

# Einblicke in Bluetooth unter Linux

7. Juni 2004

## Note légale

Dieser Beitrag ist lizenziert unter der UVM Lizenz für die freie Nutzung unveränderter Inhalte.

## Note légale

Dieser Beitrag ist lizenziert unter der GNU General Public License.

## Zusammenfassung

Das Jahr 2004 wird als das Jahr für den endgültigen Durchbruch der Bluetooth Technologie propagiert. Unter Linux hat Bluetooth aber schon mit dem Release des Kernel 2.4.6 am 3. Juli 2001 seinen Platz gefunden und wurde in den letzten zwei Jahren aktiv weiter entwickelt. Das aktuelle Linux Bluetooth Subsystem hat sich in der Praxis bewährt und schlägt mittlerweile die Brücke zu vielen anderen Subsystemen. In den meisten Fällen ist die Bluetooth Technologie hier nativ und transparent für den Endbenutzer integriert. Gute Beispiele hierfür sind das Linux CAPI, das Drucksystem CUPS oder die OpenOBEX Library. Aber auch Applikationen wie Gnokii oder MultiSync zeigen wie einfach es ist Bluetooth unter Linux zu benutzen. Dieser Vortrag soll einen Überblick über den Bluetooth Protokoll Stack geben und zeigen wie die Bluetooth Architekture unter Linux umgesetzt wurde. Neben den derzeit unterstützten Protokollen und Profilen werden aber auch auf die Schnittstellen für Programmierer und Anwender vorgestellt, die das Linux Bluetooth Subsystem so flexibel und einfach machen. Und natürlich darf auch kein Ausblick in die Zukunft fehlen, denn das BlueZ Projekt ruht sich nicht auf den derzeitigen Lorbeeren aus. Die Unterstützung von Bluetooth Eingabegeräten, wie Mäuse und Tastaturen, hat in den letzten zwei Monaten große Fortschritte gemacht und alle derzeit am Markt erhältlichen HID Geräte werden unterstützt. Aber auch im Bereich der Audio- und Videoübertragung über Bluetooth gibt es viel "Bewegung" mit dem Bluetooth 1.2 Standard kommen auch noch einige Neuerungen dazu, die teilweise jetzt schon von Linux unterstützt werden.

## 1 Zusammenfassung

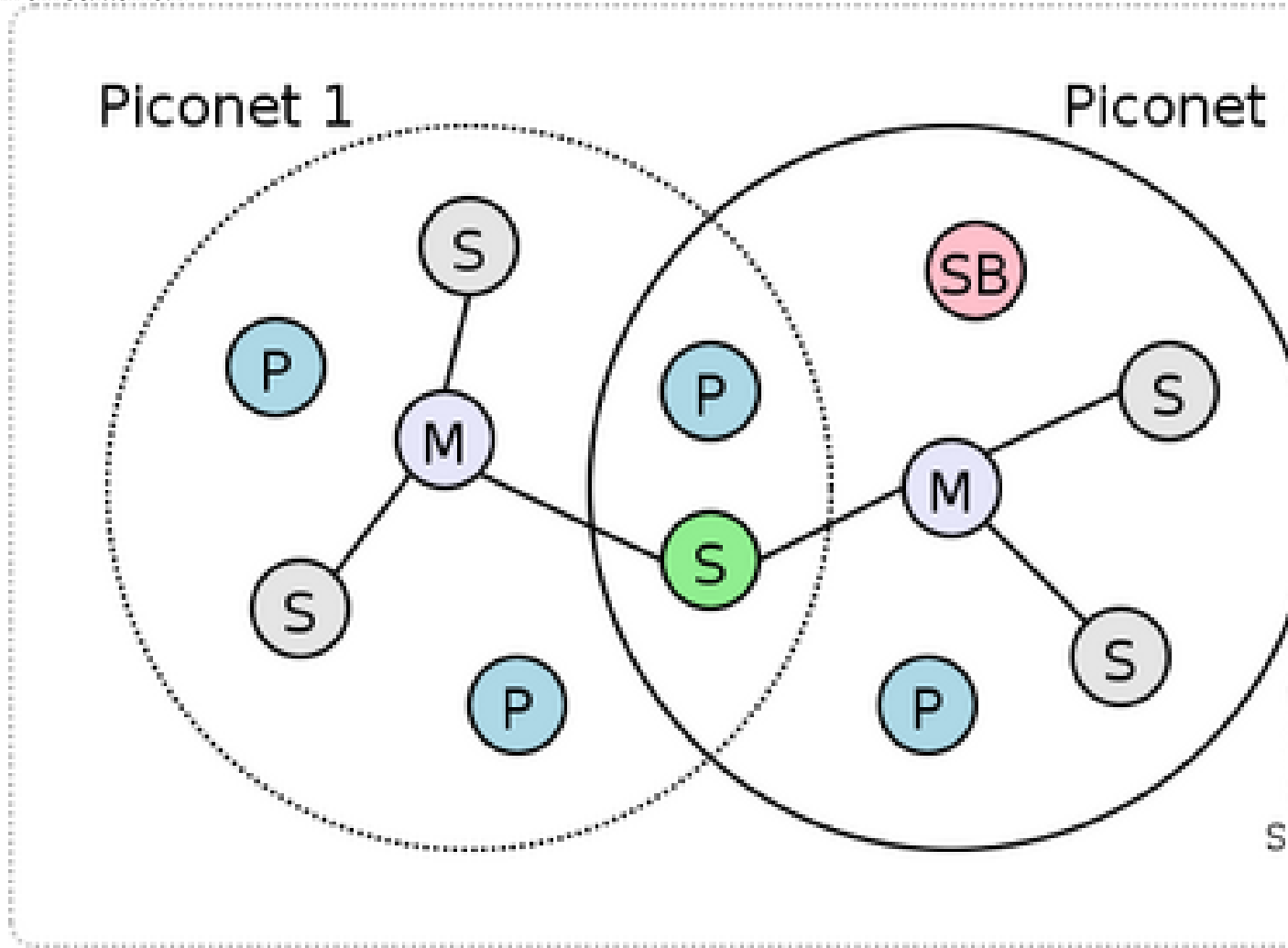
Das Jahr 2004 wird als das Jahr für den endgültigen Durchbruch der Bluetooth-Technologie propagiert. Unter Linux hat Bluetooth aber schon mit dem Release des Kernel 2.4.6 am 3. Juli 2001 Einzug gefunden und wurde in den letzten drei Jahren kontinuierlich weiter entwickelt. Das aktuelle Linux Bluetooth Subsystem hat sich in der Praxis bewährt und schlägt mittlerweile Brücken zu vielen anderen Teilbereichen von Linux. In den meisten Fällen ist Bluetooth hier nativ und transparent für den Endbenutzer integriert. Gute Beispiele sind das Linux CAPI, das Drucksystem CUPS oder die OpenOBEX Library. Aber auch Applikationen wie Gnokii oder MultiSync zeigen wie einfach es ist Bluetooth unter Linux zu benutzen.

