Doing Bluetooth Low Energy on Linux

Szymon Janc

szymon.janc@codecoup.pl

CODECOUP

OpenIoT Summit Europe, Berlin, 2016

Agenda

- Introduction
- Bluetooth Low Energy technology recap
- Linux Bluetooth stack architecture
  - Linux kernel
  - BlueZ 5
- GAP (Scanning, Advertising, Pairing etc)
- GATT
- LE CoC and 6LoWPAN
- Custom solutions
- Tips
- Future work

About me

- Embedded software engineer
- Works with embedded Linux and Android platforms since 2007
- Focused on Local Connectivity (Bluetooth, NFC)
- Open Source contributor (BlueZ, Linux, Zephyr)

- In 2015 co-founded Codecoup
  - support in Bluetooth, Linux, Android, Open Source, embedded systems
  - Internet of Things projects
  - www.codecoup.pl

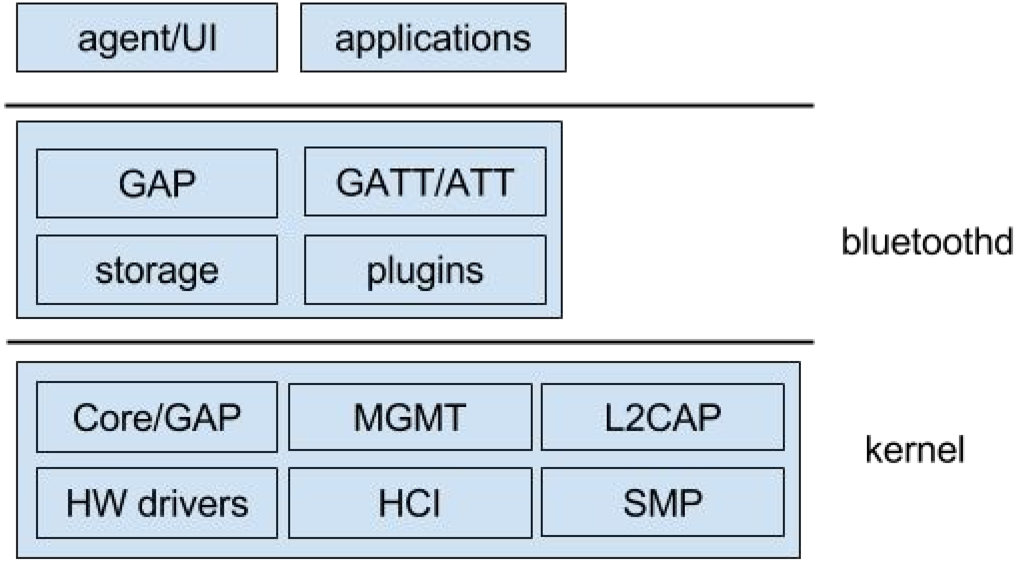
Bluetooth Low Energy

- Introduced with Bluetooth 4.0 (2010)
- Short range wireless technology (10-100 meters)
- Operates at 2.4 GHz (IMS band)
- Designed for low power usage
- Profiles (applications) use GATT
- Further improvements in 4.1 and 4.2 specifications
  - Improved security (LE Secure Connections)
  - Connection Oriented Channels

Linux Bluetooth Low Energy features

- Core Specification 4.2
- Generic Access Profile (GAP)
  - central, peripheral, observer, broadcaster
  - privacy
- Security Manager
  - Legacy Pairing, Secure Connections, Cross-transport pairing
- Generic Attribute Profile (GATT)
- L2CAP Connection Oriented Channels
- 6LoWPAN
- HID over GATT (HoG)
- Multiple adapters support
- Others

Linux Bluetooth LE Stack Architecture



agent/UI

applications

GAP

GATT/ATT

storage

plugins

bluetoothd

Core/GAP

MGMT

L2CAP

HW drivers

HCI

SMP

kernel

Linux Bluetooth LE Stack Architecture (kernel)

- Split between Linux kernel and userspace
- Kernel:
  - GAP
  - L2CAP
  - Security Manager
  - Hardware drivers
  - Provides socket based interfaces to user space
    - For data (L2CAP, HCI)
    - For control (MGMT, HCI)
  - https://git.kernel.org/cgit/linux/kernel/git/bluetooth/bluetooth-next.git/

