Adopting BlueZ in production

Challenges and caveats

George Kiagiadakis
george.kiagiadakis@collabora.com

February 1st 2025

FOSDEM'25

COLLABORA

$ whoami

- Principal Software Engineer @ Collabora
- Multimedia focus
- GStreamer, PipeWire
- Author & maintainer of WirePlumber
- Automotive Grade Linux contributor
- New to BlueZ !

My 2024 adventure

- Teamed up with colleague & Bluetooth expert, Frédéric Danis
- Deployed BlueZ as the Bluetooth stack of a real-world automotive In-Vehicle Infotainment (IVI) system
  - Bluetooth Special Interest Group (SIG) qualification is underway, via 3rd party
  - First in-production use of BlueZ (that we are aware of) by an automotive OEM
- Switched away from a proprietary stack over to BlueZ
  - With the goal being feature parity

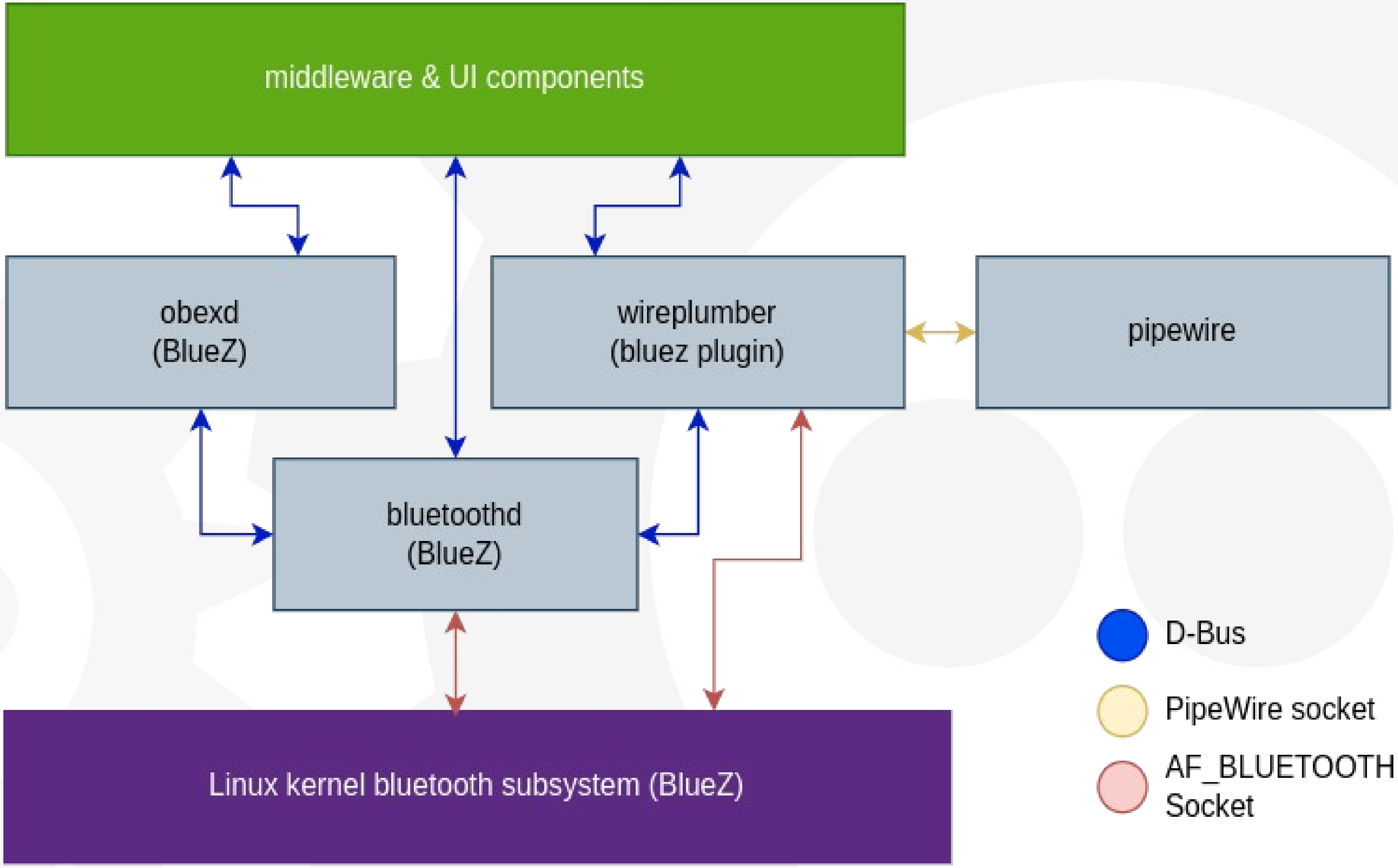
My 2024 adventure

- Improved open source components along the process
- Learned a lot about
  - Bluetooth (beyond audio)
  - Audio Digital Signal Processors (DSPs) in automotive
  - Complex audio use cases in a modern automotive
- In this talk:
  - Pass on lessons learned for other adopters

It works !