## 2 Einleitung

Die Bluetooth-Technologie wurde im Mai 1998 angekündigt mit dem Ziel eine benutzerfreundliche Alternative für bestehenden Kabelverbindungen zu schaffen. Hierzu bedient sich Bluetooth einer universellen Funkchnittstelle im 2,4 GHz ISM Band um portable Geräte wie Mobiltelefone, PDAs, Notebooks, Drucker etc. von unterschiedlichen Herstellern ohne Kabel miteinander zu verbinden. Doch Bluetooth ist mehr als ein reiner Kabelersatz. Mit zunehmender Verbreitung kommen immer weitere Applikationen hinzu, deren Anwendung erst jetzt wirklich sinnvoll erscheinen. Dies ist zum Beispiel die Kommunikation zwischen Handy

und PDA zum Surfen oder zum Austausch von Kontakten und Terminen (PIM - Personal Information Management). Auch das direkte Übertragen von Bildern an einen Drucker, ohne den Umweg über den PC, sowie die Möglichkeit Bluetooth für drahtlose LANs zu verwenden stellen weitere Anwendungen dar.

Viele Einsatzgebiete von Bluetooth sind auch schon mit anderen Technologien, wie z.B. Infrarot, möglich. Jedoch hat die IrDA Schnittstelle eine geringere Reichweite und es ist immer eine direkte Sichtverbindung zwischen zwei Geräten notwendig. Hierdurch ergeben sich reine Punk-zu-Punkt Verbindungen und meistens ist die Kommunikation auf zwei Partner zur selben Zeit beschränkt. Bluetooth hingegen kann bis zu acht aktive Geräte in einem Piconetz verwalten und erfordert keinen Sichtkontakt zwischen den einzelnen Kommunikationspartnern. Moderne Bluetooth Chips erlauben es sogar das ein Gerät Teilnehmer in zwei Piconetzen ist und somit als Bridge zwischen diesen beiden agiert. Diese Verbindung von zwei oder mehreren Piconetzen nennt man ein Scatternetz.



Die ersten Schritte in Richtung Bluetooth-Support für Linux wurden schon sehr früh unternommen. Der OpenBT Bluetooth Stack von der Firma Axis Communications wurde im April 1999 als Open Source veröffentlicht. Danach folgte der BlueDrekar von IBM, wobei dieser nur in Form von Binary Only Modulen zur Verfügung steht. Das Problem der beiden Stacks war, daß sie zu sehr *Character Devices* fixiert sind. Da Bluetooth eine Technologie zur Vernetzung von Geräten bereit stellt, bietet sich aber eine Integration in den Network Layer von Linux an. Am 3. Mai 2001 wurde der BlueZ Bluetooth Stack ( [www.bluez.org](http://www.bluez.org) <<http://www.bluez.org/>> ) von der Firma Qualcomm unter der GPL veröffentlicht. Federführung für die Entwicklung war Maxim Krasnyansky und mit dem BlueZ gab es den ersten Linux Bluetooth Stack, der sich nahtlos in den Linux Network Layer einfügte. Das Design war so überzeugend, das Linus Torvalds nur einen Monat später, und zwar am 8. Juni 2001, den BlueZ als offiziellen Linux Bluetooth Stack in den Linux Kernel 2.4.6-pre2 aufnahm. Wie jedes andere Subsystem unter Linux wurde im Laufe der Zeit auch BlueZ immer weiterentwickelt, verbessert und in andere Subsysteme integriert. Die vierte frei verfügbare Bluetooth Implementierung für Linux ist der Affix