Linux Bluetooth LE Stack Architecture (user space)

- bluetoothd
  - Central daemon
  - D-Bus interfaces for UI and other subsystems
  - Reduces exposure to low level details
  - Handle persistent storage
  - Extendible with plugins (neard, legacy GATT plugins)
- Tools
  - bluetoothctl - command line agent
  - btmon - HCI tracer
  - Set of command line tools useful for testing, development and tracing

Bluetooth Management interface

- Available since Linux 3.4
- Replaces raw HCI sockets
- Allow userspace to control kernel operations
- Provides mostly Generic Access Profile functionality (adapter settings, discovery, pairing etc)
- Required by BlueZ 5
- Specification available at doc/mgmt-api.txt in bluez.git
- http://www.bluez.org/the-management-interface/
- btmgmt tool for command line

BlueZ D-Bus API overview

- Use standard D-Bus ObjectManager and Properties interface
- Adapters and remote devices represented as objects
  - /org/bluez/hci0
  - /org/bluez/hci0/dev_00_11_22_33_44_55
- With versioned interfaces
  - org.bluez.Adapter1, org.bluez.Device1 etc
  - org.bluez.GattService1, org.bluez.GattCharacteristic1 etc
- Manager and Agent style interfaces for external components
  - org.bluez.AgentManager1, org.bluez.Agent1
- As of BlueZ 5.42 GATT D-Bus interfaces are declared stable

Basic operations (GAP)

- Adapter settings
- Device discovery
- Connection management
- Pairing

- org.bluez.Adapter1 - adapter control
- org.bluez.Device1 - device control
- org.bluez.Agent1 - UI pairing agent

Scanning - devices discovery

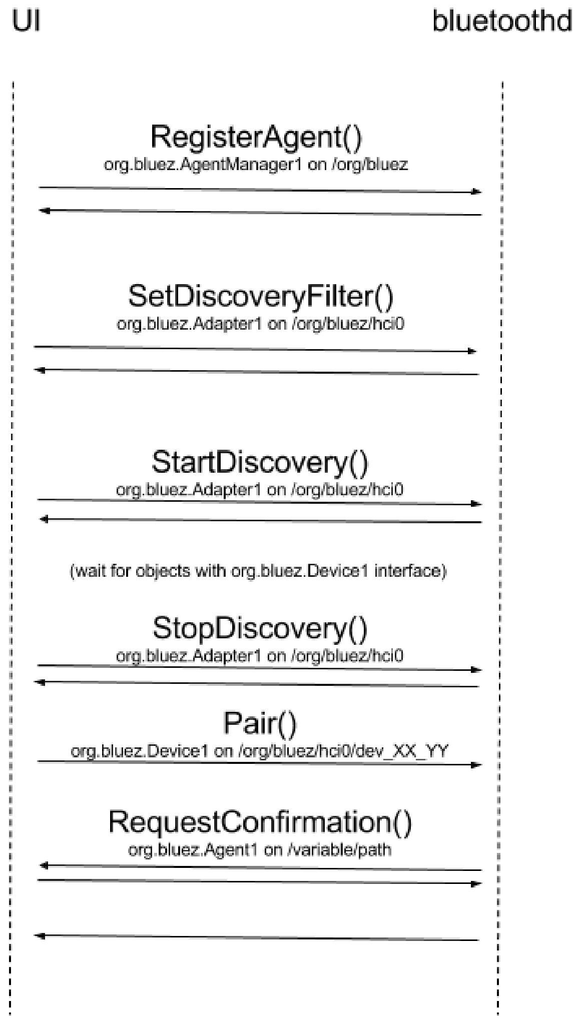
- org.bluez.Adapter1 interface
- StartDiscovery() and StopDiscovery() methods control discovery sessions
- SetDiscoveryFilter(dict filter) for discovery session tuning
    - UUID based filtering
    - RSSI or Pathloss threshold
    - Transport (type of scan)
    - Multiple clients filters are internally merged
- Objects with org.bluez.Device1 interface represent remote devices
- While devices are being discovered new objects are created (or updated)

