Bluetooth programming for Linux

Marcel Holtmann
BlueZ Project

Andreas Vedral
FH Bochum

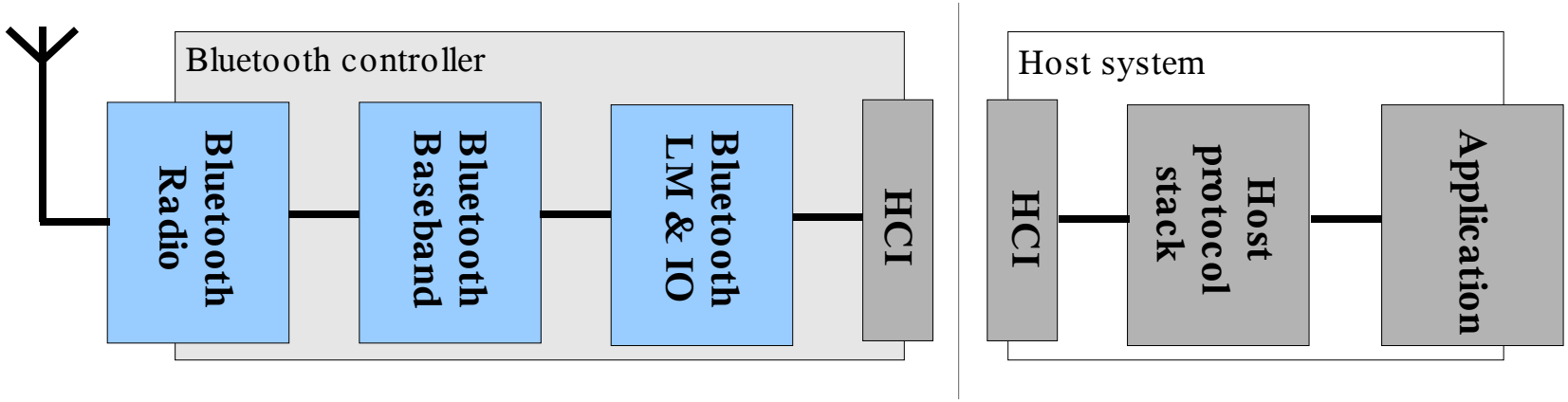
Wireless Technologies Congress 2003
Sindelfingen, Germany

Agenda

- Short introduction into Bluetooth
- History of Bluetooth and Linux
- The official Linux Bluetooth stack
- Integration into other Linux subsystems
- Supported protocols and profiles
- The programming interfaces
- Protocol decoding
- Linux programming tools

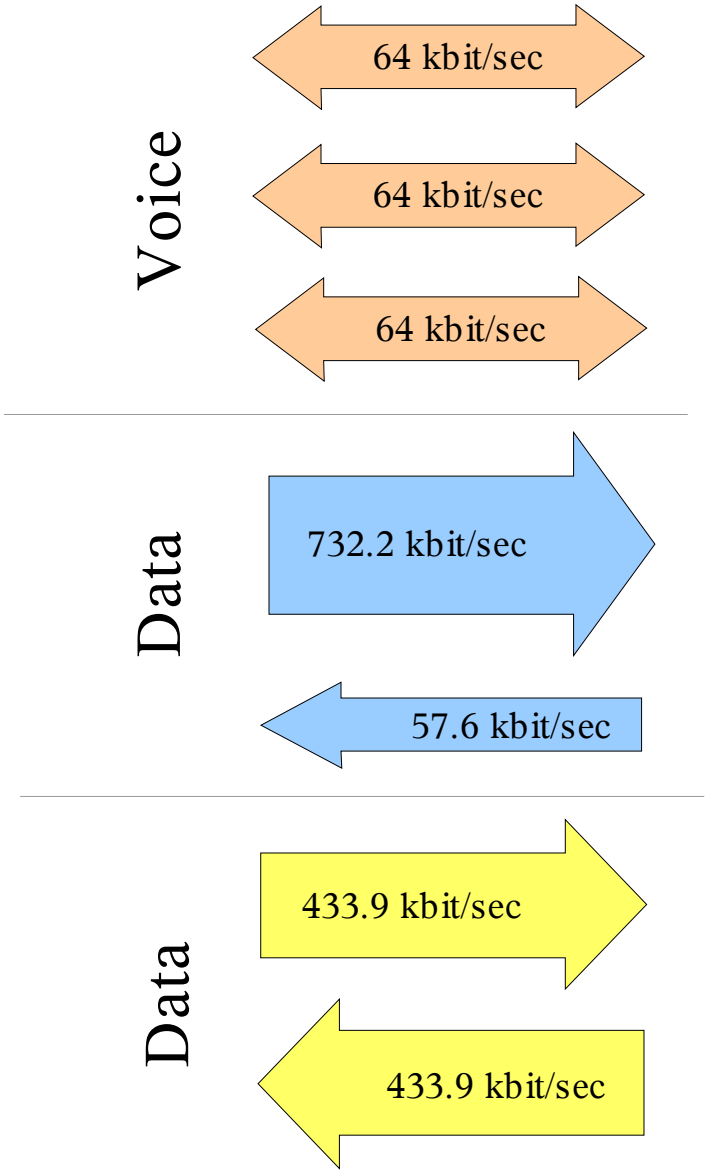
What is Bluetooth?

