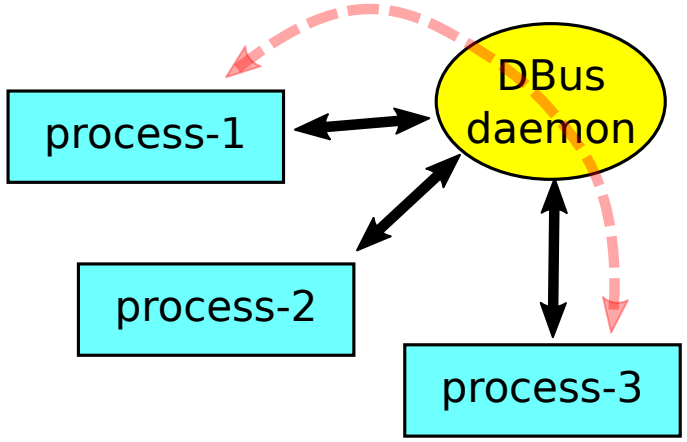
Saftlib without DBus

Michael Reese

October 30, 2018

Outline

DBus



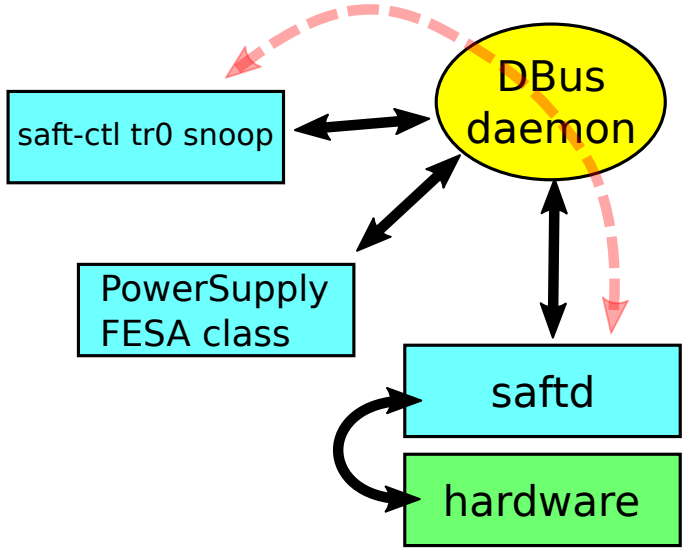
DBus provides

- typed high-level IPC between two processes
- daemon allows communication between all connected processes (function calls, signals, properties)
- low level C-API, rarely used directly

High-level APIs in various languages

- Gio: C-API, part of GTK+ support libraries
- Glibmm: C++ wrapper around Gio and Glib

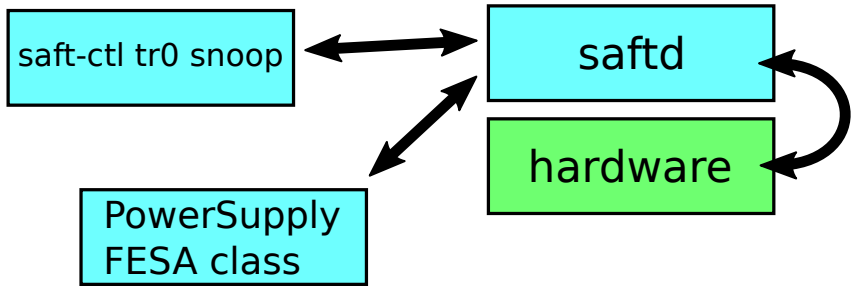
DBus use in Saftlib



Disadvantages using DBus

- Glibmm library dependency
- DBus daemon is additional process
  - higher CPU load
  - more difficult RT-scheduling (priorities?)
- DBus data transfer is relatively slow
  - latency for signals (2 hops)
  - execution time for remote function calls (4 hops)
  - encoding/decoding large amounts of data

Saftlib without DBus



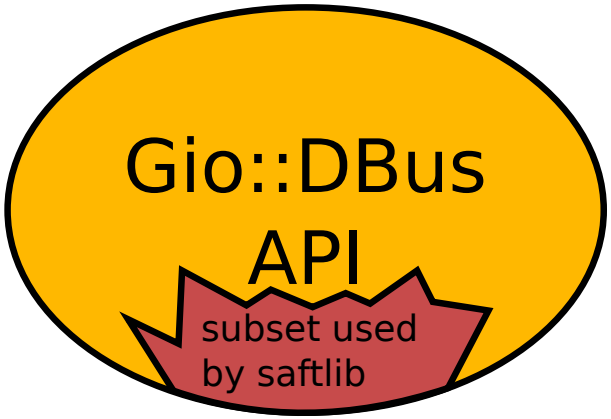
Advantages if saftd has DBus-daemon functionality

