Proceedings of the Linux Symposium

Volume One

July 20nd–23th, 2005
Ottawa, Ontario
Canada

Conference Organizers

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

Review Committee

Gerrit Huizenga, *IBM*
Matthew Wilcox, *HP*
Dirk Hohndel, *Intel*
Val Henson, *Sun Microsystems*
Jamal Hadi Salimi, *Znyx*
Matt Domsch, *Dell*
Andrew Hutton, *Steamballoon, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*

The BlueZ towards a wireless world of penguins

Marcel Holtmann
BlueZ Project
marcel@holtmann.org

Abstract

The Bluetooth wireless technology is getting more and more attention. There are a lot of devices available and most of them are working perfect with Linux, because Linux has the BlueZ. This is the codename of the official Bluetooth protocol stack for Linux. It is possible to use Bluetooth for simple cable free serial connections, dialup networks, TCP/IP networks, ISDN networks, human interface devices, printing, imaging, file transfers, contact and calendar synchronization etc. All these services are designed to integrate seamlessly into existing and established parts of Linux, like the kernel TTY layer, the network subsystem, the CUPS printing architecture, the OpenOBEX library and so on.

1 Introduction

The Bluetooth technology was announced in May 1998 with the goal to create an easy usable cable replacement. Therefor it uses radio transmission within the 2.4 GHz ISM band to connect mobile devices like mobile phones, handhelds, notebooks, printer etc. from different manufactures without any cables. But Bluetooth is more than a simple cable replacement technology and with more and more devices using Bluetooth we see scenarios that are now making perfect sense. Examples for this are the communication of mobile phones with handhelds for exchanging contact and calendar information. Also the wireless printing of pictures without the interaction of a desktop computer.

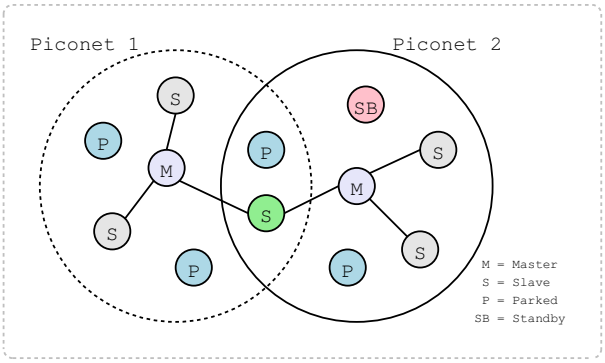

Figure 1: Bluetooth topology

Many of these applications are also possible with other technologies like IrDA or IEEE 802.11 (WiFi), but Bluetooth make the use a lot easier and defines clear application profiles.

The first steps into supporting Bluetooth with Linux are done by Axis Communications and they released their OpenBT Bluetooth Stack in April 1999. Also IBM released its BlueDrekar which was only available as binary modules. The problem of both stacks was that they are character device driven, but the Bluetooth technology is for connecting devices. So it is bet-

ter to intergrate it into the Linux network layer and to use the socket interface as primary API. On May 3, 2001, the Bluetooth protocol stack called BlueZ which was written by Qualcomm was released under GPL. This new stack followed the socket based approach. One month later it was picked up by Linus Torvalds and integrated into the Linux 2.4.6-pre2 kernel. Another Bluetooth stack for Linux was released by Nokia Research Center in Helsinki and it is called Affix. The open source community already decided to support BlueZ as official Bluetooth protocol stack Linux and it became one of the best implementations of the Bluetooth specification.

2 Bluetooth architecture

The Bluetooth architecture separates between three different core layers; hardware, host stack and applications. The hardware consists of radio, baseband and the link manager and this will be found in Bluetooth chips, dongles and notebooks. The control of the hardware is done via the host controller interface (HCI) and for the communication between the host stack and the Bluetooth hardware a hardware specific host transport driver is used. For the USB and UART transports it is possible to use general drivers, because these host transport are part of the Bluetooth specification. For PCMCIA or SDIO vendor specific driver are needed.

BlueZ implements the host stack and also the applications. The lowest layer is the logical link control and adaptation protocol (L2CAP). This protocol uses segmentation and reassembly (SAR) and protocol and service multiplexing (PSM) to abstract from the Bluetooth packet types and low-level connection links. Starting with Bluetooth 1.2 this protocol layers was extended with retransmission and flow control (RFC).

