Proceedings of the Linux Symposium

June 27th–30th, 2007
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc.*
Dirk Hohndel, *Intel*
Martin Bligh, *Google*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
John Feeney, *Red Hat, Inc.*
Len DiMaggio, *Red Hat, Inc.*
John Poelstra, *Red Hat, Inc.*

Desktop integration of Bluetooth

Marcel Holtmann
BlueZ Project
marcel@holtmann.org

Abstract

The BlueZ project has enhanced the Bluetooth implementation for Linux over the last two years. It now seamlessly integrates with D-Bus and provides a really simple and easy to use interface for the UI applications. The current API covers all needed Bluetooth core functionalities and allows running the same daemons on all Linux distributions, the Maemo or OpenMoko frameworks, and other embedded systems. The user interface is the only difference between all these systems. This allows GNOME and KDE applications to share the same list of remote Bluetooth devices and many more common settings. As a result of this, the changes to integrate Bluetooth within the UI guidelines of Maemo or OpenMoko are really small. In return, all Maemo and OpenMoko users help by fixing bugs for the Linux desktop distributions like Fedora, Ubuntu, etc., and vice versa.

1 Introduction

The desktop integration of Bluetooth technology has always been a great challenge since the Linux kernel was extended with Bluetooth support. For a long time, most of the Bluetooth applications were command-line utilities only. With the D-Bus interface for the BlueZ protocol stack, it became possible to write desktop-independent applications. This D-Bus interface has been explicitly designed for use by desktop and embedded UI applications (see Figure 1).

For the desktop integration of Bluetooth, three main applications are needed:

- Bluetooth status applet;
- Bluetooth properties dialog;
- Bluetooth device wizard.

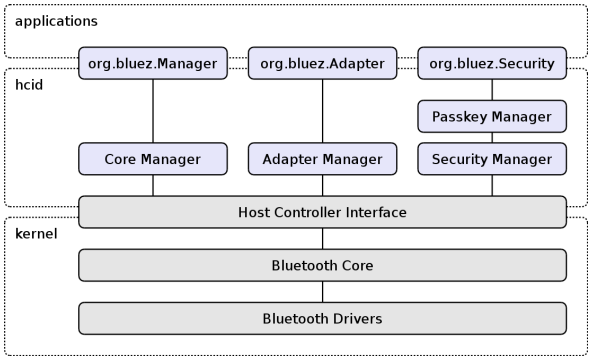

Figure 1: D-Bus API overview

2 Bluetooth applet

The Bluetooth applet is the main entry point when it comes to device configuration and handling of security-related interactions with the user, like the input of a PIN code.

One of the simple tasks of the applet is to display a Bluetooth icon that reflects the current status of the Bluetooth system such as whether a device discovery is in progress, or a connection has been established, and so on. It is up to the desktop UI design guidelines to decide if the icon itself should change or if notification messages should be displayed to inform the user of status changes.

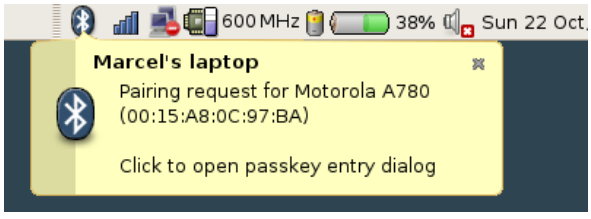

Figure 2: Bluetooth applet notification

Besides the visible icon, the applet has to implement the default passkey and authorization agent interfaces.

These two interfaces are used to communicate with the Bluetooth core daemon. The task of the applet is to display dialog boxes for requesting PIN codes or authorization question to the end user. The input will be handed back to the daemon which then actually interacts with the Bluetooth hardware.

Additionally, the applet might provide shortcuts for frequently used Bluetooth tasks. An example would be the launch of the Bluetooth configuration dialog or device setup wizard.

Figure 2 shows the notification of a pairing request for the GNOME Bluetooth applet.

3 Bluetooth properties

While the applet shows the current status of the Bluetooth system and handles the security related tasks, the properties dialog can be used to configure the local Bluetooth adapter (see Figure 3).