- Universal interface for wireless communication
  - Communication over short range ad-hoc networks
  - Worldwide license free ISM band
  - Low cost radio modules
  - Minimal power consumption
  - Transmission of data and voice
  - Application profiles



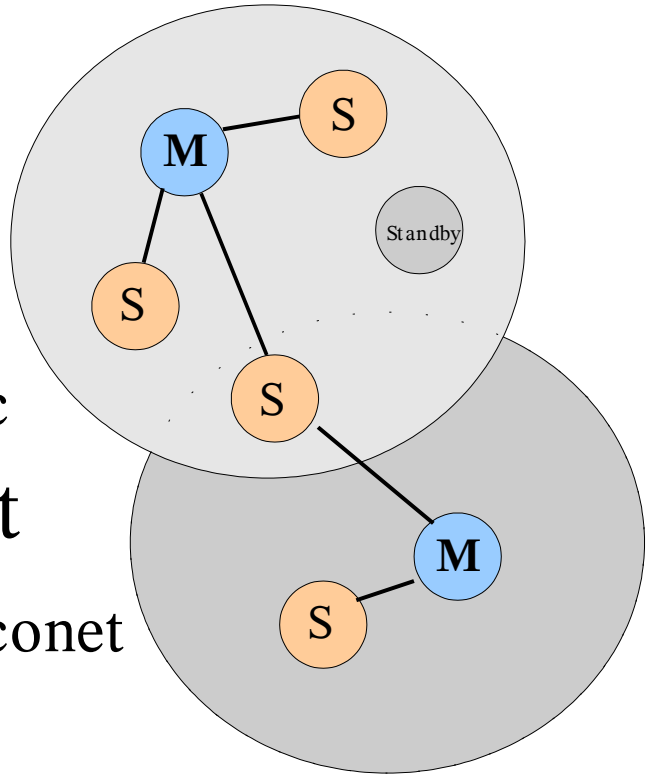
Bluetooth wireless technology

- Up to 3 voice channels
- Asynchrony data transfer
- Symmetric data transfer
- Spread spectrum frequency hopping radio
  - 79 channels
  - Frequency changed after each packet, 1600 hops per second
  - Packet sizes with one, three or five slots

