Bluetooth Smart Nodes for Mobile Ad-hoc Networks

Jan Beutel
Computer Engineering and Networks Lab
December 17, 2002
Overview

• Why have a Prototyping Platform?

• Short introduction to Bluetooth Concepts

• BTnode hardware features

• System Software

• Pro’s and Con’s – Competitor systems

• Getting started on prototyping – ISP programming

• Demo applications
A device supporting Bluetooth ...

Magnitudes smaller than a PDA ...

... but equally flexible and programmable.
Ad-hoc networking today

- Almost no (large) ad-hoc systems around
- Playing with multihop on WLAN (*Mobihoc, Mobicom*)
- NS-2 plots get boring in the long run
  - Specific MAC layer problem analysis

So why is nobody doing the real thing?

- Bulky
- Short standalone operating times
- Hard to manage multitude of devices
- Limitations on features and communication front-ends
- Often only point to point
- SDK’s only for selected “customers”
The Bluetooth Protocol Stack

Applications

TCP/IP  OBEX

SDP  RFCOMM  ...

L2CAP

Host Controller Interface

Audio

Link Manager

Baseband

RF

on host

on module
Bluetooth Profiles

- Vertical slices of the complete protocol specification
- Application and hardware specific tailoring
- Guarantee interoperability
Host Controller Interface

Standard interface for protocol software that resides on the host processor to access the lower levels of the protocol stack in an attached Bluetooth hardware module.

- **HCI_COMMAND**
- **HCI_EVENT**

<table>
<thead>
<tr>
<th>OGF</th>
<th>OCF</th>
<th>PARAMETERS</th>
</tr>
</thead>
</table>

![Diagram showing the relationship between different protocol layers and interfaces.](image)

- Higher protocol layers on the host processor of the Bluetooth device.
- Physical interface (e.g., USB, RS-232, UART) passes traffic and control signals.
- Lower protocol layers on the Bluetooth module.
Bluetooth Connections

- Managed by the host controller
- Statemachine for each connection

- Link Layer Control & Adaptation (L2CAP)
  - connection-oriented & connectionless data
  - protocol multiplexing for a single “air interface”
  - packet segmentation & reassembly
  - channel abstraction
Bluetooth Piconets

• Communication organized in piconets
  – controlled by one master
  – up to 7 active slaves
  – 255 inactive (parked) slaves

• Master-Slave
  – Implements centralized control
  – Synchronization of all slaves
  – Only master-slave communication

• Multiple piconets
  – separate channels
  – no coordination
Bluetooth Scatternets

- Using gateway nodes
- No coordination
- Role switch necessary

**Problem:**
- Synching to two FH piconets
- Slave is invisible when connected
- Hidden station
- Protocol complexity

- Networking has not yet been defined and is up to the application
  - Draft for Bluetooth Spec. 1.2
Advantages and Caveats

- All digital high level interfaces
- Standard interfaces, available on many platforms
- 100’s kbit/sec data rates
- Advanced protocol features

- Power consumption, Price
  - Is showing significant improvements
- HW and firmware bugs and features
  - Access to developer information
- Connectivity (max. 8 nodes per piconet)
- Complex to handle, mostly because of connection orientated nature, QoS and security issues
- Inquiry and connect takes long
  - Draft for Bluetooth Spec 1.2 has a new scheme here
Hardware Requirements

Autonomous wireless communication and computing platform based on a Bluetooth radio module and a microcontroller.

Requirements:
- Small form factor, low component count
- Standardized wireless interface
- Flexible and cost effective deployment of large quantities of networking nodes
BTnode hardware details

- Atmel ATmega 128l MCU
  8-Bit RISC
  (max. 8 MHz ~8MIPS)
- Real time clock
- 128 kB Flash ROM
  64 kB SRAM
  4 kB EEPROM
- Generic sensor interfaces
- UART and I2C data interface
- Power and frequency management
- Ericsson Bluetooth radio
- Integrated PIFA antenna
Designing for power aware operation

Features
- Single power supply (3.6 – 16 V), single internal voltage (3.3 V)
- MCU with 6 power down modes
- Frequency scaling: 7.3 MHz - 57 kHz
- Battery charge indicator
- Direct current access shunts for all components
- Optional switchable power supply for Bluetooth module
- Internal Vcc available at every connector to power external sensor modules

Power consumption @ 3.3V, 7.3 MHz [mW]  

<table>
<thead>
<tr>
<th>State</th>
<th>max</th>
<th>typ</th>
<th>Lifetime [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth Connected/CPU On</td>
<td>250</td>
<td>160</td>
<td>12-19</td>
</tr>
<tr>
<td>Bluetooth Idle/CPU On</td>
<td>95</td>
<td>67</td>
<td>32-45</td>
</tr>
<tr>
<td>Bluetooth Off/CPU Idle</td>
<td>15</td>
<td>12</td>
<td>202-252</td>
</tr>
<tr>
<td>Bluetooth Off/CPU Sleep</td>
<td>6</td>
<td>&lt;0.5</td>
<td>504-6048</td>
</tr>
</tbody>
</table>