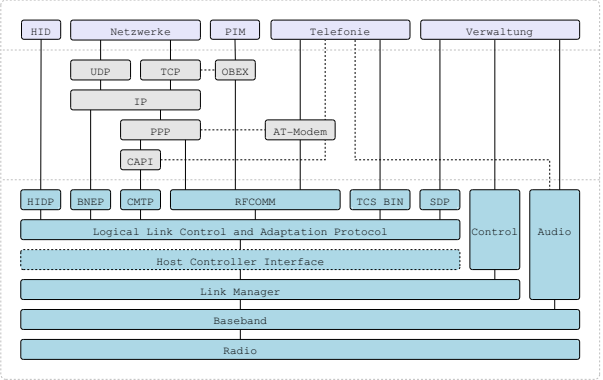

Figure 2: Bluetooth architecture

All protocol layers above L2CAP are presenting the interaction with the applications. In general it is possible to use L2CAP directly (at least with Linux), but this is not specified within the Bluetooth specification. The real Bluetooth parts of the host stack are only the L2CAP layer and the adaption layer which integrates it into other subsystems or protocol suites, like TCP/IP and OBEX.

3 Design of BlueZ

The main components of BlueZ are integrated into the Linux kernel as part of the network subsystem. It provides its own protocol family and uses the socket interface. This basic design makes it easy for application to adapt the Bluetooth technology and the integration is simple and straight forward. The use of different Bluetooth hardware is handled by the hardware abstraction inside the kernel. The BlueZ core supports the usage of 16 Bluetooth adapters at the same time. The list of supported devices is growing every day and currently over 300 different working adapters are known.

Besides the BlueZ core and the hardware abstraction also the L2CAP layer is running inside the kernel. It provides a socket interface

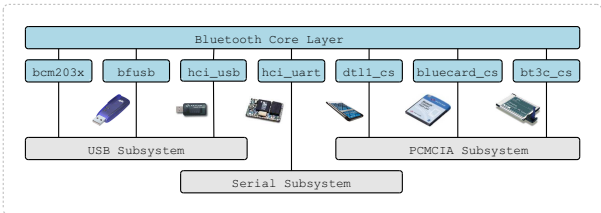

Figure 3: BlueZ core

with sequential packet characteristics and in future the RFC feature will add the stream interface.

With RFCOMM it is possible to emulate terminal devices. It is called cable replacement for legacy applications. With BlueZ it is possible to access this protocol layer from two levels. One is again a socket interface and this time with stream characteristics and the second is through the Linux TTY layer. RFCOMM emulates a full serial port and it is possible to use the point-to-point protocol (PPP) over it to create dialup or network connections.

The Bluetooth network encapsulation protocol (BNEP), the CAPI message transport protocol (CMTP) and the human interface device protocol (HIDP) are transport protocols for the network layer, the CAPI subsystem and the HID driver. Their main job is to shrink the protocol overhead and keep the latency low.

All of these protocols are implemented inside the kernel. Other Bluetooth protocols are implemented as libraries, like the service discovery protocol (SDP) or the object exchange protocol (OBEX). Some of them are also directly integrated into applications. For example the hardcopy cable replacement protocol (HCRP) and the audio video distribution transport protocol (AVDTP).

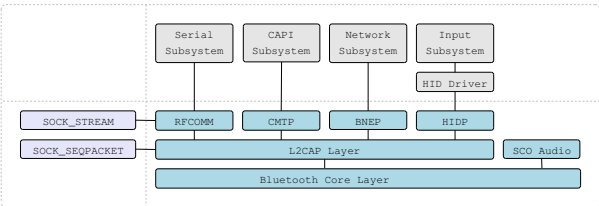

Figure 4: BlueZ protocols

Almost every Linux distribution contains a Bluetooth enabled kernel and a decent version of the BlueZ library and the BlueZ utilities.

4 Bluetooth configuration

After plugging in a Bluetooth USB dongle or inserting a Bluetooth PCMCIA card a call to `hciconfig` will show the new device.