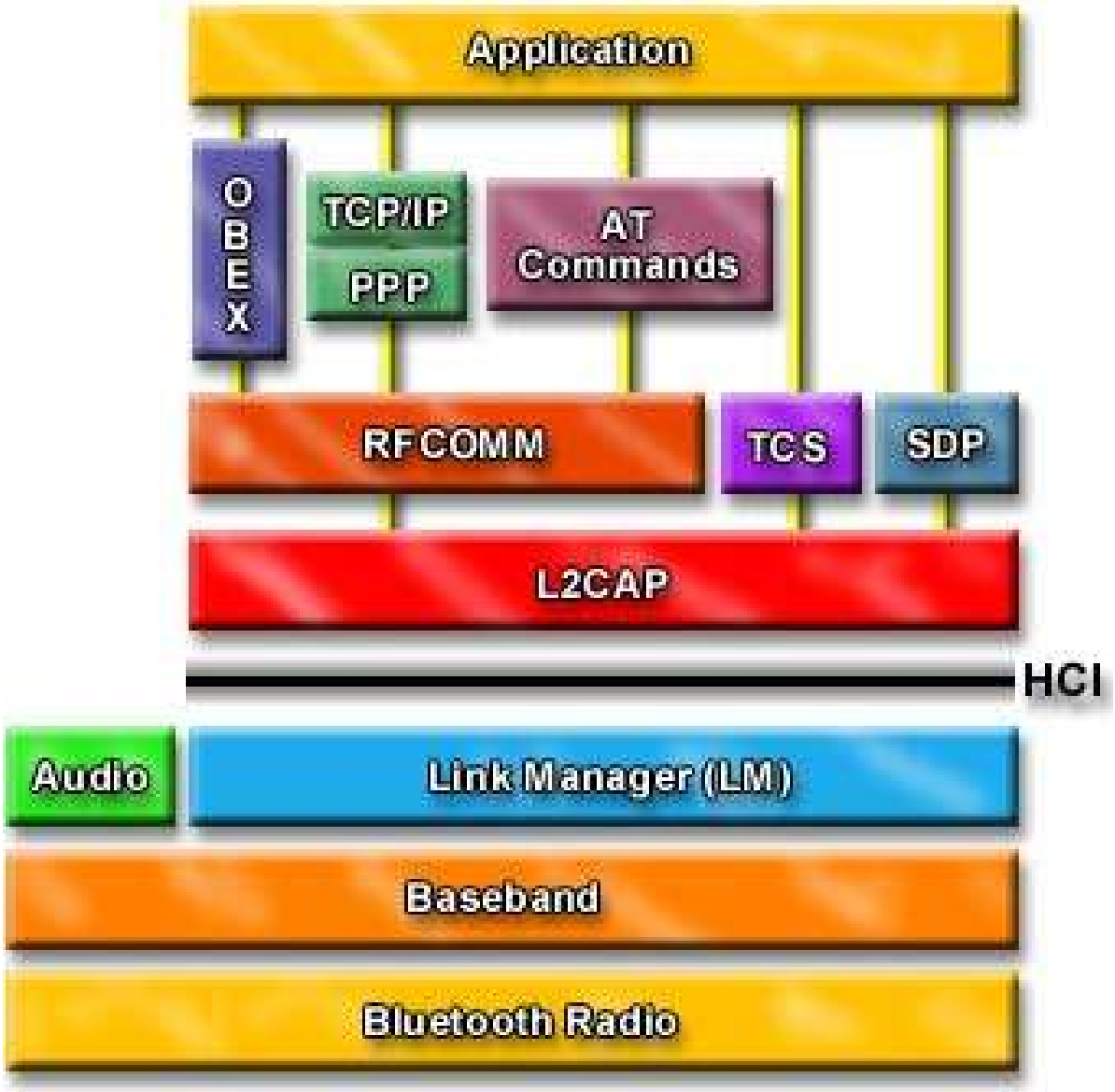


Bluetooth network topologies

- Two or more devices form a piconet
  - Every piconet has one master
  - Up to 7 active slaves
  - Maximal 255 passive devices
  - Master assigns hopping sequence
  - The piconet capacity is 1 Mbit/sec
- Two piconets form a scatternet
  - Master in one, slave in another piconet
  - Slave in two different piconets
  - A device can only be master in one piconet



