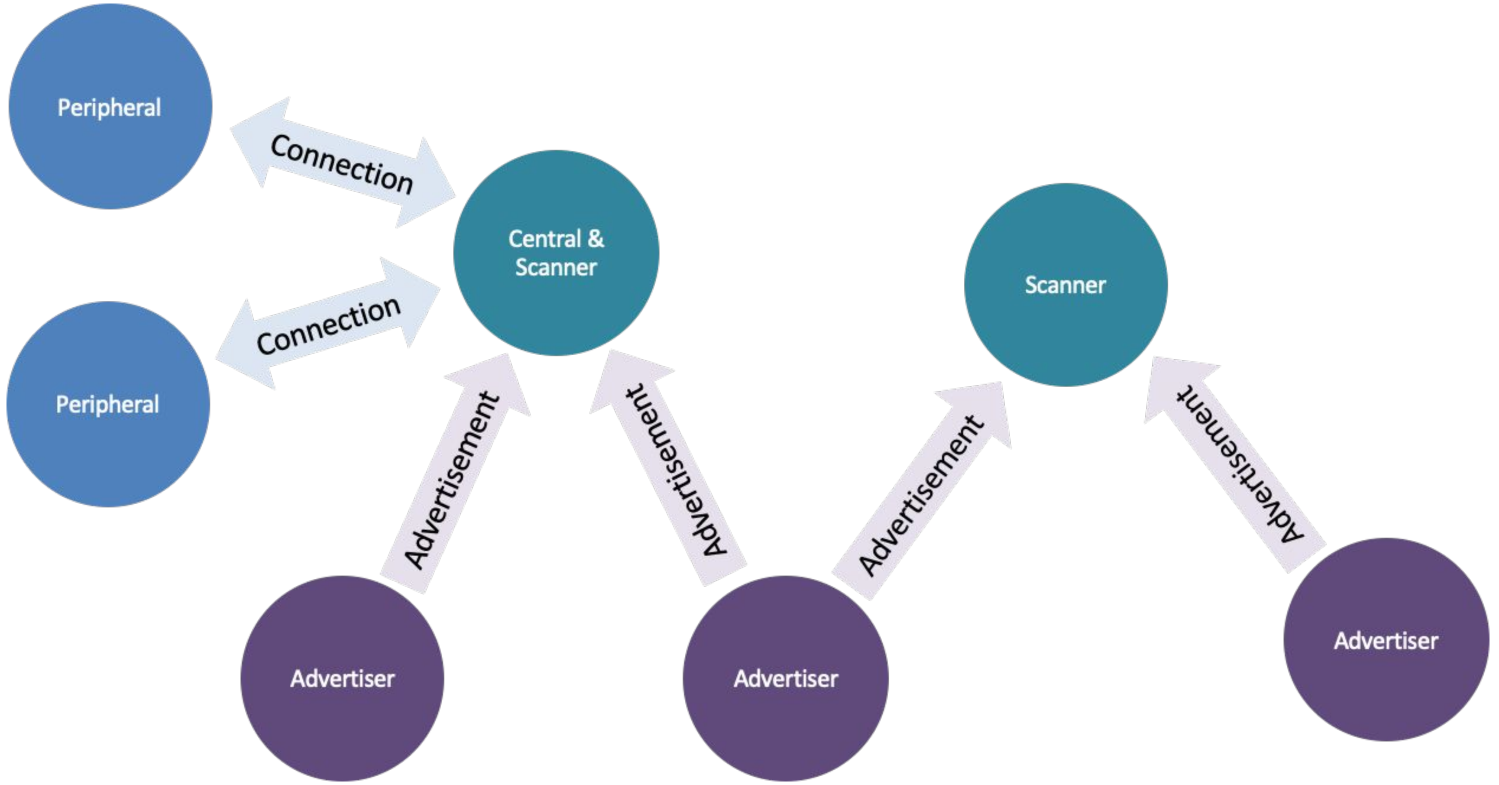
What is Bluetooth?