Advertising

- Allows external applications to register Advertising Data
- Support for multiple advertising instances
- org.bluez.LEAdvertisement1
    - Implemented by external application
    - Properties define advertising type and what to include
    - AD is constructed by stack (required data types are always included)
- org.bluez.LEAdvertisingManager1 on /org/bluez/hciX
    - RegisterAdvertisement()
    - UnregisterAdvertisement()
- Currently no support for configuring Scan Responses
- doc/advertising-api.txt

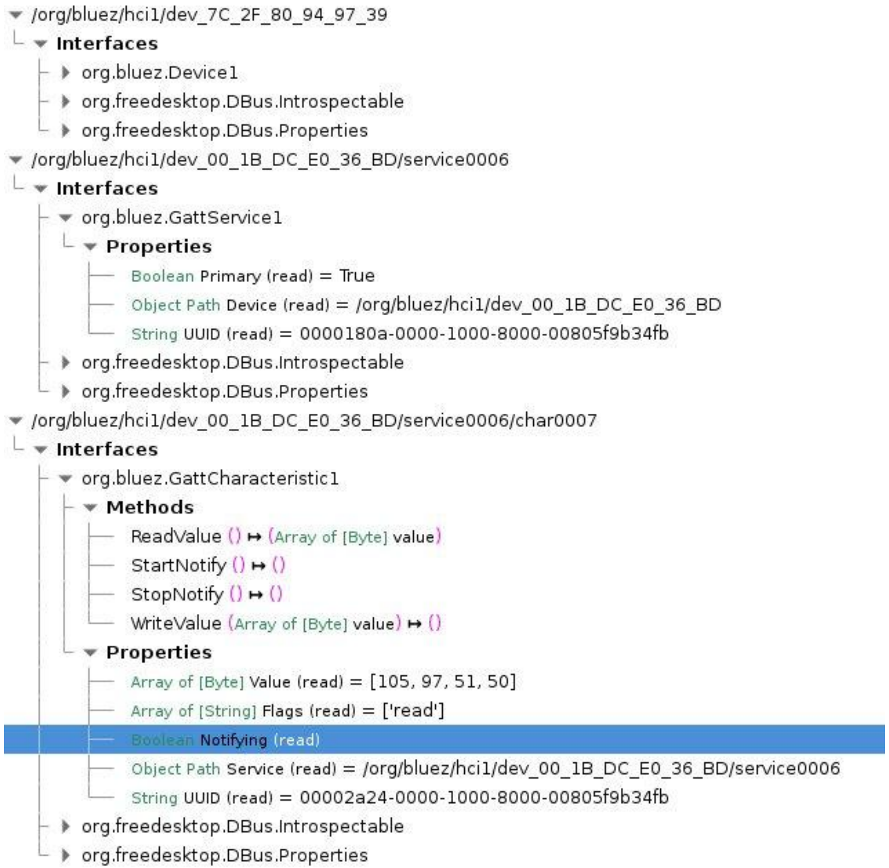
Pairing

- bluetoothd relies on agents for user interaction
  - User can be a human where agent is UI
  - But it can also be any policy implementation
- org.bluez.AgentManager1
  - RegisterAgent(object agent, string capability) - registers an agent handler with specified **local** capability
  - RequestDefaultAgent(object agent) - sets registered agent as default
- org.bluez.Agent1
  - Implemented by application
  - Called by bluetoothd when user input is needed eg. to enter or confirm passkey
- Each application can register own agent
- Default agent used for incoming requests
- or for outgoing requests if application has no agent registered


UI
bluetoothd
RegisterAgent()
org.bluez.AgentManager1 on /org/bluez
SetDiscoveryFilter()
org.bluez.Adapter1 on /org/bluez/hci0
StartDiscovery()
org.bluez.Adapter1 on /org/bluez/hci0
(wait for objects with org.bluez.Device1 interface)
StopDiscovery()
org.bluez.Adapter1 on /org/bluez/hci0
Pair()
org.bluez.Device1 on /org/bluez/hci0/dev_XX_YY
RequestConfirmation()
org.bluez.Agent1 on /variable/path