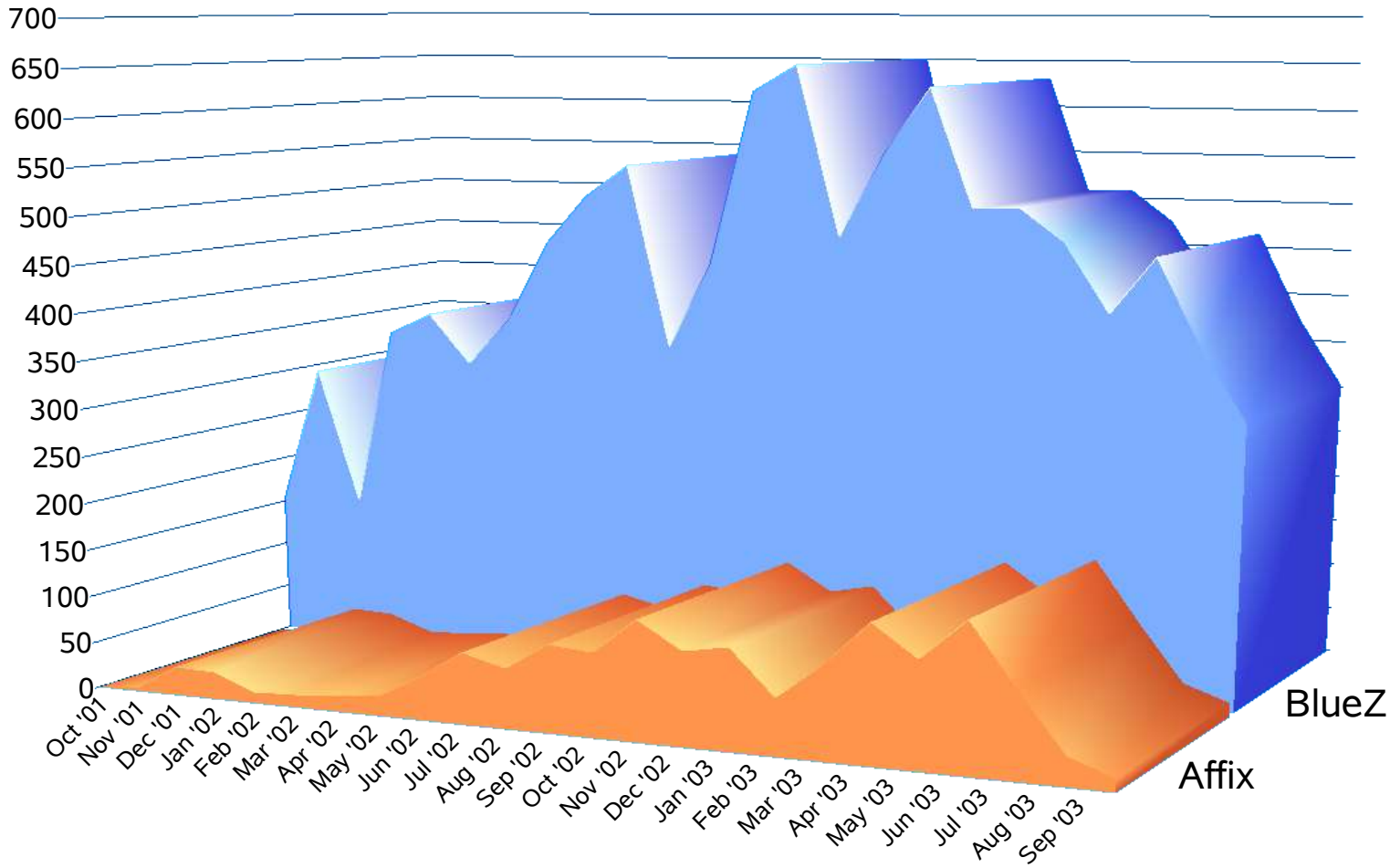
Bluetooth protocol stack



Bluetooth and Linux

- AXIS OpenBT Stack
  - April 1999
  - http://developer.axis.com/software/bluetooth/
- IBM BlueDrekar
  - 25. Juli 2000
  - http://www.alphaworks.ibm.com/tech/BlueDrekar/
- Qualcomm BlueZ
  - 3. Mai 2001
  - http://www.bluez.org
- Nokia Affix Bluetooth Stack
  - 23. November 2001
  - http://affix.sourceforge.net

Traffic on the mailing lists



History of Bluetooth and Linux

- Feb. 1998 Formation of Bluetooth SIG
- 20-05-1998 Announcement of Bluetooth
- Apr. 1999 Announcement of OpenBT
- 26-06-1999 Bluetooth 1.0a specification
- 01-12-1999 Bluetooth 1.0b specification
- 25-07-2000 Announcement of BlueDrekar
- **01-12-2000 Bluetooth 1.1 specification**
- 03-05-2001 Announcement of BlueZ
- **03-07-2001 BlueZ 1.0 is part of Linux 2.4.6**
- 23-11-2001 Announcement of Affix
- 02-08-2002 BlueZ 2.0 is part of Linux 2.4.19
- 28-11-2002 BlueZ 2.2 is part of Linux 2.4.20
- **13-06-2003 Bluetooth is fully integrated into Linux 2.4.21**
- 25-08-2003 Linux 2.4.22 with support for ISDN over Bluetooth
- Nov. 2003 Linux 2.4.23 with qualification ready Bluetooth stack

Bluetooth support for Linux

- Official kernel releases
  - Linux 2.4.18
    - Old core system with USB and UART drivers
  - Linux 2.4.19
    - Updated core system plus Nokia card driver
  - Linux 2.4.20
    - BNEP support plus Anycom and 3Com card drivers
  - Linux 2.4.21
    - RFCOMM support plus BCSP driver
  - Linux 2.4.22
    - CMTP support and BlueFRITZ! USB driver
- Bluetooth patches
  - http://www.holtmann.org/linux/kernel/