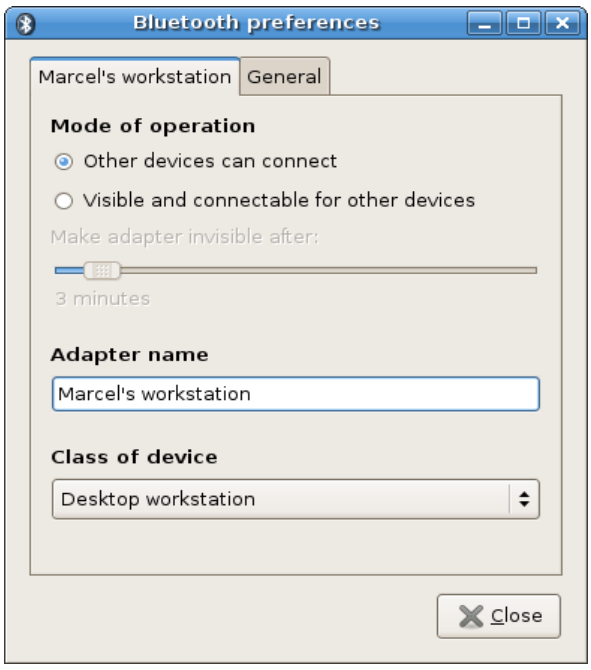

Figure 3: Bluetooth adapter configuration

The D-Bus interface restricts the possible configurable options to the local adapter name, class of device, and mode of operation. No additional options have been found useful. The Bluetooth core daemon can adapt other options automatically when needed.

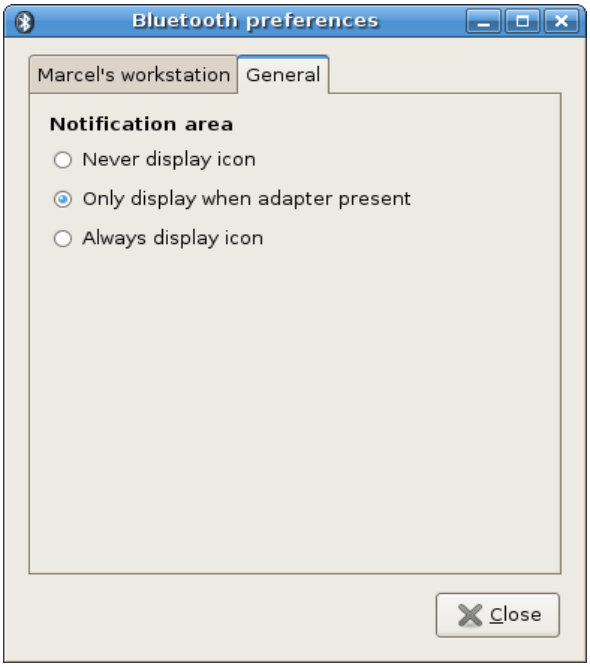

Figure 4: Bluetooth adapter configuration

In addition to the Bluetooth adapter configuration, the Bluetooth properties application can also control the behavior of the applet application (see Figure 4)—for example, the visibility of the Bluetooth status icon. It is possible to hide the icon until an interaction with the user is required.

These additional configuration options are desktop- and user-specific. The GNOME desktop might implement them differently than KDE.

4 Bluetooth wizard

With the status applet and the properties dialog, the desktop task for user interaction, notification, and the general adapter configuration are covered. The missing task is the setup of new devices. The Bluetooth wizard provides an automated process to scan for devices in range and setup any discovered devices to make them usable for the user (see Figure 5).

The wizard uses the basic Bluetooth core adapter interface to search for remote devices in range. Then, it presents the user a list of possible devices filtered by the class of device. After device selection, the wizard tries to automatically setup the services. For these tasks it uses special Bluetooth service daemons.