A *very* large specification with a long history at this point

- 5.2 spec: **3256 pages**

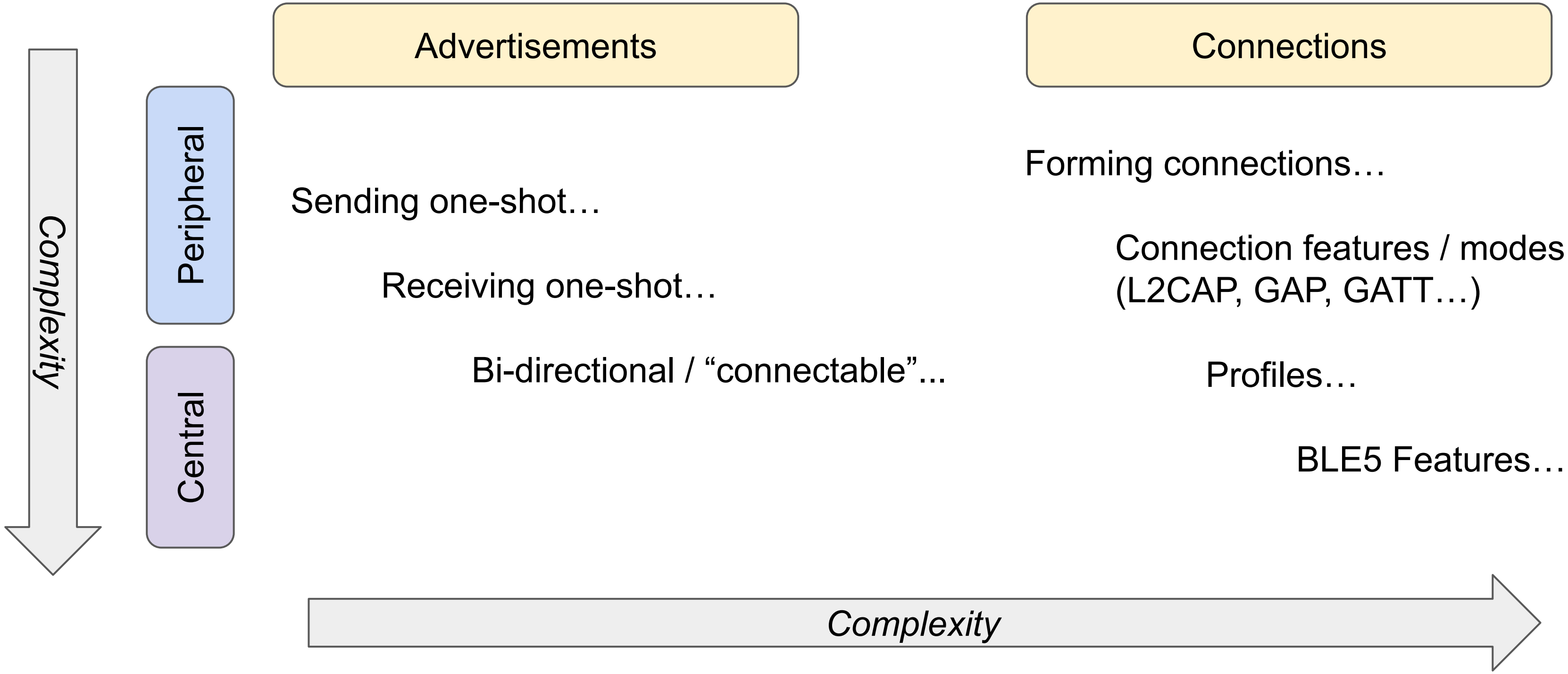
- Most “IoT” use cases can restrict to *Vol 6: Low Energy Controller*
  - Part A: Physical Layer Specification
  - Part B: Link Layer Specification
  - CSS: Part A: Data Types Specification
  - So ~250 pages

BLE Network Toplogy



What is Bluetooth

A *very* broad spec, which has logical chunks of features:

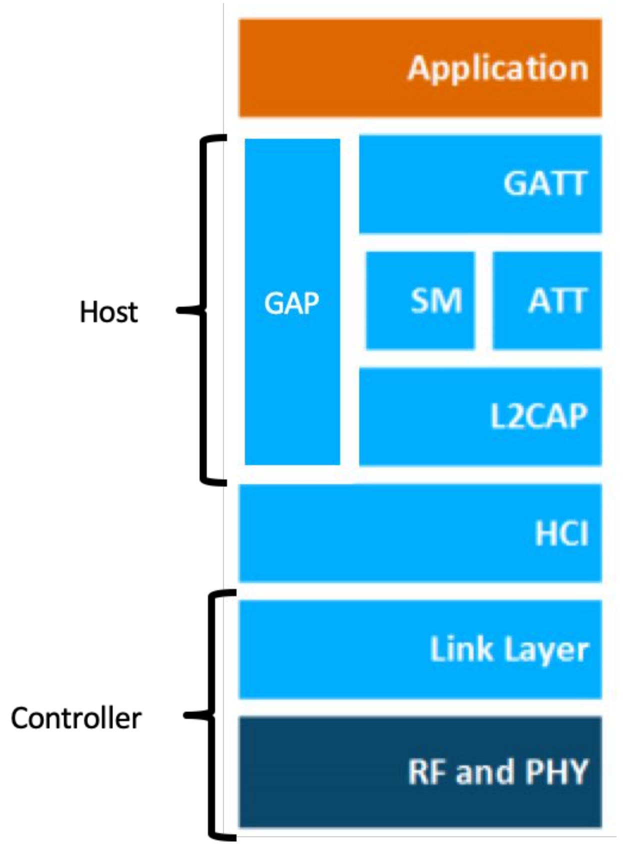


Formally, BLE defines its implementation into layers

- Host – Configuration and Server
  - GAP – Generic Access Profile
    - Configure advertising
  - GATT – Generic ATTribute profile
    - Configure connections

- HCI - Host Controller Interface

- Controller - Communication
  - Link Layer – send packets
  - RF and PHY – send bits



How does Tock implement (“implement”?) BLE?