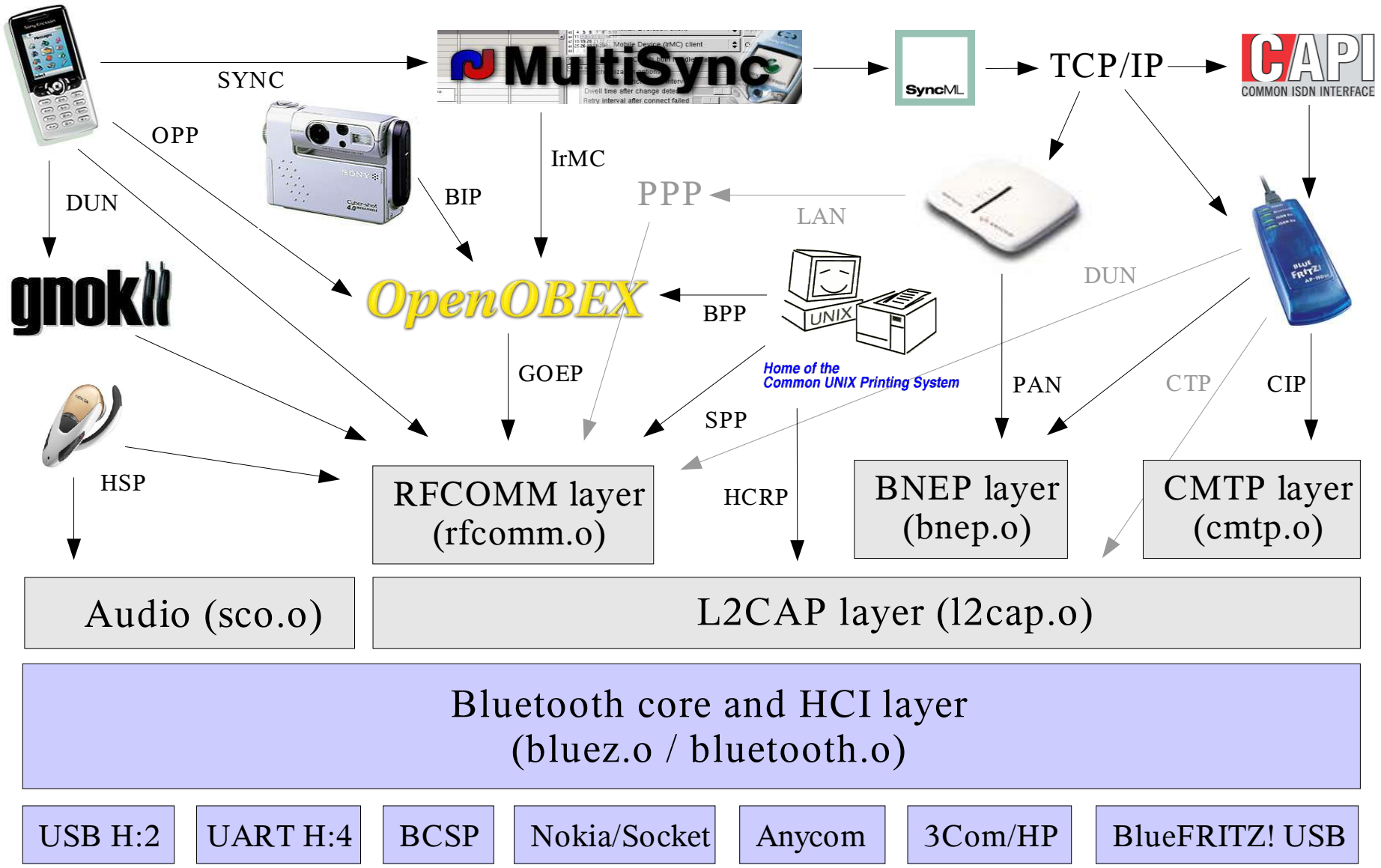
Features of BlueZ

- Real hardware abstraction
  - Generic and vendor specific drivers
  - Over 150 supported Bluetooth adapters
  - Up to 16 host adapters at the same time
- Machine architecture independent
  - Little and big endian
  - 32 bit and 64 bit systems
  - SMP safe
  - Hyperthreading
  - Preempt ready

More features

- BSD sockets interface
  - HCI raw socket
  - L2CAP sequential packet and datagram
  - SCO sequential packet
  - RFCOMM stream
- Complete modular design
  - Kernel code
  - User space programs and tools
- Bluetooth library with user API
  - Handling of Bluetooth addresses and devices

The BlueZ architecture



Supported protocols and profiles

- Current protocols
  - HCI, L2CAP, SDP, RFCOMM, OBEX, BNEP, CMTP, HIDP, HCRP, IrMC, SyncML
- Future protocols
  - TCS-BIN, AVDTP, AVCTP
- Current profiles
  - GAP, SDAP, SPP, GOEP, DUN, FAX, LAN, PUSH, SYNC, FTP, PAN, CIP, HID, HCRP
- Future profiles
  - CTP, Intercom, BPP, BIP, HSP, HFP, SAP

Advantages of BlueZ

- Full source code is available under the GPL
- Socket based interfaces
- Simple API for special HCI or SDP tasks
- Access to all Bluetooth host layers
- Big user and developer community
- Very good interoperability with Bluetooth 1.0b and 1.1 devices
- Full Bluetooth 1.2 support is planned