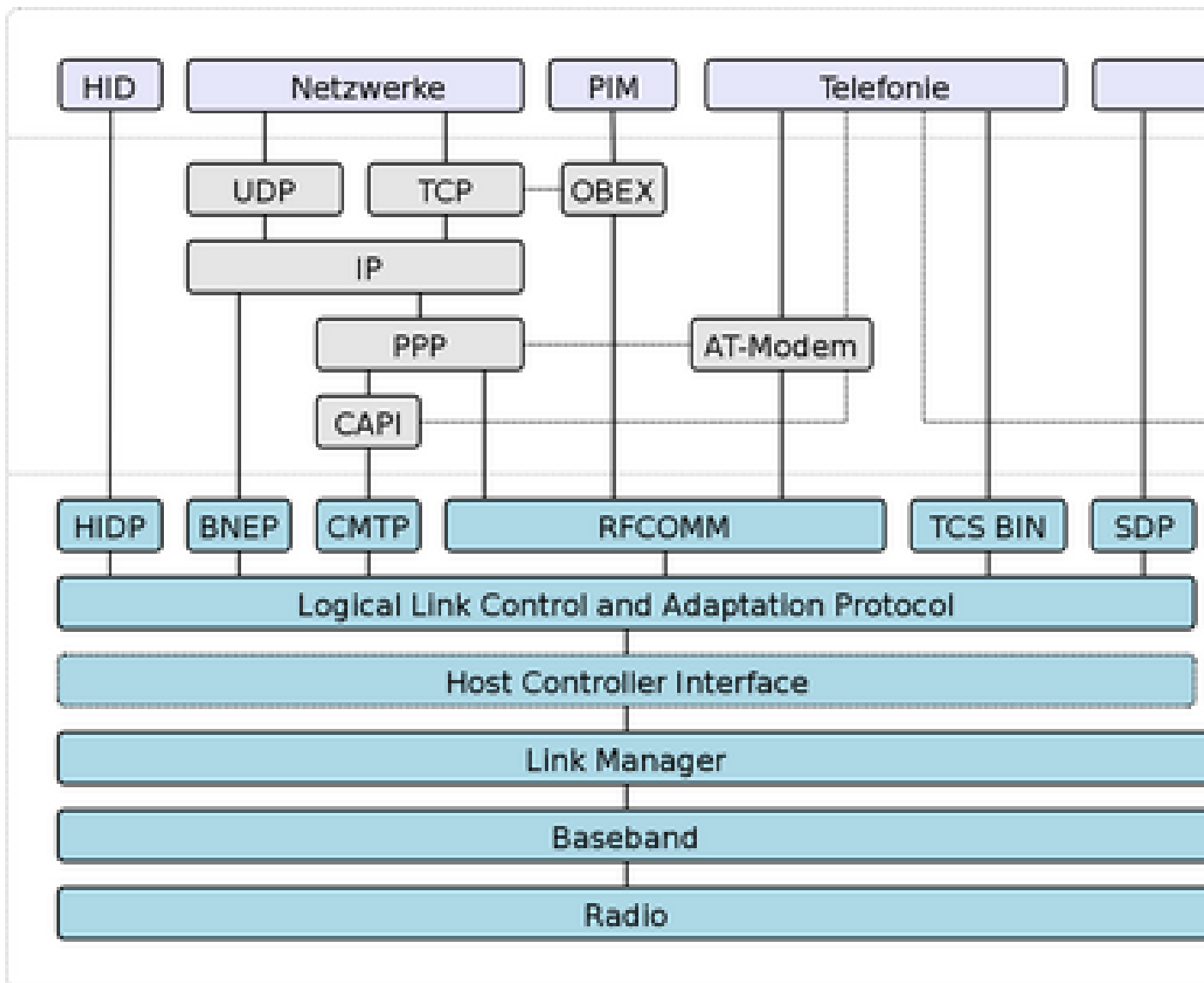
Stack vom Nokia Research Center in Helsinki. Dieser Stack erschien gut 6 Monate nach der Veröffentlichung von BlueZ und hatte keine Chance mehr den etablierten Bluetooth Stack aus dem Linux Kernel zu ersetzen. Durch die große Unterstützung der Community ist der BlueZ mittlerweile zu einem der stabilsten und besten Implementierungen der Bluetooth Spezifikation geworden.

### 3 Die Bluetooth Architektur

Beim Aufbau von Bluetooth kann man zwischen der Hardware, dem Host Stack und der Applikation unterscheiden. Die Hardware umfasst hierbei das Bluetooth Radio, Baseband und die Link Manager Implementierung. Zur Steuerung wird das *Host Controller Interface* (HCI) benutzt, welches auch die Schnittstelle zwischen Hardware und Host Stack definiert. Dazwischen liegt der *Host Transport Driver*. Dieser Treiber bestimmt, wie die HCI Kommandos, Events und Datenpakete vom Host Stack zur Hardware und umgekehrt transportiert werden. Die zwei am häufigsten eingesetzten Transport Spezifikationen sind HCI USB (aka H:2) für USB Adapter und HCI UART (aka H:4) für serielle Bluetooth Module. Darüber hinaus gibt es aber auch hersteller-spezifische Formate für z.B. PCMCIA und Compact Flash Karten. Der bekannteste Vertreter ist hier das BCSP von Cambridge Silicon Radio.

Der Bluetooth Host Stack abstrahiert von dem doch sehr hardwarenahen HCI und benutzt dazu das *Logical Link Control and Adaptation Protocol* (L2CAP). Der HCI Layer ist eng an die Paketgrößen des Basebands gebunden und mit dem *Segmentation and Reassembly* (SAR) Mechanismus wird diese Beschränkung aufgehoben. Zusätzlich stellt L2CAP auch einen verbindungsorientierten und einen verbindungslosen Datenkanal mit *Protocol and Service Multiplexing* (PSM) zur Verfügung. Hierdurch ist es höherwertigen Protokollen und Applikationen möglich, Datenpakete mit einer Größe von 64 KB zu senden und zu empfangen. Die Bluetooth 1.2 Spezifikation erweitert die L2CAP Schicht um Flow- und Errorcontrol.

Alle Protokolle über L2CAP stellen eine Verbindung zu Applikationen oder zu Subsysteme des Betriebssystems dar. Zwar ist auch ein direkter Datenaustausch über den L2CAP Layer möglich, aber generell wird immer noch ein weiteres Protokoll benutzt, das die Daten aus den höherwertigen Schichten optimal für den Transport über L2CAP aufbereitet. Der bekannteste Vertreter ist hier das *Bluetooth Network Encapsulation Protocol*, welches IP Pakete in Ethernet ähnliche Frames verpackt um diese dann mit möglichst geringen Overhead über L2CAP zu senden.



