®

OpenPrinting

Common Print Dialog Backends

Till Kamppeter – OpenPrinting

September 20, 2021

The problem

- To control printing, GUI applications use **print dialogs**
- **Many different print dialogs**, usually from the GUI toolkit used (GTK, Qt, …)
- Each one has **its own implementation** to connect to CUPS, Print-to-File, and other print technologies
- Print dialog **development does not keep up** with changes, like temporary queues in CUPS, or addition of a new print technology (cloud service, ...)
  - Printing not considered very important
  - Newly introduced print technology not considered worthwhile
  - Developers do not have time
  - Long release cycles of GUI toolkit projects vs. fast pace in printing development

The idea

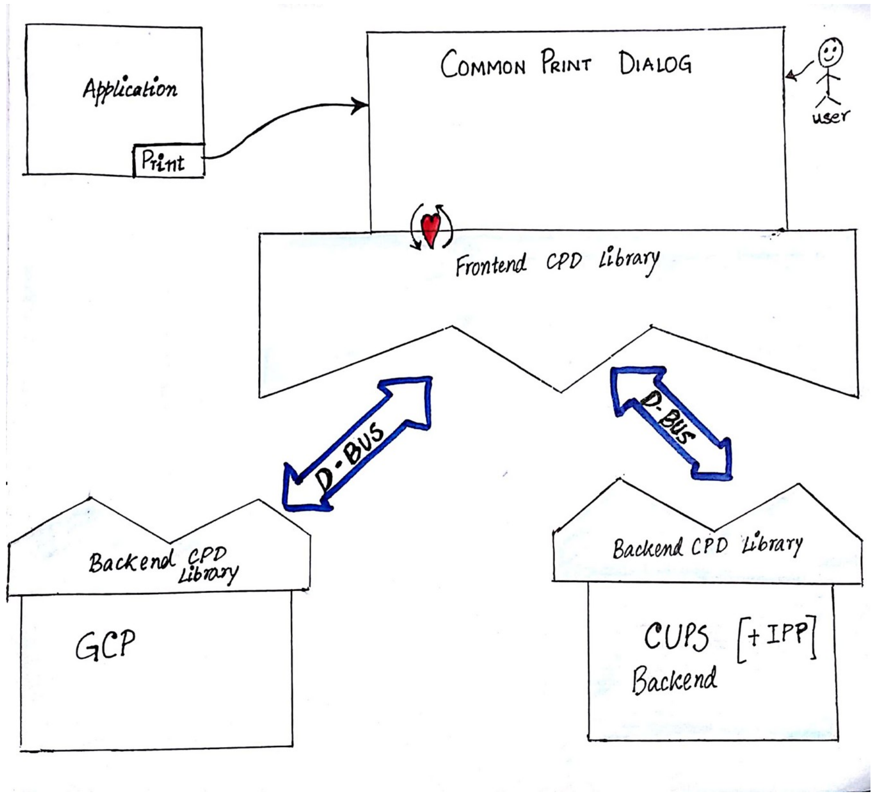
- Long time ago we tried a **Common Print Dialog**, but **failed** due to lack of human resources and/or funding
- Later we Aveek Basu remembered this project **suggested a revival**, but I was unsure.
- Fixing a CUPS-related bug in the **GTK print dialog** I discovered that it uses **backends** for different print technologies
- All this brought up the idea of **Common Print Dialog Backends** in me:
  - Dialog itself still from the GUI toolkits (GTK, Qt, LibreOffice, ...)
  - GUI-independent backends for each print technology (CUPS, Print to file, ...)
  - Connection Dialog ↔ Backend: **D-Bus** (separately sandboxable)
  - Backend and frontend libraries

The idea

- Backends maintained by **maintainer of print technology**
  - CUPS backend: OpenPrinting
  - GlobalCloud Print backend: GlobalCloud
  - …
- **Print dialog detects installed backends** and shows the printers of the respective print technologies
- **User sees always the same printers** with the same user-settable options in all print dialogs (GTK, Qt, LibreOffice, …)
- Print service provider can **supply backend via Snap Store**
- Maintainer of print technology changes something → He issues backup update and all print dialogs are up-to-date

The implementation

- I posted this a project idea in the **Google Summer of Code 2017** …
- … and **Nilanjana Lodh** picked it up and implemented it (her original drawing):



The Implementation

- Libraries are on the **OpenPrinting GitHub**
  - Frontend/Backend libraries: **`cpdb-libs`**
  - CUPS backend: **`cpdb-backend-cups`**
  - Print to file backend: **`cpdb-backend-file`**
- There are also **packages in Ubuntu** (Universe)

The new problem

- **No one** did a frontend implementation to submit to the GUI toolkit projects GTK and Qt yet.
- This could save us from problems like
  - CUPS added the new **`cupsEnumDests()`** API to **support its temporary queues** many years ago, GTK switched to it this year, **Qt did not switch yet**.
  - The architecture of CUPS will significantly change with version 3.0 ...

Questions / Comments