Bluetooth on modern Linux

Szymon Janc
szymon.janc@codecoup.pl

Embedded Linux Conference, San Diego, 2016

Agenda

- Introduction
- Bluetooth technology recap
- Linux Bluetooth stack architecture
  - Linux kernel
  - BlueZ 5 (bluetoothd, obexd) and BlueZ for Android
  - D-Bus interfaces
  - External components integration (PulseAudio, NetworkManager etc)
- Bluetooth Low Energy support
  - D-Bus interfaces for GATT and advertising
  - 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

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

Bluetooth

- Short range wireless technology (10-100 meters)
- Operates at 2.4 GHz (IMS band)
- Profiles – definitions of possible applications
- 1.x – 1999 – many problems, including interoperability issues
- 2.0 + EDR – 2004 – Enhanced Data Rate, up to 2.1 Mbits/s
- 2.1 + EDR – 2007 – Secure Simple Pairing
- 3.0 + HS – 2009 – up to 24 Mbits/s (using WiFi)
- 4.0 – 2010 – Low Energy
- 4.1 – 2013 – Further LE improvements
- 4.2 – 2014 – LE security improvements, IoT

Linux Bluetooth features

- Core Specification 4.2 (GAP, L2CAP, RFCOMM, SDP, GATT)
  - Classic Bluetooth (BR/EDR)
  - Bluetooth Smart (Low Energy)
- Audio and media (A2DP, AVRCP)
- Telephony (HFP, HSP)
- Networking (PAN, 6LoWPAN)
- Input device (HID, HoG)
- OBEX (FTP, OPP, MAP, PBAP)
- Others

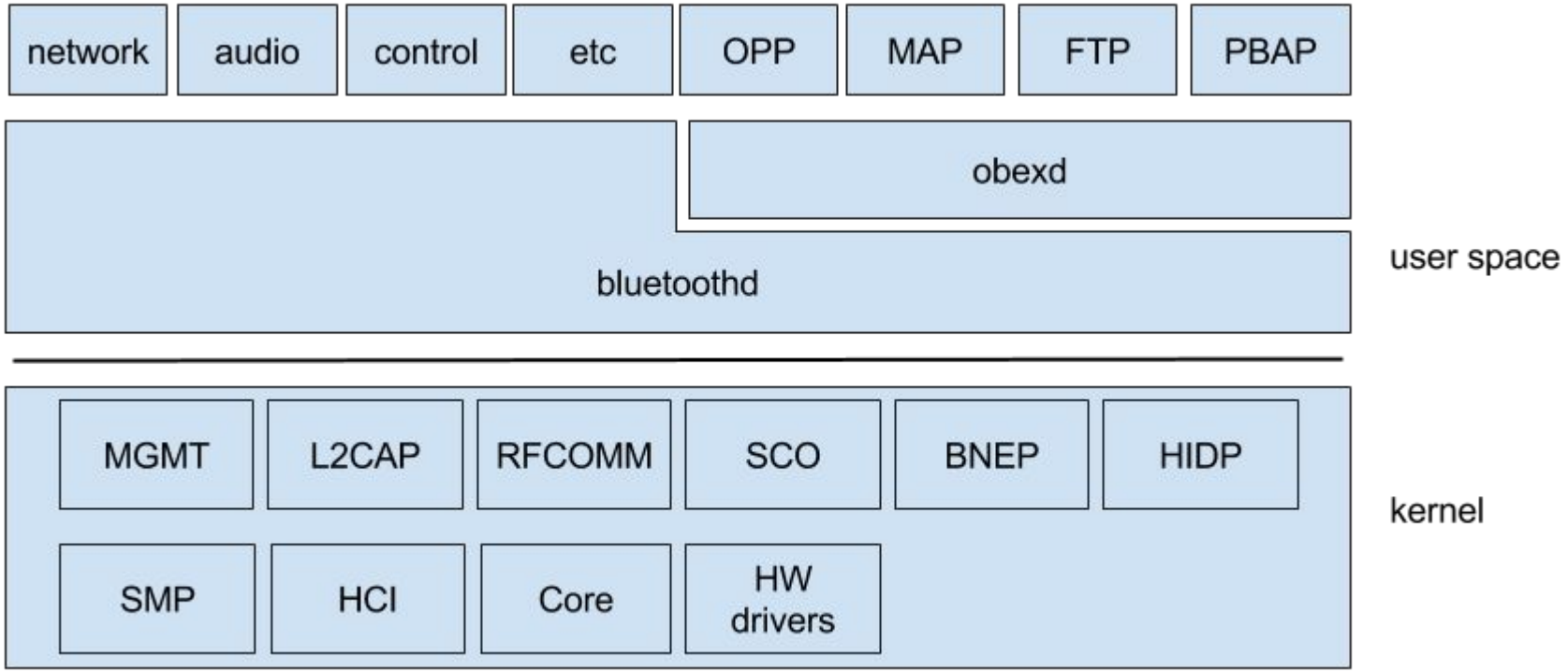
Linux Bluetooth Stack Architecture (kernel)