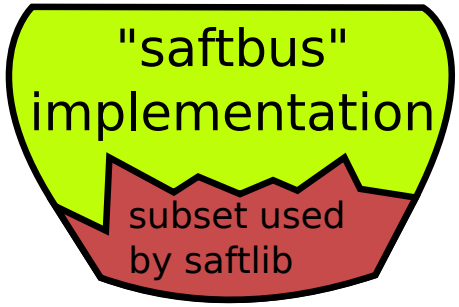
- only saftd process needed
- fewer hops for data transfer
- in the future: glibmm dependence can be dropped

Challenges / Disadvantages

- re-implement DBus functionality
- DBus tools (d-feet, busctl) cannot be used anymore

Reuse existing Gio::DBus API

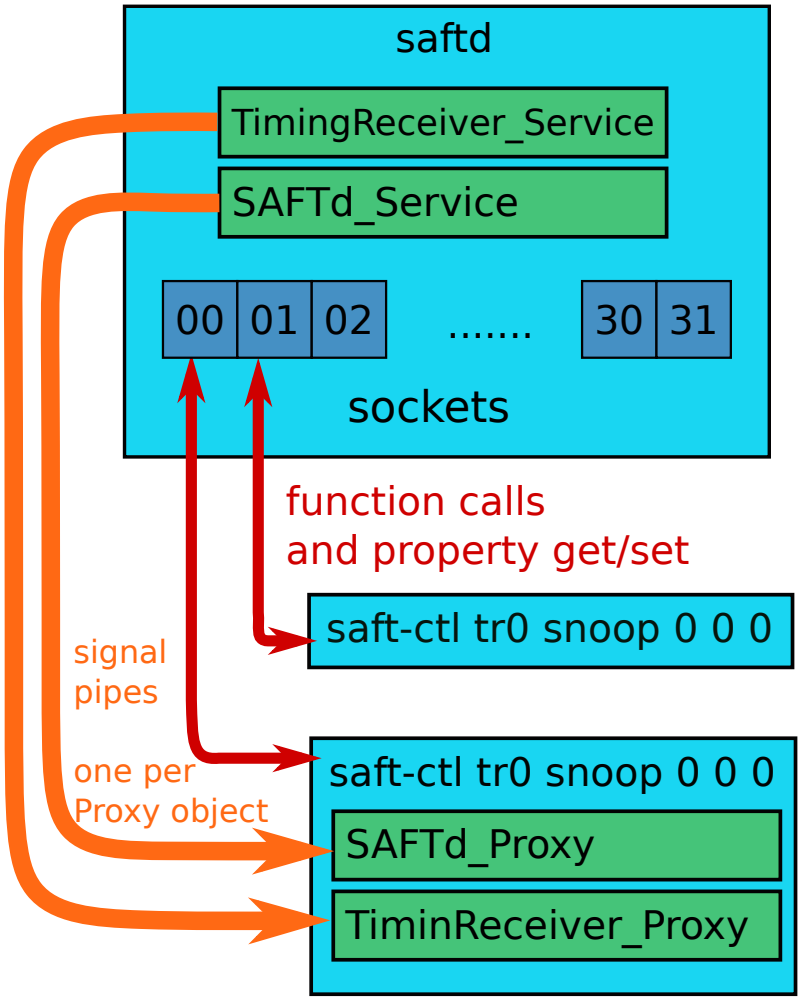




Approach

- Saftlib uses a subset of the complete Gio::DBus API
- rewrite an implementation of that part of the API and imitate its functionality (saftbus)
- additional module inside Saftlib codebase
- Saftlib code is largely unchanged

Implementation details: sockets and pipes



System resources

- saftd runs one single thread
- function calls and properties via sockets
- one socket per client process
- finite number of sockets (32)
- signals via pipes, one pipe per Proxy object

Implementation details: saftlib codebase

Saftbus in the saftlib repository

- branch in git repository: `git checkout saftbus-option`
- located in subdirectory `saftlib/saftbus`
- integrated in autotools build system of saftlib
  with its own `saftlib/saftbus/Makefile.am`
- saftbus is optional: `./configure --enable-saftbus`

Saftlib changes outside saftlib/saftbus

- add `#include <saftlib_ipc.h>`
- namesapce change `Gio::DBUS::` → `IPC_METHOD::`
  `IPC_METHOD` macro is set by ./configure script
- proxy code: `Connection` → `ProxyConnection`
  saftbus has different classes for Service and Proxy objects

No impact on user code

Saftlib API remains unchanged

- no change in user code required (e.g. fesl)
- fast arrays via pipes (`type="AAu"`) are still supported
- recompilation required
- link with `-lsaftlib -lsaftbus`
  (pkg-config saftlib –libs)

Useful (future) changes with modified API

Breaking the API allows further simplifications

- `int wait_for_signal(int timeout_ms)` blocking call
  - based on poll system call
  - simplify user code by replacing local `Glib::MainLoop`
  - potentially faster because a subset of Proxys can be selected
- get rid of `PropertyChanged` signals
  - they just eat up signal bandwidth
  - need to change device APIs (XML and driver code) to not rely on `PropertyChanged` signals
  - use properties for properties and signals for notifications
- get rid of `Glibmm dependence`
  - simplified deployment
  - simplified use