GATT

- Internal plugins (and their APIs) are deprecated
- Replaces profile specific APIs
- Stable since 5.42
- Local and remote services share same D-Bus API
    - org.bluez.GattService1
    - org.bluez.GattCharacteristic1
    - org.bluez.GattDescriptor1
- Remote hierarchy under device path
    - /org/bluez/hci0/dev_AA/serviceXX/charYYYY/descriptorZZZZ
- org.bluez.Device1.ServicesResolved=true indicates discovery has completed



```
▾ /org/bluez/hci1/dev_7C_2F_80_94_97_39
  ▾ Interfaces
    ▸ org.bluez.Device1
    ▸ org.freedesktop.DBus.Introspectable
    ▸ org.freedesktop.DBus.Properties
▾ /org/bluez/hci1/dev_00_1B_DC_E0_36_BD/service0006
  ▾ Interfaces
    ▾ org.bluez.GattService1
      ▾ Properties
          Boolean Primary (read) = True
          Object Path Device (read) = /org/bluez/hci1/dev_00_1B_DC_E0_36_BD
          String UUID (read) = 0000180a-0000-1000-8000-00805f9b34fb
    ▸ org.freedesktop.DBus.Introspectable
    ▸ org.freedesktop.DBus.Properties
▾ /org/bluez/hci1/dev_00_1B_DC_E0_36_BD/service0006/char0007
  ▾ Interfaces
    ▾ org.bluez.GattCharacteristic1
      ▾ Methods
          ReadValue () ↦ (Array of [Byte] value)
          StartNotify () ↦ ()
          StopNotify () ↦ ()
          WriteValue (Array of [Byte] value) ↦ ()
      ▾ Properties
          Array of [Byte] Value (read) = [105, 97, 51, 50]
          Array of [String] Flags (read) = ['read']
          Boolean Notifying (read)
          Object Path Service (read) = /org/bluez/hci1/dev_00_1B_DC_E0_36_BD/service0006
          String UUID (read) = 00002a24-0000-1000-8000-00805f9b34fb
    ▸ org.freedesktop.DBus.Introspectable
    ▸ org.freedesktop.DBus.Properties
```

GATT (II)

- Register local profiles and services
    - org.bluez.GattManager1
        - RegisterApplication()
        - UnRegisterApplication()
- Local profile
    - org.bluez.GattProfile1
    - Bluetoothd will add matched devices to auto-connect list
- Local service
    - Represented as objects hierarchy
        - Service is root node
        - Characteristic is child of service
        - Descriptor is child of characteristic
    - grouped under Object Manager
    - Objects should not be removed

```
-> /com/example
  |   - org.freedesktop.DBus.ObjectManager
  |
  -> /com/example/service0
  | |   - org.freedesktop.DBus.Properties
  | |   - org.bluez.GattService1
  | |
  | -> /com/example/service0/char0
  | |   - org.freedesktop.DBus.Properties
  | |   - org.bluez.GattCharacteristic1
  | |
  | -> /com/example/service0/char1
  |   |   - org.freedesktop.DBus.Properties
  |   |   - org.bluez.GattCharacteristic1
  |   |
  |   -> /com/example/service0/char1/desc0
  |     - org.freedesktop.DBus.Properties
  |     - org.bluez.GattDescriptor1
  |
  -> /com/example/service1
        |   - org.freedesktop.DBus.Properties
        |   - org.bluez.GattService1
        |
        -> /com/example/service1/char0
        - org.freedesktop.DBus.Properties
        - org.bluez.GattCharacteristic1
```

HID over GATT (host)

- Supported by bluetoothd internally - ‘hog’ plugin
- Only host support
- ‘Claims’ HID service so it won’t be visible on D-Bus
- Requires uhid support in kernel
- “Just works” experience
  - Pair mouse/keyboard
  - Service is probed and connected
  - Input device is created
  - Device is added to whitelist for reconnection

```
[15674.721290] input: BluetoothMouse3600 as /devices/virtual/misc/uhid/0005:045E:0916.0002/input/input18
[15674.721494] hid-generic 0005:045E:0916.0002: input,hidraw0: BLUETOOTH HID v1.00 Mouse [BluetoothMouse3600] on 5C:E0:C5:34:AE:1C
```