- Split between Linux kernel and userspace
- Kernel:
  - Low level protocols (L2CAP, RFCOMM, BNEP, HIDP, etc)
  - Security (SSP, SMP)
  - Hardware drivers
  - Provides socket based interfaces to user space
    - For data (L2CAP, RFCOMM, SCO, HCI)
    - For control (MGMT, HCI, BNEP, HIDP)
  - https://git.kernel.org/cgit/linux/kernel/git/bluetooth/bluetooth-next.git/

Linux Bluetooth Stack Architecture (user space)

- bluetoothd
  - central daemon
  - D-Bus interfaces for UI and other subsystems
  - Reduces exposure to low level details
  - Extendible with plugins (eg neard for NFC, sixaxis for DS3 support)
- obexd
  - daemon for OBEX profiles
  - D-Bus interface for UI
  - Similar architecture to bluetoothd
- Tools
  - bluetoothctl - command line agent
  - btmon - HCI tracer
  - Set of command line tools useful for testing, development and tracing

Linux Bluetooth Stack Architecture



BlueZ for Android

- Subproject in same git tree - android/ subfolder
- Separate bluetoothd daemon
- Designed as drop-in replacement for Android Bluedroid stack
  - Implements Android BT HAL API
  - No D-Bus interfaces
- Share common code with BlueZ
  - Kernel subsystem
  - common components in user space (ATT, GATT, AVRCP, AVDTP, HoG etc)
- Not to be used in GNU/Linux
- PTS qualification instructions provided (partially useful for GNU Linux)

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

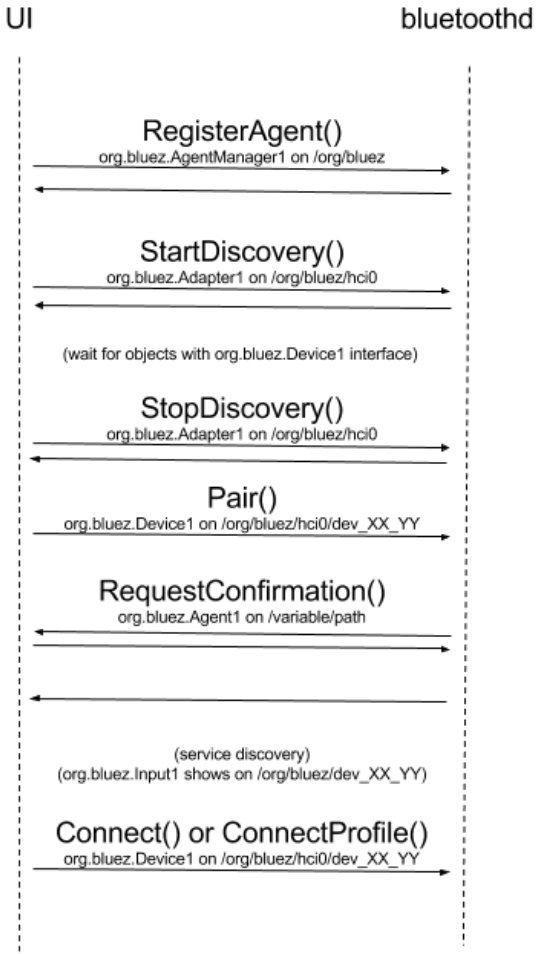
BlueZ D-Bus API overview

- 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 (supported profiles, configuration etc)
  - org.bluez.Adapter1, org.bluez.Media1 etc
  - org.bluez.Device1, org.bluez.Network1 etc
- Manager and Agent style interfaces for external components
  - org.bluez.AgentManager1, org.bluez.Agent1

Basic operations (GAP)