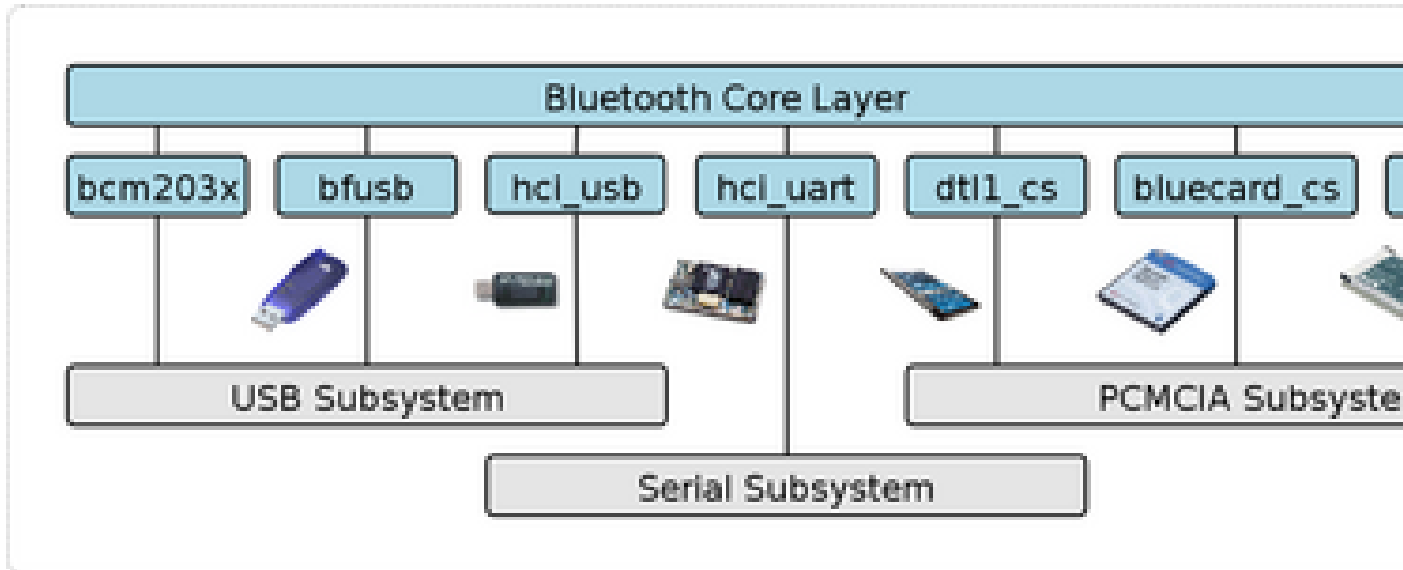
Der eigentliche Bluetooth Host Stack besteht nur aus zwei Protokollebenen. Zum einen ist das die L2CAP Schicht und zum anderen die Verbindungsschicht zu den Applikationen bzw. zu anderen Subsystemen. Die anderen Subsysteme sind etablierte Protokollsuiten, welche für die Benutzung über Bluetooth adaptiert wurden. Als prominente Beispiele wären hier TCP/IP und OBEX zu nennen.

Moderne Bluetooth Chips bieten genügend Rechenleistung um Applikationen direkt auf dem Bluetooth Module ablaufen zu lassen. Hierdurch wird dann kein zusätzlicher Microcontroller benötigt. In solchen Embedded Systemen befinden sich alle Ebenen des Bluetooth Stacks in einem Modul. Da der Speicher bei diesen Systemen aber begrenzt ist, verzichtet man im allgemeinen auf den HCI Layer und benutzt einen L2CAP Layer, der direkt mit dem Link Manager die Daten austauscht.

## 4 Der Aufbau des BlueZ

Die Hauptkomponenten des Bluetooth Stack von Linux sind direkt im Kernel implementiert. Dazu integriert sich BlueZ in das Network Subsystem und stellt eine eigene Adress- und Protokollfamilie zur Verfügung. Als Schnittstellen für die Applikationen wird das Standard BSD Socket Interface benutzt. Dieses grundlegende Designkonzept macht den BlueZ-Stack sehr flexibel und es ist einfach die Bluetooth-Technologie in eigene Applikationen einzubinden. Für die Benutzung von unterschiedlichen Bluetooth-Adaptoren wird eine vollständige Hardwareabstraktion zur Verfügung gestellt. Hiermit ist das Entwickeln von speziellen Hardwaretreibern sehr

einfach.