Currently the Bluetooth subsystem provides the following service daemons that can be used by the wizard or any other Bluetooth application:

- Network service
  - PAN support (NAP, GN and PANU)
  - LAN access (work in progress)
  - ISDN dialup (work in progress)
- Input service
  - HID support (report mode with recent kernel versions)
  - Emulated input devices (headset and proprietary protocols)
  - Wii-mote and PlayStation3 Remote
- Audio service
  - Headset and Handsfree support
  - High quality audio support (work in progress)
- Serial service
  - Emulated serial ports

Figure 5: Bluetooth device selection

5 Big picture

The BlueZ architecture has grown rapidly and the whole system became really complex. Figure 6 shows a simplified diagram of the current interactions between the Bluetooth subsystem of the Linux kernel, the Bluetooth core daemons and services, and the user interface applications.

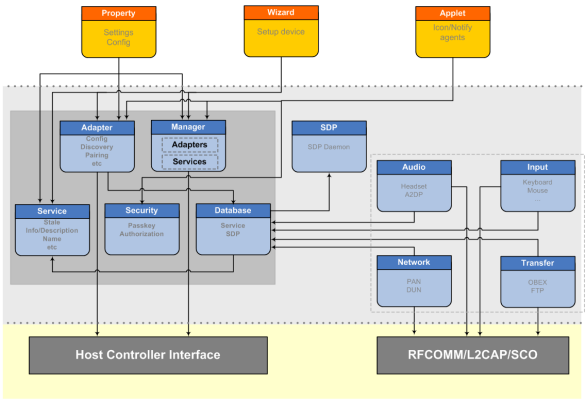

Figure 6: BlueZ architecture

All communication between daemons and a user application are done via D-Bus. Figure 7 gives an overview on how this interaction and communication via D-Bus works.

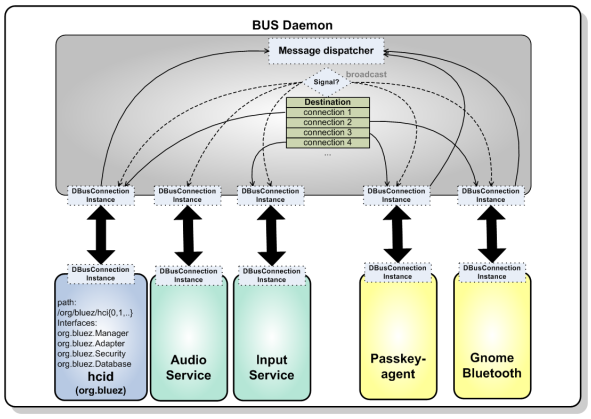

Figure 7: D-Bus communication

6 Conclusion

The *bluez-gnome* project provides an implementation for all three major Bluetooth applications needed by a modern GNOME desktop. For KDE 4, a similar set of

applications exists that uses the same D-Bus infrastructure for Bluetooth. A KDE 3 backport is currently not planned.

The desktop applications don't have to deal with any Bluetooth low-level interfaces. These are nicely abstracted through D-Bus. This allows other desktop or embedded frameworks like Maemo or OpenMoko to replace the *look and feel* quite easily (see Figure 6).

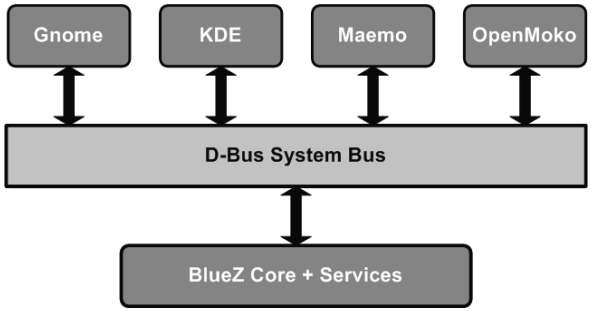

Figure 8: User interface separation

The goal of the BlueZ Project is to unify desktop and embedded Bluetooth solutions. While the user interface might be different, the actual protocol and service implementation will be the same on each system.

References

[1] Special Interest Group Bluetooth:
Bluetooth Core Specification Version 2.0 + EDR,
November 2004

[2] freedesktop.org:
D-BUS Specification Version 0.11