BlueZ host adapter setup

- Ethernet device like configuration
  - Linux Bluetooth stack specific settings
  - Host controller related configuration

```
# 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

# hciconfig hci0 up
# hciconfig -a
hci0:   Type: USB
        BD Address: 00:50:F2:7A:33:78 ACL MTU: 192:8  SCO MTU: 64:8
        UP RUNNING PSCAN ISCAN
        RX bytes:77 acl:0 sco:0 events:9 errors:0
        TX bytes:30 acl:0 sco:0 commands:8 errors:0
        Features: 0xff 0xff 0x0f 0x00
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH HOLD SNIFF PARK
        Link mode: SLAVE ACCEPT
        Name: 'Microsoft Bluetooth Transceiver'
        Class: 0x000100
        Service Classes: Unspecified
        Device Class: Computer, Uncategorized
        HCI Ver: 1.1 (0x1) HCI Rev: 0x1f9 LMP Ver: 1.1 (0x1) LMP Subver: 0x1f9
        Manufacturer: Cambridge Silicon Radio (10)
```

Using the Bluetooth device

- Simple command line tools
  - Scanning for devices in range
  - Get information about devices and connections
  - Link quality, RSSI, transmit power level etc.

```
# hcitool scan
Scanning ...
        00:02:C7:1E:1D:B8       iPAQ H3970
        00:04:0E:21:06:FD       AVM BlueFRITZ! AP-X
        00:0A:D9:5C:75:57       Sony Ericsson T68i
        00:A0:57:AD:22:0F       ELSA Vianect Blue ISDN
        00:90:02:63:E0:83       Bluetooth Printer
        00:80:37:06:78:92       Ericsson T39m
        08:00:46:0E:CE:50       SONY Cyber-shot
        00:04:61:50:D4:3E       EPox BT-PM01B 80D43E
# hcitool info 00:04:0E:81:06:FD
Requesting information ...
        BD Address:  00:04:0E:21:06:FD
        Device Name: AVM BlueFRITZ! AP-X
        LMP Version: 1.1 (0x1) LMP Subversion: 0x1
        Manufacturer: AVM Berlin (31)
        Features: 0x2f 0xbe 0x05 0x00
                <3-slot packets> <5-slot packets> <encryption> <slot offset>
                <role switch> <RSSI> <channel quality> <SCO link>
                <HV2 packets> <HV3 packets> <A-law log> <CVSD>
                <power control>
```

BlueZ and HCI

- From the user side
  - The BlueZ HCI API is a raw socket
  - The internal protocol is H:4
  - Only one command/event per write/read
  - API for abstracting HCI commands and events
- The kernel side
  - Easy kernel API for writing host drivers
  - Existing drivers for H:2 (USB) and H:4 (UART)
  - Currently 8 different drivers
  - The Affix stack also uses the BlueZ host driver API

Bluetooth user space library

- Handling of Bluetooth device addresses
  - New data type bdaddr_t
  - The special address BDADDR_ANY
  - The functions bacpy, baswap and bacmp
  - Address conversion with str2ba, ba2str, strtoba and batostr

```
#include <bluetooth/bluetooth.h>

void main(int argc, char *argv[])
{
      bdaddr_t bdaddr;
      char *str, addr[18];

      str2ba("00:a5:b4:c3:d2:e1", &bdaddr);

      ba2str(&bdaddr, addr);
      str = batostr(&bdaddr);
      printf("%s %s\n", addr, str);
      free(str);

      bacpy(&bdaddr, BDADDR_ANY);
}
```

The Bluetooth sockets

- Full socket interface
  - L2CAP
    - Connection-oriented (SOCK_SEQPACKET)
    - Connectionless (SOCK_DGRAM)
  - RFCOMM
    - Data stream (SOCK_STREAM)
    - Sockets can be converted to a TTY device
    - Uses the L2CAP in-kernel socket interface
- Complete abstraction from HCI
  - Creation and clearing of ACL connections
  - Sending and receiving of data packets

Bluetooth socket programing

- Definition of the socket data types
  - Address types sockaddr_hci, sockaddr_l2, sockaddr_rc etc.
  - Options for setsockopt, getsockopt and ioctl

```
static int rfcomm_connect(bdaddr_t *src, bdaddr_t *dst, uint8_t channel)
{
      struct sockaddr_rc addr;
      int sk;

      if ((sk = socket(PF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM)) < 0)
            return -1;

      addr.rc_family = AF_BLUETOOTH;
      bacpy(&addr.rc_bdaddr, src);
      addr.rc_channel = 0;
      if (bind(sk, (struct sockaddr *) &addr, sizeof(addr)) < 0) {
            close(sk);
            return -1;
      }

      addr.rc_family = AF_BLUETOOTH;
      bacpy(&addr.rc_bdaddr, dst);
      addr.rc_channel = channel;
      if (connect(sk, (struct sockaddr *) &addr, sizeof(addr)) < 0) {
            close(sk);
            return -1;
      }

      return sk;
}
```

```
str2ba("00:a5:b4:c3:d2:e1", &bdaddr);
sk = rfcomm_connect(BDADDR_ANY, &bdaddr, 1);
```