Privacy

- Allows to use Resolvable Private Address (RPA) instead of Identity (public) address
- Address appears random for non-bonded devices
- Bonded devices can resolve RPA
- Prevents tracking
- Linux supports both local privacy and remote privacy
  - When device is paired its Identity Resolving Key (IRK) is stored and used for resolving RPAs
  - Providing IRK for local adapter allows kernel to generate and use RPAs
  - RPA is time rotated
- Bluetoothd handles remote device IRK storage and loading
  - After pairing Address property on org.bluez.Device1 is updated with resolved identity address
- No support for local privacy in bluetoothd yet
  - bluetoothd will create local random IRK (per adapter) and load it to kernel
  - Patch is available on linux-bluetooth mailing list

LE Connection Oriented Channels

- Available since kernel 3.14
- Easy to use, just like any L2CAP socket
- Set address type to LE and provide PSM number
  - Unfortunately obtaining address type from D-Bus is not possible

```
struct sockaddr_l2 addr;

sk = socket(PF_BLUETOOTH, type, BTPROTO_L2CAP);

/* Bind to local address */
addr.l2_family = AF_BLUETOOTH;
addr.l2_bdaddr = LOCAL_ADDR;
addr.l2_bdaddr_type = BDADDR_LE_PUBLIC;
bind(sk, (struct sockaddr *) &addr, sizeof(addr));

/* Connect to remote */
addr.l2_bdaddr = REMOTE_ADDR;
addr.l2_psm = 0x80;
connect(sk, (struct sockaddr *) &addr, sizeof(addr))
```

6LoWPAN over BT LE

- Available since kernel 3.16
- No stable interface yet, need to use debugfs
- But simple to use
  - modprobe bluetooth_6lowpan
  - echo “1” > /sys/kernel/debug/bluetooth/6lowpan_enable
  - echo "connect 00:1B:DC:E0:36:BD 1" > /sys/kernel/debug/bluetooth/6lowpan_control
  - bt0 interface is created
  - ping6 -I bt0 fe80::21b:dcff:fee0:36bd

Custom solutions

- Don’t want/need full bluetoothd for your tiny custom app?
- src/shared folder in bluez.git contains LGPL licenced components
  - Used by bluetoothd and other BlueZ tools
  - Library like C API
  - Easy to integrate
  - MGMT, ATT, GATT, crypto, advertising, ECC, GAP and more
  - No API stability guaranteed
- Ideal for beacons or simple peripheral applications
  - peripheral/ folder for peripheral example (LGPL)
- User channel
  - Gives HCI exclusive access to user space application
  - Sample in tools/eddystone.c (GPL)

Tips

- Use D-Bus API (documentation in doc/) whenever possible
- Python D-Bus examples in test/
- bluetoothctl tool as C D-Bus sample (GPL)
- Don’t use hcitool unless you really know what you are doing
    - Use bluetoothctl or btmgmt instead
- For HCI traces use btmon instead of hcidump
- Stuck with ancient kernel?
    - Use Linux Backports project https://backports.wiki.kernel.org/
    - Example https://bluez-android.github.io/
- Extra kernel configuration via sysfs
    - /sys/class/bluetooth
- Extra kernel informations and experimental features via debugfs
    - /sys/kernel/debug/bluetooth

Tips (II)

- Bluetoothd configuration
  - /etc/bluetooth/main.conf
- Want to contribute?
  - Join #bluez on irc.freenode.net
  - linux-bluetooth@vger.kernel.org mailing list for patches
  - Read HACKING file
- Reporting a bug?
  - #bluez-users on irc.freenode.net or linux-bluetooth@vger.kernel.org list
  - Provide HCI traces
  - Enable bluetoothd debug logs (‘bluetoothd -n -d -E’ or SIGUSR2)

Future work

- Management API for BT 6LoWPAN
- Included services support for GATT D-Bus API
- Bluetooth 5 features
- LE out-of-band pairing (neard)
- Removal of gattrib code
- Improving support for dual-mode devices
    - New DeviceLE1 and DeviceBR1 interfaces (RFC)
    - Extending Adapter1 interface

Questions?

Doing Bluetooth Low Energy on Linux

Szymon Janc

szymon.janc@codecoup.pl

CODECOUP

OpenIoT Summit Europe, Berlin, 2016