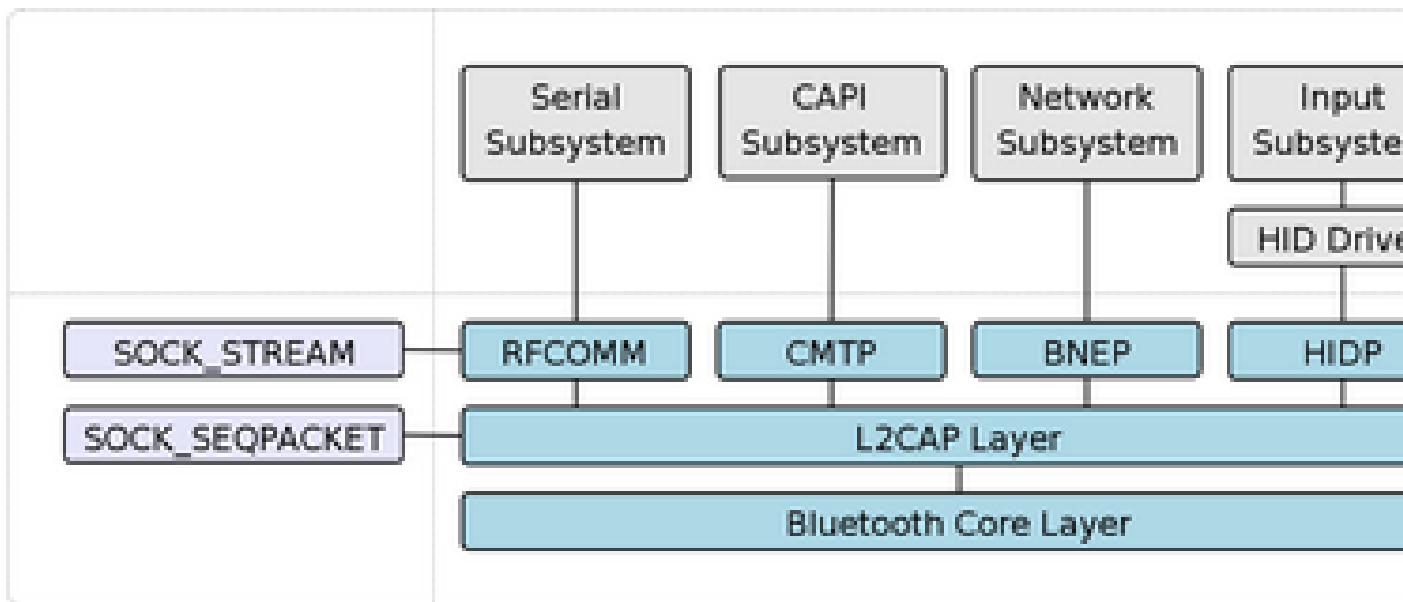


Der Bluetooth Core Layer unterstützt bis zu 16 Adapter gleichzeitig und derzeit sind über 200 verschiedene Bluetooth-Adapter bekannt, die durch einen der Host Treiber unterstützt werden. Auf den BlueZ-Webseiten wird eine stetig wachsende Liste der unterstützten Hardware zur Verfügung gestellt.

Neben der Hardwareabstraktion (dem HCI Layer) sind noch weitere Bluetooth Protokolle als Kernel Code realisiert. Mit dem *Logical Link Control and Adaptation Protocol* wird unter Linux ein BSD Socket Interface mit Sequential Packet Eigenschaften zur Verfügung gestellt. Durch die Bluetooth 1.2 Spezifikation wird hier auch ein Stream orientierter Zugriff zwischen zwei Bluetooth Geräten möglich sein.

Der RFCOMM Layer stellt die Emulation der seriellen Schnittstelle über Bluetooth dar. Dieses so genannte *Cable Replacement for Legacy Applications* bietet unter Linux zwei Schnittstellen an. Zum einen ist das auch wieder ein BSD Socket Interface, aber diesmal mit Stream Eigenschaften. Zum anderen gibt es eine Verbindung zum Linux TTY Layer, so das man für den seriellen Port geschriebene Applikationen ohne den Quellcode zu verändern auch über Bluetooth benutzen kann. Über diese Terminal Emulation kann man dann mit Hilfe des *Point-to-Point Protocol* (PPP) auch Netzwerkverbindungen aufbauen.

Das *Bluetooth Network Encapsulation Protocol* (BNEP), das *CAPI Message Transport Protocol* (CMTP) und das *Human Interface Device Protocol* (HIDP) sind reine Transport Protokolle, die die Verbindungen zum Network Layer, zum CAPI Layer und zum HID Treiber von Linux herstellen. Die Hauptaufgabe dieser drei Protokolle ist es die Datenpakete der jeweils höherwertigen Schicht, möglichst ohne viel Overhead und mit einer geringen Latenzzeit, über L2CAP zu transportieren.



Neben den vorgestellten Protokollschichten, die als Kernel Module realisiert wurden, sind alle anderen Bluetooth Protokolle in Form von Libraries oder direkt in den Applikationen implementiert. Das *Service Discovery Protocol* (SDP) ist eine eigenständige Library, die auf dem L2CAP Layer als Transportmedium aufsetzt und der Bluetooth OBEX Support ist Teil der OpenOBEX Library ([openobex.sf.net](http://openobex.sf.net/) <<http://openobex.sf.net/>>).

## 5 Voraussetzungen für Bluetooth unter Linux

Alle Linux Distribution bringen mittlerweile einen für Bluetooth vorbereiteten Kernel und auch die notwendigen vorkompilierten Pakete mit. Setzt man einen selbstkompilierten Linux Kernel ein, dann sollte man folgende Optionen unter *Bluetooth subsystem* aktivieren:

```

<M> Bluetooth subsystem support
<M>   L2CAP protocol support
<M>   SCO links support
<M>   RFCOMM protocol support
[*]     RFCOMM TTY support
<M>   BNEP protocol support
[*]     Multicast filter support
[*]     Protocol filter support
<M>   CMTTP protocol support
<M>   HIDP protocol support
Bluetooth device drivers --->
<M> HCI USB driver
[ ]   SCO (voice) support
<M> HCI UART driver
[*]   UART (H4) protocol support
[*]   BCSP protocol support
[ ]   Transmit CRC with every BCSP packet
<M> HCI BCM203x USB driver
<M> HCI BlueFRITZ! USB driver
<M> HCI DTL1 (PC Card) driver
<M> HCI BT3C (PC Card) driver
  
```

```
<M> HCI BlueCard (PC Card) driver
<M> HCI UART (PC Card) device driver
<M> HCI VHCI (Virtual HCI device) driver
```

Natürlich sind die Kernel Module nicht alles, was zum Betrieb von Bluetooth unter Linux notwendig ist. Ein großer Teil von BlueZ ist im Userspace implementiert. Um die Bluetooth-Technologie zu benutzen, werden die BlueZ Libraries und die Utilities benötigt. Entweder kompiliert man diese aus den Sourcen oder man installiert einfach die Pakete der Distribution. Ab der Version 2.7 sind keine weiteren BlueZ Pakete notwendig. Alle Libraries und Applikationen zentral in diesen beiden Paketen untergebracht sind.