Host controller programming

- Abstraction of the Bluetooth HCI specification
  - Data types for HCI commands and events
  - General functions for working with HCI raw socket

```
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>

void main(int argc, char **argv)
{
      int dev_id, num_rsp, length, flags;
      inquiry_info *info = NULL;
      bdaddr_t bdaddr;
      int i;

      dev_id  = 0;  /* device hci0 */
      length  = 8;  /* ~10 seconds */
      num_rsp = 10;
      flags   = 0;

      num_rsp = hci_inquiry(dev_id, length, num_rsp, NULL, &info, flags);

      for (i = 0; i < num_rsp; i++) {
            baswap(&bdaddr, &(info+i)->bdaddr);
            printf("\t%s\n", batostr(&bdaddr));
      }

      free(info);
}
```

Protocol traffic decoding

- Recording of HCI packets
  - hcidump
- Only for local connections
  - ACL and SCO data packets
- Decoding of higher protocol layers
  - HCI and L2CAP
  - SDP, RFCOMM, BNEP, CMTP and HIDP
- No sniffing on radio or baseband traffic
  - No replacement for a Bluetooth protocol analyzer

Decoding a RFCOMM session

```
< HCI Command: Create Connection(0x01|0x0005) plen 13
  57 75 5C D9 0A 00 18 CC 01 00 00 00 01
> HCI Event: Command Status(0x0f) plen 4
  00 01 05 04
> HCI Event: Link Key Request(0x17) plen 6
  57 75 5C D9 0A 00
< HCI Command: Link Key Request Reply(0x01|0x000b) plen 22
  57 75 5C D9 0A 00 0D 50 23 41 F0 94 44 8D 9D FC 87 76 14 B8
  DB 84
> HCI Event: Command Complete(0x0e) plen 10
  01 0B 04 00 57 75 5C D9 0A 00
> HCI Event: Connect Complete(0x03) plen 11
  00 29 00 57 75 5C D9 0A 00 01 01
< ACL data: handle 0x0029 flags 0x02 dlen 12
    L2CAP(s): Connect req: psm 3 scid 0x0040
< HCI Command: Write Link Policy Settings(0x02|0x000d) plen 4
  29 00 0F 00
> HCI Event: Command Complete(0x0e) plen 6
  01 0D 08 00 29 00
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 16
    L2CAP(s): Connect rsp: dcid 0x005e scid 0x0040 result 0 status 0
< ACL data: handle 0x0029 flags 0x02 dlen 16
    L2CAP(s): Config req: dcid 0x005e flags 0x0000 clen 4
    MTU 1024
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 17
> ACL data: handle 0x0029 flags 0x01 dlen 1
    L2CAP(s): Config rsp: scid 0x0040 flags 0x0000 result 0 clen 4
    MTU 251
> ACL data: handle 0x0029 flags 0x02 dlen 16
    L2CAP(s): Config req: dcid 0x0040 flags 0x0000 clen 4
    MTU 251
< ACL data: handle 0x0029 flags 0x02 dlen 14
    L2CAP(s): Config rsp: scid 0x005e flags 0x0000 result 0 clen 0
< ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x5e len 4 [psm 3]
      RFCOMM(s): SABM: cr 1 dlci 0 pf 1 ilen 0 fcs 0x1c
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x40 len 4 [psm 3]
      RFCOMM(s): UA: cr 1 dlci 0 pf 1 ilen 0 fcs 0xd7
< ACL data: handle 0x0029 flags 0x02 dlen 18
    L2CAP(d): cid 0x5e len 14 [psm 3]
      RFCOMM(s): PN CMD: cr 1 dlci 0 pf 0 ilen 10 fcs 0x70 mcc_len 8
      dlci 2 frame_type 0 credit_flow 15 pri 7 ack_timer 0 frame_size 246 max_retrans 0 credits 7
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 17
> ACL data: handle 0x0029 flags 0x01 dlen 2
    L2CAP(d): cid 0x40 len 15 [psm 3]
      RFCOMM(s): PN RSP: cr 0 dlci 0 pf 0 ilen 10 fcs 0xaa mcc_len 8
      dlci 2 frame_type 0 credit_flow 14 pri 7 ack_timer 0 frame_size 246 max_retrans 0 credits 7
< ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x5e len 4 [psm 3]
      RFCOMM(s): SABM: cr 1 dlci 2 pf 1 ilen 0 fcs 0x59
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x40 len 4 [psm 3]
      RFCOMM(s): UA: cr 1 dlci 2 pf 1 ilen 0 fcs 0x92
< ACL data: handle 0x0029 flags 0x02 dlen 12
    L2CAP(d): cid 0x5e len 8 [psm 3]
      RFCOMM(s): MSC CMD: cr 1 dlci 0 pf 0 ilen 4 fcs 0x70 mcc_len 2
      dlci 2 fc 0 rtc 1 rtr 1 ic 0 dv 1 b1 0 b2 0 b3 0 len 0
> ACL data: handle 0x0029 flags 0x02 dlen 13
    L2CAP(d): cid 0x40 len 9 [psm 3]
      RFCOMM(s): MSC CMD: cr 0 dlci 0 pf 0 ilen 4 fcs 0xaa mcc_len 2
      dlci 2 fc 0 rtc 1 rtr 1 ic 0 dv 0 b1 0 b2 0 b3 0 len 7
< ACL data: handle 0x0029 flags 0x02 dlen 12
    L2CAP(d): cid 0x5e len 8 [psm 3]
      RFCOMM(s): MSC RSP: cr 1 dlci 0 pf 0 ilen 4 fcs 0x70 mcc_len 2
      dlci 2 fc 0 rtc 1 rtr 1 ic 0 dv 0 b1 0 b2 0 b3 0 len 0
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 13
    L2CAP(d): cid 0x40 len 9 [psm 3]
      RFCOMM(s): MSC RSP: cr 0 dlci 0 pf 0 ilen 4 fcs 0xaa mcc_len 2
      dlci 2 fc 0 rtc 1 rtr 1 ic 0 dv 1 b1 0 b2 0 b3 0 len 7
< ACL data: handle 0x0029 flags 0x02 dlen 9
    L2CAP(d): cid 0x5e len 5 [psm 3]
      RFCOMM(d): UIH: cr 1 dlci 2 pf 1 ilen 0 fcs 0x86
      21
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
< ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x5e len 4 [psm 3]
      RFCOMM(s): DISC: cr 1 dlci 2 pf 1 ilen 0 fcs 0xb8
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x40 len 4 [psm 3]
      RFCOMM(s): UA: cr 1 dlci 2 pf 1 ilen 0 fcs 0x92
< ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x5e len 4 [psm 3]
      RFCOMM(s): DISC: cr 1 dlci 0 pf 1 ilen 0 fcs 0xfd
< ACL data: handle 0x0029 flags 0x02 dlen 12
    L2CAP(s): Disconn req: dcid 0x005e scid 0x0040
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> HCI Event: Number of Completed Packets(0x13) plen 5
  01 29 00 01 00
> ACL data: handle 0x0029 flags 0x02 dlen 8
    L2CAP(d): cid 0x40 len 4 [psm 3]
      RFCOMM(s): UA: cr 1 dlci 0 pf 1 ilen 0 fcs 0xd7
> ACL data: handle 0x0029 flags 0x02 dlen 12
    L2CAP(s): Disconn rsp: dcid 0x005e scid 0x0040
< HCI Command: Disconnect(0x01|0x0006) plen 3
  29 00 13
> HCI Event: Command Status(0x0f) plen 4
  00 01 06 04
> HCI Event: Disconn Complete(0x05) plen 4
  00 29 00 16
```