This device is still unconfigured and like a network card it needs to be activated first. This can be done via `hciconfig hci0 up` or in the background by `hcid`.

The detailed output shows the Bluetooth device address (BD_ADDR) and additional information like name, class of device and manufacturer specific details. With `hciconfig` all of these settings can be changed.

Now it is possible to scan for other Bluetooth devices in range. For this and some other actions `hcitool` is used.

```
# hciconfig -a
hci0: Type: USB
      BD Address: 00:02:5B:01:66:F5 ACL MTU: 384:8 SCO MTU: 64:8
      UP RUNNING PSCAN ISCAN
      RX bytes:3217853 acl:79756 sco:0 events:199989 errors:0
      TX bytes:77188889 acl:294284 sco:0 commands:206 errors:0
      Features: 0xff 0xff 0x8f 0xfe 0x9b 0xf9 0x00 0x80
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF PARK
      Link mode: SLAVE ACCEPT
      Name: 'Casira BlueCore4 module'
      Class: 0x3e0100
      Service Classes: Networking, Rendering, Capturing
      Device Class: Computer, Laptop
      HCI Ver: 2.0 (0x3) HCI Rev: 0x77b LMP Ver: 2.0 (0x3)
        LMP Subver: 0x77b
      Manufacturer: Cambridge Silicon Radio (10)
```

Figure 5: Local Bluetooth adapter

```
# hcitool scan
Scanning ...
  00:80:37:25:55:96 Pico Plug
  00:E0:03:04:6D:36 Nokia 6210
  00:90:02:63:E0:83 Bluetooth Printer
  00:06:C6:C4:08:27 Anycom LAP  00:06:C6:C4:08:27
  00:04:0E:21:06:FD Bluetooth ISDN Access Point
  00:0A:95:98:37:18 Apple Wireless Keyboard
  00:A0:57:AD:22:0F ELSA Vianect Blue ISDN
  00:80:37:06:78:92 Ericsson T39m
  00:01:EC:3A:45:86 HBH-10
```

Figure 6: Scanning for Bluetooth devices

After the scan the program `sdptool` can be used to retrieve the available services of a remote Bluetooth device. The service list identifies the supported Bluetooth profiles and reveals protocol specific information that are used by other tools. The example shows a dialup networking service and a fax service which both are using the RFCOMM channel 1.

Some tools have integrated SDP browsing support and will determine the needed service information by themself. Others don't have this capability, because it is not always useful. Every SDP request involves the creation of a piconet and this can fail or timeout. So for all Bluetooth tools running at boot time this is not a desired behavior.

With the Bluetooth device address and the channel number it is possible to setup a RFCOMM TTY terminal connection for using AT commands or PPP for Internet access. The command `rfcomm bind 0 00:E0:03:04:6D:36 1` creates the device `/dev/rfcomm0` which is connected to the RFCOMM channel 1 on the mobile phone with the Bluetooth address `00:E0:03:04:6D:36`. The connection itself is not created by this command. It will first established when an application, like `pppd`, opens this device node and terminated when the last user closes it.

```
# sdptool browse 00:E0:03:04:6D:36
Browsing 00:E0:03:04:6D:36 ...
Service Name: Dial-up networking
Service RecHandle: 0x10000
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100

Service Name: Fax
Service RecHandle: 0x10001
Service Class ID List:
  "Fax" (0x1111)
  "Generic Telephony" (0x1204)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Profile Descriptor List:
  "Fax" (0x1111)
    Version: 0x0100
```

Figure 7: Requesting service information

5 Bluetooth networks

For creating network connection over Bluetooth the preferable method is using a personal area network (PAN) with BNEP. The old method was called LAN access using PPP and it used PPP over RFCOMM. This was a bad decision for the performance and now this profile is deprecated. A PAN connection can be created with the command `pand --connect 00:06:C6:C4:08:27` and after a successful connect `ifconfig` will show a `bnep0` device with the same MAC address as the BD_ADDR of the remote device.

This network device is a virtual network card,

```
# ifconfig -a
bnep0 Link encap:Ethernet  HWaddr 00:06:C6:C4:08:27
      BROADCAST MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:4 (4.0 b) TX bytes:0 (0.0 b)
```

Figure 8: Bluetooth network device

but not limited in its functionality. It is possible to use all Ethernet related commands on it and besides IPv4 and IPv6 it can also used inside an IPX network. Even methods like bridging or network address translation (NAT) are working without any problems.