Da BlueZ sich in den Network Layer integriert, ist es eigentlich nicht notwendig irgendwelche zusätzlichen Device Nodes anzulegen. Erst beim Einsatz der RFCOMM Terminal Emulation werden diese benötigt. Wer unter einem 2.6er Kernel mit *udev* arbeitet braucht sie hier keine Sorgen zu machen. Alle notwendigen Integrationen in das Driver Model von Linux sind vorhanden und die Device Nodes werden bei Bedarf automatisch angelegt. Unter Linux 2.4 ist dies nicht der Fall. Hier muss man das von Hand erledigen, wenn die eingesetzte Linux Distribution dies noch nicht getan hat. In den BlueZ-Utilities gibt es das Script *create\_dev*, das genau diese Aufgabe erledigt. In der Regel werden diese Grundvoraussetzungen von jeder Linux Distribution heutzutage erfüllt und man kann eigentlich gleich loslegen und Bluetooth benutzen.

## 6 Bluetooth-Konfiguration

Nach dem Einstecken eines USB-Adapters oder einer PCMCIA-Karte sollte ein Aufruf von *hciconfig* das Bluetooth-Gerät anzeigen:

```
# hciconfig
hci0:  Type: USB
      BD Address: 00:00:00:00:00:00 ACL MTU: 0:0 SCO MTU: 0:0
      DOWN
      RX bytes:0 acl:0 sco:0 events:0 errors:0
      TX bytes:0 acl:0 sco:0 commands:0 errors:0
```

Dieses *hci0* -Gerät ist, wie eine Netzwerkkarte, noch unkonfiguriert und inaktiv. Um es zu aktivieren, muss das Kommando *hciconfig hci0 up* ausgeführt werden. Danach sollte ein weiterer Aufruf von *hciconfig -a* das Gerät als aktiv anzeigen.

```
# hciconfig -a
hci0:  Type: USB
      BD Address: 00:04:3E:28:09:2D ACL MTU: 678:7  SCO MTU: 120:12
      UP RUNNING PSCAN ISCAN
      RX bytes:101 acl:0 sco:0 events:13 errors:0
      TX bytes:300 acl:0 sco:0 commands:13 errors:0
      Features: 0xff 0xff 0x05 0xf8 0x1b 0x18 0x00 0x80
      Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
      Link policy: RSWITCH HOLD SNIFF PARK
      Link mode: SLAVE ACCEPT
      Name: 'Matsushita Bluetooth Module'
      Class: 0x1c010c
      Service Classes: Rendering, Capturing, Object Transfer
```

```
Device Class: Computer, Laptop
HCI Ver: 1.2 (0x2) HCI Rev: 0xb6 LMP Ver: 1.2 (0x2) LMP Subver: 0x3d
Manufacturer: Zeevo, Inc. (18)
```

Nun sind die Bluetooth-Adresse und auch der Chip-Hersteller zu erkennen. Dieses Device kann jetzt benutzt werden, um nach anderen Bluetooth-Geräten in der Umgebung zu suchen. Außerdem ist es auch für andere Geräte sichtbar. Mit dem Programm *hciconfig* können alle grundlegenden Einstellungen vorgenommen werden. Es ist möglich, den Namen und auch die Geräteklasse, wie z.B. Desktoprechner oder Laptop, zu verändern. Diese Konfiguration muss aber nicht jedes Mal von Hand ausgeführt werden, sondern kann mit Hilfe des *hcid* automatisch geschehen. In der Datei */etc/bluetooth/hcid.conf* werden die grundlegenden Einstellungen für die Bluetooth-Geräte festgelegt, und jedes neue Gerät wird dann mit diesen Eigenschaften konfiguriert und automatisch aktiviert.

Der *hcid* wird normalerweise direkt beim Systemstart aus dem Script */etc/init.d/bluetooth* gestartet und ist die ganze Zeit aktiv. Neben der Gerätekonfiguration ist im *hcid* auch der Security Manager integriert, der die Pin-Abfragen vornimmt und den Austausch der Linkschlüssel verwaltet. Ab der Version 2.6 der BlueZ-Utilities besteht auch die Möglichkeit ein D-Bus fähigen Pin-Helfer zu benutzen. Weitere Informationen hierzu sind in der Manpage des *hcid* zu finden.

Um nun den ersten Kontakt zur Außenwelt aufzunehmen, benutzt man das Programm *hcitool*. Ein einfacher Aufruf von *hcitool scan* sucht nach allen verfügbaren Bluetooth-Geräten in der näheren Umgebung und zeigt diese mit der Bluetooth-Adresse und dem Namen an:

```
# 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: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
```

Ausgestattet mit diesen Informationen ist es nun möglich, die Services der einzelnen Geräte zu ermitteln sowie Verbindungen zu definieren und aufzubauen. Bei neuen und unbekanntenen Geräten hilft das Programm *sdptool*, die unterstützten Services abzufragen:

```
# 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 ReHandle: 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
```

Die oben aufgelistete Ausgabe zeigt einen *Dial-up networking* und einen *Fax-Service* an, welche jeweils eine RFCOMM-Verbindung auf dem Kanal 1 benötigen. Diese notwendige Verbindung kann mit dem Programm *rftcomm* aufgebaut werden.

```
# rftcomm connect 0 00:E0:03:04:6D:36 1
Connected /dev/rftcomm0 to 00:E0:03:04:6D:36 on channel 1
Press CTRL-C for hangup
```