Two major approaches:

- In-kernel
  - Written from scratch
  - HIL and syscall interfaces provide high-level operations
- Userspace
  - Use vendor / third-party stacks
  - Much (*much*) more complete implementation
  - Various attempts, only one has really ‘stuck’ to date

Userspace / Vendor BLE stacks

- Most-used is Nordic's "`nrf-serialization`" in libtock-c
  - BLE stack actually runs on remote MCU
  - BLE stack is all Nordic, black-box code
    - (almost certainly a C library)
  - High-level commands/events are passed across an RPC interface

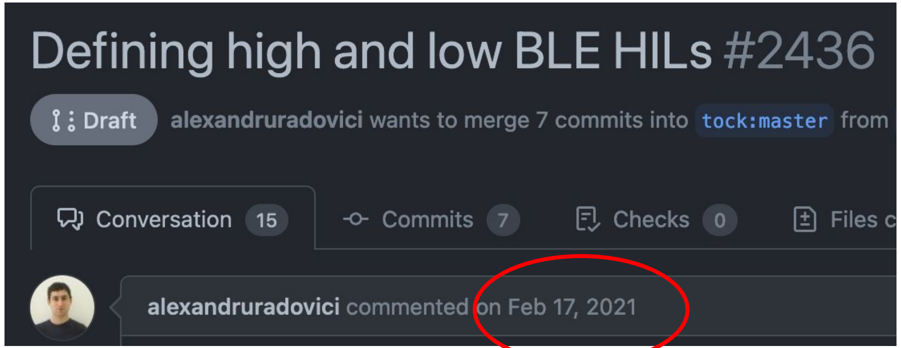
- Several false-starts on integrating third-party / open-source on-device stacks
  - NimBLE, Apache MyNewt, probably others
  - Sad reality: There is not a great FOSS BLE stack
  - Those that exist are all large, complex C libraries

Tock’s in-kernel support

- `kernel/src/hil/ble_advertising.rs`
  - Interfaces to send and receive advertisements
    - One-shot only — i.e., cannot handle connectable advertisements
  - Implemented by two chips to-date:
    - `chips/nrf52/src/ble_radio.rs`
    - `chips/apollo3/src/ble.rs`
- `capsules/extra/src/ble_advertising_driver.rs`
  - Provides {start,stop} {advertising,scanning}
    - User-configurable intervals, but most timing/events handled in-kernel
  - Multiplexing
    - Each process is its own “BLE Device”, i.e. given unique static address

State of BLE development (or lack thereof)

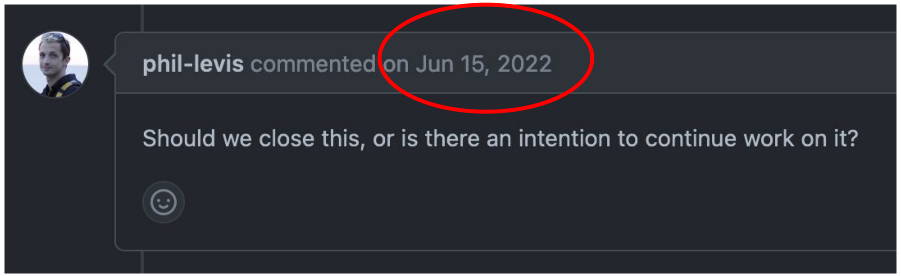
- →


Defining high and low BLE HILs #2436

Draft alexandruradovici wants to merge 7 commits into tock:master from

alexandruradovici commented on Feb 17, 2021

- →


phil-levis commented on Jun 15, 2022

Should we close this, or is there an intention to continue work on it?

Some cold reality: More features are big chunks

- Has not been a lot of demand for BLE
  - Or… we don’t see it because what we have is so far away that demand never bothers

- Perfect may be the enemy of the good here
  - Bluetooth HCI is famously bad, but inventing our own has not gone anywhere either

- We should consider leaning into getting smaller pieces working, at expense of generality in the short term
  - Perhaps an emphasis on ‘making it easy to be a simple peripheral’?