Programming tools

- Standard tools and compiler
  - gcc, ld, libtool, make etc.
  - autoconf, automake etc.

```
# ls -la
total 38
drwxr-xr-x   2 holtmann staff    296 Sep 25 18:18 .
drwxr-xr-x  21 holtmann staff    624 Sep 25 18:04 ..
-rw-r--r--   1 holtmann staff      0 Sep 25 18:00 AUTHORS
-rwxr-xr-x   1 holtmann staff     68 Sep 25 18:18 bootstrap
-rw-r--r--   1 holtmann staff      0 Sep 25 18:00 ChangeLog
-rw-r--r--   1 holtmann staff    121 Sep 25 18:02 configure.in
-rw-r--r--   1 holtmann staff  18009 Sep 25 18:00 COPYING
-rw-r--r--   1 holtmann staff     30 Sep 25 18:04 main.c
-rw-r--r--   1 holtmann staff     78 Sep 25 18:04 Makefile.am
-rw-r--r--   1 holtmann staff      0 Sep 25 18:00 NEWS
-rw-r--r--   1 holtmann staff      0 Sep 25 18:00 README
```

```
AC_INIT()
AM_INIT_AUTOMAKE(example-project, 0.1)

CFLAGS="-Wall -g -O2"

AC_PROG_CC
AC_PROG_INSTALL

AC_OUTPUT(Makefile)
```

configure.in

```
bin_PROGRAMS = example

example_SOURCES = main.c

example_LDADD = -lbluetooth
```

Makefile.am

```
#!/bin/sh
aclocal \
      && automake -a -c 2> /dev/null \
      && autoconf
```

bootstrap

Any questions
???