Hierbei wird nun das Terminal Device */dev/rftcomm0* mit dem RFCOMM Kanal 1 auf dem Gerät *00:E0:03:04:6D:36* verbunden und kann danach von jedem anderen Programm wie ein normales Modem benutzt werden. Es ist nur notwendig, */dev/ttyS0* oder */dev/modem* durch */dev/rftcomm0* in der Konfiguration des Einwahlprogramms zu ersetzen. Nach dem Drücken der Tastenkombination *CTRL-C* wird die RFCOMM-Verbindung zum Bluetooth-Gerät wieder beendet. Bei einer regelmäßigen Benutzung ist es sinnvoll, ein Terminal-Device fest an ein Gerät zu binden. Beim Öffnen des Devices wird dann automatisch die Verbindung hergestellt und beim Schließen wieder abgebaut. Dieses Binden von Devices geschieht mit folgendem Befehl:

```
# rftcomm bind 0 00:E0:03:04:6D:36 1
```

Die Syntax dieses Kommandos ist identisch mit dem RFCOMM *connect* -Befehl. Eine Liste aller gebundenen Devices kann mit *rftcomm -a* angezeigt werden.

```
# rftcomm -a
rftcomm0: 00:E0:03:04:6D:36 channel 1 closed
rftcomm1: 00:80:37:06:78:92 channel 1 clean
```

Mit dem Kommando *rftcomm release 0* wird die Bindung von */dev/rftcomm0* wieder entfernt.

Um die Benutzung schon bekannter Bluetooth-Geräte noch einfacher zu gestalten, können diese Bindungen auch in der Konfigurationsdatei */etc/bluetooth/rftcomm.conf* eingetragen werden:

```
rfcomm0 {
    # Bluetooth address of the device
    device 00:E0:03:04:6D:36;

    # Channel for the connection
    channel 1;
}
```

Für alle eingetragenen Terminal-Devices kann man nun Verbindungen ohne Angabe der Bluetooth-Adresse aufbauen. Ein *rfcomm bind all* bindet alle Geräte aus der Konfigurationsdatei an die entsprechenden Devices und eignet sich somit für den Aufruf aus dem */etc/init.d/bluetooth*- Skript.

## 7 Netzwerkverbindungen mit Bluetooth

Um Netzwerkverbindungen zwischen zwei oder mehreren Bluetooth-Geräten aufzubauen, gibt es generell zwei Möglichkeiten. Die *LAN Access using PPP*-Methode benutzt das *Point-to-Point Protocol* über die RFCOMM-Terminal Devices, während PAN auf das *Bluetooth Network Encapsulation Protocol* aufsetzt. Unter Linux können beide Möglichkeiten eingesetzt werden. Für das ältere *LAN Access using PPP* kommt das Programm *dund* zum Einsatz. Eine einfache Verbindung zu einem Access Point kann folgendermaßen etabliert werden:

```
# dund --connect 00:06:C6:C4:08:27
```

Durch den großen Overhead an Protokollinformationen sind die Netzwerkverbindungen mit PPP über RFCOMM nicht besonders schnell. Man verschwendet einen großen Anteil des Datendurchsatzes mit unnötigen Informationen. Mit dem *Bluetooth Network Encapsulation Protocol* wird die Performance deutlich verbessert. Hier wird das Programm *pand* benutzt, welches die gleiche Syntax wie *dund* besitzt. Für eine PAN-Verbindung zu einem anderen Linux-Rechner oder Access Point reicht folgendes Kommando aus:

```
# pand --connect 00:06:C6:C4:08:27
```

Bei einer erfolgreichen Verbindung zeigt dann der Befehl *ifconfig -a* ein *bnepO* Interface an, welches wie jede andere Netzwerkkarte benutzt und konfiguriert werden kann:

```
# 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)
```

Nach dem manuellen Setzen einer IP-Adresse oder der Benutzung von DHCP kann man über dieses Interface Daten mit Hilfe von TCP/IP austauschen. Da sich BNEP unter Linux wie eine Netzwerkkarte verhält, kann man die gewohnten Methoden, wie das *Bridging* oder *Network Address Translation* (NAT), benutzen, um die TCP/IP-Pakete in andere Netze weiterzuleiten. In dem BlueZ PAN HOWTO und auf den Webseiten des Projektes findet man weitere Links und Tipps zur Konfiguration und Installation von Bluetooth und Netzwerken unter Linux.

Eine weitere Art von Netzwerken bzw. Dialup-Verbindungen über Bluetooth ist mit dem *Common ISDN Profile* (CIP) möglich. Hierbei werden ISDN-fähige Bluetooth-Geräte direkt als CAPI-Devices eingebunden. Die Geräte aus der BlueFRITZ! Serie von AVM unterstützen dieses neue Profil bereits. Über die Bluetooth-Luftschnittstelle kommt das *CAPI Message Transport Protocol* zum Einsatz, welches den Transport von CAPI Nachrichten über Bluetooth regelt. Für Linux gibt es ein Kernel-Modul, das es erlaubt, diese Geräte wie jede andere fest eingebaute ISDN-Karte anzusprechen. Ein Verbindung zu einem CIP Gerät erfolgt mit dem Programm *ciptool*.

```
# ciptool connect 00:04:0E:21:06:FD
```

Wenn alles erfolgreich verlaufen ist, wird das Programm *capiinfo* einen neuen ISDN-Controller und dessen Leistungsmerkmale anzeigen. Das so über Bluetooth eingebundene ISDN-Gerät kann nun mit dem CAPI Plugin des *pppd* für die Einwahl ins Internet oder für den Aufbau eines Netzwerkes benutzt werden.