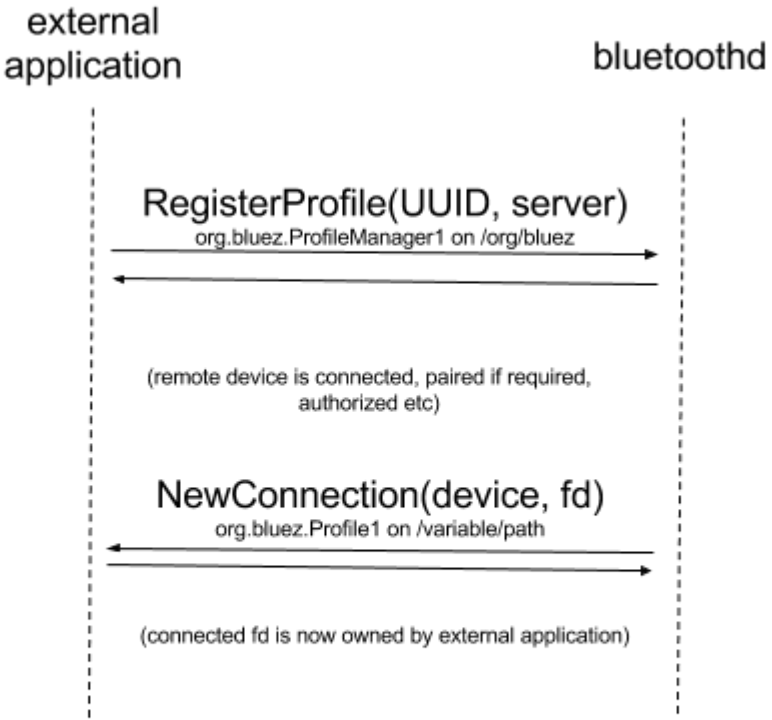
- Adapter settings
- Device discovery
- Connection management
- Pairing

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



External profiles - org.bluez.ProfileManager1

- Generic interface for implementing external profiles
- profile (a separate process) implements org.bluez.Profile1 interface
- Register object with org.bluez.ProfileManager1 interface
- Set UUID and SDP details
- Set security level, authentication, role, PSM or RFCOMM channel etc
- bluetoothd takes care of all tasks needed for connection creation
- bluetoothd will pass connection (fd and properties) to external process



Audio

- org.bluez.Media1
  - register local org.bluez.MediaEndpoint1 endpoints
- org.bluez.MediaEndpoint1
  - Allow to select and set endpoint configuration
- org.bluez.MediaTransport1
  - Represents configured stream
  - Allows to acquire FD by external application
  - Provides information like UUID, codec, volume etc.
- A2DP support in PulseAudio 5.0
- No native ALSA support (legacy IPC removed)
  - Legacy audio IPC removed
  - ALSA plugin implementing D-Bus API?

Telephony

- Implemented as external profiles
- Since PulseAudio 6.0
- Since oFono 1.13
  - together with PulseAudio (ofono backend)
- oFono is handling signaling (AT commands)
- PA is handling voice (SCO)
- Simple HSP support in PA
  - Native backend
  - No need for telephony subsystem
  - PA is handling basic AT commands
  - Suitable for desktop voice use cases (Hangouts, Skype etc)

Networking

- Support for PAN profile
  - PANU, NAP and GN roles
- Support in NetworkManager 1.0 (0.9.8.6)
- Support in ConnMan 1.11
- org.bluez.NetworkService1 for tethering
  - On /org/bluez/hciX
  - Register(uuid, bridge)
  - All connections use same bridge
- org.bluez.Network1
  - On /org/bluez/hciX/dev_YY
  - Connect(uuid)
  - Returns network interface name (eg bnep0)

obexd

- Provides similar D-Bus APIs as bluetoothd
  - org.bluez.obex service
  - Agent style API for authorization
  - Versioned interfaces
- Profiles implemented as external profiles (org.bluez.Profile1)
- D-Bus Session Bus
- Provides support for OBEX based profiles
  - File Transfer Profile (FTP)
  - Object Push Profile (OPP)
  - Phone Book Access Profile (PBAP)
  - Message Access Profile (MAP)

D-Bus Advertising (experimental)

- 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

D-Bus GATT (experimental)

- Internal plugins (and their APIs) are deprecated
- Replaces profile specific APIs
- 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

D-Bus GATT (experimental) (II)

- Register local profiles and services
  - org.bluez.GattManager1
    - {Un}RegisterProfile()
    - {Un}RegisterApplication()
- 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
```

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

```
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, HFP 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/)
- Python D-Bus examples in test/
- 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 (input.conf, network.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

- Improving support for dual-mode devices
  - New DeviceLE1 and DeviceBR1 interfaces (RFC)
  - Extending Adapter1 interface
- Management API for BT 6LoWPAN
- Deduplicating BlueZ and BfA code

Questions?

Bluetooth on modern Linux

Szymon Janc
szymon.janc@codecoup.pl

Embedded Linux Conference, San Diego, 2016