- Supporting:
  - Media player (AVRCP / A2DP / BIP)
  - Phone calls (HFP)
  - Text messaging (MAP)
  - Contact lists (PBAP)
  - Gaming controllers (HID)

Architecture



Improvements & additions

- NEW: Hands-Free Profile (HFP) telephony support in PipeWire (replaces oFono)
  - D-Bus API to control phone calls
  - https://gitlab.freedesktop.org/pipewire/pipewire/-/merge_requests/2087
- NEW: Basic Imaging Profile (BIP) subset support in BlueZ [obexd]
  - Enables album cover art for the media player
  - Merged in BlueZ 5.79
  - https://marc.info/?l=linux-bluetooth&m=172655888903683
- Fixed A/V Remote Control Profile (AVRCP) issues in BlueZ
  - https://marc.info/?l=linux-bluetooth&m=172805273430458

Challenges

Bluetooth stack differences

- BT controller exposed as a serial device
- Current: Proprietary BT software stack is a userspace daemon
  - Grabs the serial port and talks HCI with the controller
  - Nothing in kernel space
  - Allows code reuse with other operating systems
- New: BlueZ stack is divided between kernel & user space
  - Serial port attached within the kernel
  - HCI, L2CAP, SCO socket types (AF_BLUETOOTH family)
  - Similar to IP/TCP/UDP in the AF_INET family

Misaligned expectations

- “Profiles” implemented by different components
  - No Bluetooth version guarantee (host BT version, profiles, etc)
  - Must run qualification tests to assert a version (across projects)
- Not straightforward to filter HCI packets [yet?]
  - Can be a security requirement to implement a firewall
  - No support in the kernel
- No support for containers (network namespaces) [yet?]
  - Containers may be used to isolate “domains” on the same SoC
  - BlueZ daemon(s) must run on the host

Abstraction level differences

- BlueZ hides complexity, but low level info is sometimes expected
    - Customer code had been developed with access to low level info
    - Using lots of logs to monitor activities/compliance
- In some cases, not enough error granularity in BlueZ
- BlueZ maintains list of paired devices internally
    - Customer's middleware was already doing that → friction

Vendor specific HCI commands

- BT Controller and host communicate with HCI commands
  - “Host Controller Interface”
  - Vendors implement non-standardized functionality with custom commands
- Affects many profiles (HFP, A2DP, RSSI & more)
  - HFP: custom commands to configure hardware audio path
    - SCO link data is routed directly to the hw audio DSP
    - Need to configure format, sample rate, codec, etc…
    - Pure unnecessary pain, no kernel infrastructure (yet)

Security concerns of sending raw HCI commands

- Sending custom HCI commands requires CAP_NET_RAW
  - Some may also require CAP_NET_ADMIN
- Customer was not pleased
  - Kudos to them
- BlueZ daemon also requires these capabilities
  - This is part of the BlueZ design
  - BlueZ daemon is “authenticated” via CAP_NET_ADMIN to be able to execute privileged operations

Non Challenges

vCard parsing for Phone Book Access Profile

- vCard / VCF (Virtual Contact File): file describing a contact
  - Simple text-based format
- Finding a suitable parser was hard… (for use in C/C++)
  - Tried GNOME libraries, KDE libraries, random libraries on GitHub
  - Generally way too many dependencies
- Looked at Automotive Grade Linux (AGL) for inspiration
  - AGL gave the solution: don’t use any library!
- vCard parsing is **really simple!**
  - Easier to write from scratch than to package any library

Next steps

- Expose PipeWire telephony support on the desktop
  - GNOME Calls
  - Other desktop apps …. ?
- Hoping to get AGL to adopt this work and introduce Bluetooth support in their demonstrator (again) in 2025

Thank you!

We are hiring - col.la/careers