Another possibility to create network connections is using ISDN and the CAPI subsystem. The Bluetooth part is called common ISDN profile (CIP) and it uses CMTP. Once a connection to an ISDN access point with `ciptool connect 00:04:0E:21:06:FD` has been created, a virtual ISDN card will be presented by the CAPI subsystem and the standard tools can be used.

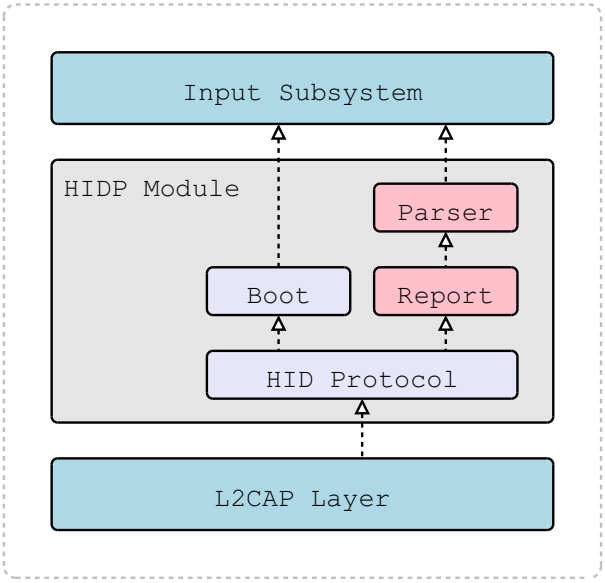

Figure 9: HID architecture

6 Printing over Bluetooth

For accessing printers over Bluetooth it is possible to do this via RFCOMM or HCRP. The Bluetooth CUPS backend supports both methods and is able to choose the best one by itself. The setup of a Bluetooth printer is very easy. The only thing that it is needed is an URI and this is created from its BD_ADDR by removing the colons. For accessing a printer with the Bluetooth device address `00:90:02:63:E0:83` the URI `bluetooth://00900263E083/` should be given to CUPS.

7 Bluetooth input devices

With the human interface device specification for Bluetooth it is also possible to use wireless mice and keyboards. Since HID devices can disconnect and reconnect at any time it is necessary to run `hidd --server` to handle such events. To bind a mouse or keyboard to your system it is only needed to contact it once. This initial connection can be done with the command `hidd --connect 00:0A:95:98:37:18`. All further connects are initiated by the device.

8 Bluetooth audio

The Bluetooth technology can be used for data communication, but it also support audio connections. For example headsets for voice connection and headphones for high-quality stereo transmission. For both device types an integration into the ALSA sound system is planned. Like all other subsystem or library integrations done by BlueZ so far, this will be almost invisible for the end user. First beta versions of the ALSA plugins exists by now and a final version is expected very soon.

9 Other applications

Many more applications started to integrate native Bluetooth support. The popular examples are the contact and calendar synchronization program MultiSync and the Gnokii tool for accessing Nokia mobile phones. Both programs can also use IrDA or cable connections as transport and Bluetooth is only another access method. In most programs the lines of Bluetooth specific code are very small, but the mobility increases a lot.

10 Conclusion

Since 2001 a lot of things have been improved and the current Bluetooth subsystem is ready for every day usage. But the development is not finished and the end user experience can be still be improved. The GNOME Bluetooth subsystem and the KDE Bluetooth framework are two projects to integrate Bluetooth into the desktop.

With Bluetooth the need of cables is decreasing and BlueZ tries to paint the Linux world blue.

References

[1] Special Interest Group Bluetooth: *Bluetooth Core Specification v1.2*, 2003.

[2] Special Interest Group Bluetooth: *Bluetooth Network Encapsulation Protocol Specification*, 2003.

[3] Special Interest Group Bluetooth: *Common ISDN Access Profile*, 2002.

[4] Special Interest Group Bluetooth: *Human Interface Device Profile*, 2003.

[5] Infrared Data Association (IrDA): *Object Exchange Protocol OBEX, Version 1.3*, 2003.