## 8 Drucken über Bluetooth

Die Version 2.7 der BlueZ-Utilities enthält ein CUPS Backend, das es ermöglicht Bluetooth Drucker oder Druckeradapter anzusprechen. Die ersten auf dem Markt erhältlichen Drucker unterstützen nur das Drucken über das *Serial Port Profile*. Da man auch hier RFCOMM als Protokoll eingesetzt hat, wurde unnötig Bandbreite verschwendet. Die Bluetooth SIG hat hier, genau wie mit BNEP, nachgebessert und das *Hardcopy Cable Replacement Profile* (HCRP) entwickelt. Die Definition von HCRP ist relativ allgemein gehalten, aber das Haupteinsatzgebiet sind Bluetooth Drucker und Scanner.

Die Einbindung eines Bluetooth Druckers ist sehr einfach und muß nur einmal durchgeführt werden. Hierbei ist es wichtig, das man die Adresse des Druckers kennt. Diese muss dann in eine URI umgeformt und CUPS beim Einrichten mitgeteilt werden. Eine gültige URI sieht z.B. so aus:

```
bluetooth://00900263E083/
```

Hier wird der Drucker *00:90:02:63:E0:83* angesprochen und beim Angeben der URI lässt man einfach die Doppelpunkte der Bluetooth-Adresse weg. BlueZ unterstützt sowohl das Drucken über RFCOMM als auch über HCRP. Es wird automatisch erkannt welches der beiden Profile unterstützt wird. Bietet ein Gerät beide Dienste an, dann wird HCRP ausgewählt.

## 9 Weitere Bluetooth Anwendungen

Die Einbindung von Bluetooth-Eingabegeräten, wie z.B. Maus und Tastatur, ist derzeit noch in der Entwicklungsphase. Es existiert das Programm *bthid* mit dem man diese Geräte schon jetzt sehr gut unter Linux benutzen kann. In Zukunft sollen diese Geräte mit Hilfe eines generischen HID Treibers im Linux Kernel direkt unterstützt werden. Die BlueZ-Utilities 2.7 enthalten hierzu schon den Code eines ersten Prototypen, für dessen Einsatz ist aber ein Linux 2.6 Kernel und ein zusätzlicher Patch notwendig.

Eine andere sehr erfolgreiche Applikation ist MultiSync ( [www.multisync.org](http://www.multisync.org/) <<http://www.multisync.org/>> ). Hiermit ist es möglich Kontakt- und Kalenderdaten zwischen verschiedenen Systemen drahtlos über Bluetooth zu synchronisieren. Es werden Mobiltelefone, PDAs mit Windows Betriebssystem oder *Open Palm-top Integrated Environment* (OPIE) unterstützt. Durch den modularen Aufbau ist es sehr einfach MultiSync mit eigenen Plugins zu erweitern.

Das Gnokii Project ( [www.gnokii.org](http://www.gnokii.org/) <<http://www.gnokii.org/>> ) demonstriert sehr deutlich, wie einfach es ist, vorhandene Applikationen mit Bluetooth Support auszustatten. Der Anteil an Bluetooth spezifischen Code ist sehr gering, aber das Resultat an Mobilität dafür um so größer.

## 10 Fazit

Mit Hilfe von Bluetooth werden die lästigen Kabel bald der Vergangenheit angehören. Durch die native Integration in den Linux Kernel und die auf BSD Sockets basierenden API entstehen ungeahnte Möglichkeiten zur Nutzung der Bluetooth-Technologie. Mit den RFCOMM Sockets hat man z.B. eine TCP-ähnliche Infrastruktur und kann sehr leicht Client/Server-Systeme aus dem Netzwerkbereich für eine Kommunikation über Bluetooth portieren. Des weiteren können auf einfache Art und Weise vorhandene Programme oder Libraries mit Bluetooth-Funktionalitäten erweitert werden.

Mit dem *GNOME Bluetooth Subsystem* und dem *KDE Bluetooth Framework* gibt es zwei Projekte, die mit Hilfe von BlueZ, den einfachen und grafischen Zugang zur Bluetooth Welt eröffnen wollen. Für die Zukunft sehe ich *Blau...*

## 11 Literatur

1. Special Interested Group Bluetooth: *Bluetooth Core Specification v1.2* , 5. November 2003, <http://www.bluetooth.org/spec/> <<http://www.bluetooth.org/spec/>>
2. Special Interested Group Bluetooth: *Bluetooth Network Encapsulation Protocol Specification* , 14. Februar 2003, <http://www.bluetooth.org/spec/> <<http://www.bluetooth.org/spec/>>
3. Special Interested Group Bluetooth: *Common ISDN Access Profile* , 16. November 2002, <http://www.bluetooth.org/spec/> <<http://www.bluetooth.org/spec/>>
4. Infrared Data Association (IrDA): *Object Exchange Protocol OBEX, Version 1.3* , 2. Januar 2003, <http://www.irda.org/standards/pubs/OBEX13.zip> <<http://www.irda.org/standards/pubs/OBEX13.zip>>
5. Heiko Holtkamp: *Einführung in Bluetooth* , 5. November 2003, <http://www.rvs.uni-bielefeld.de/~heiko/bluetooth/bluetooth.pdf> <<http://www.rvs.uni-bielefeld.de/~heiko/bluetooth/bluetooth.pdf>>
6. Marcel Holtmann: *Zur Zeit als die Pinguine blaue Zähne bekamen* , 19. August 2003, <http://www.holtmann.org/papers/bluetooth/uptimes2003.html> <<http://www.holtmann.org/papers/bluetooth/uptimes2003.html>>