*on 840 mAh Li-ion
# More performance specs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>3000</td>
<td>3200</td>
<td>Pentium4</td>
<td>2000</td>
<td>40000</td>
<td>500</td>
<td>108</td>
</tr>
<tr>
<td>PDA</td>
<td>1000</td>
<td>300</td>
<td>XScale</td>
<td>400</td>
<td>1000</td>
<td>64</td>
<td>11</td>
</tr>
<tr>
<td>BTnode</td>
<td>110</td>
<td>12</td>
<td>ATmega 128l</td>
<td>7.3</td>
<td>0.128</td>
<td>0.064</td>
<td>0.768</td>
</tr>
</tbody>
</table>

**First prototype series**

- Bill of material: 50 parts
- Parts: 90 CHF
- Assembly: 15 CHF
- Bluetooth: 65 CHF
- Unit cost @ 200 units: 170 CHF
System software architecture

Lightweight OS made up of **device drivers** and **dispatcher for scheduling**

- Low level interrupt driven drivers and libraries for peripherals and interfaces
- Event driven application model facilitates coarse grained cooperative multithreading
- 30 kB code size in ROM, 1-2 kB in RAM (with 128 byte UART buffers)
Multiple drivers and the dispatcher

```c
void handler(/* ... */) {}
void main() {
  btn_disp_ev_reg(RECEIVE_EV, handler, 0);
  btn_disp_run();
}
```

**Diagram:**
- Application
  - handler()
  - read()
- Driver
  - read buffer
  - receive_irq()
- Hardware
  - Dispatcher
  - btn_disp_run()
  - event buffer
  - RECEIVED_EV
  - EVENT
  - handler_func
  - RECEIVED_EV, handler()
  - OTHER_EV, func2()
Getting going on BTnodes ...

- Documentation available on the Web
  - Design documents
  - API documentation
  - Links
  - Mailing list

- Software development tools
  - Gnu toolchain
  - avr-gcc, avr-libc, downloader, gdb
  - Available for Linux, Windows and Mac OS X

- BTnode System Software
  - Snapshot packages and CVS repository
Required hardware

• In system programmer
  – Atmel AVR ISP
  – or STK500

• RS-232 level converter

• Some cables
  – Programming cable
  – UART cable
  – 2 x serial 9 pin

• A BTnode
Developer debugging support

• Serial programming on UART using ISP
• Reset button
• 4 LED’s
• JTAG interface
• Serial console for debugging output `printf()`
• Example programs in the BTnode system software

• Linux emulation of BTnode system

• Bootloader resident in Flash memory
  – Remote update of the bootcode via Bluetooth link
  – Use of extra SRAM blocks as cache
How fast is it to get going?

Smart Its Hackfest

SW install          ½ h
First steps on helloworld.c  ½ h
Own program         ½ h

Under 2 hours for 15 newcomers this November in Zurich
A look at our competitors

- UC Berkeley Motes (*Culler et al.*)
- Microsoft wake-on wireless (*Bahl*)
- Motes and IPAQ's (*Estrin*)
- UCLA i-Badge (*Srivastava*)
- UCLA/WINS/Rockwell nodes (*Srivastava*)
- Picoradio testbed (*Rabaey*)
- Amorphous Computing - Pushpins (MIT)
- ParkTab (*Weiser*)
- Active Bat, Piconet (*Hopper et al.*)
- TeCo Smart It's (*Beigl*)

- Wireless LAN
- Pagers
- “Garage door opener” radio systems

- Infineon/Ericsson Bluetooth ROK 104001 module with SDK
UCB Motes compared

- Less I/O and memory
- Originally had a different host CPU, now ATmega128
- Different radio frontend
  - Baseband and MAC processing on host CPU
  - Much more fine grained RTOS
  - Any application development requires RT systems know-how
  - No interface to other wireless enabled appliances

- TinyOS has shown to be pretty hard to get started on
  - Public documentation has not been good
  - Difficult “non standard” design flow
  - Development out of operating system research for ES

- Lot’s of visionary stuff and showcases, not much system documentation
Demo App: Bluetooth enabled Appliances

Btnodes can communicate with other Bluetooth enabled devices using standard Bluetooth profiles for SMS (AT Commands) and object push (OBEX).

Demo App: Better Avalanche Rescue through Sensors

Avalanche victims fatalities
- 0-15min: 8% fatalities
- **15-35min: most victims suffocate**
- 35-90min: 27% survive with air pockets
- 90-130min: suffocation even with air pockets
- >130min: 3% survive e.g. air channels

→time really matters!

Today’s beacon technology very crude
- periodical pulses: the louder, the closer
- 457kHz, 0.1W
- 80m range

December 17, 2002
Demo App: Better Avalanche Rescue through Sensors

- **Heart rate, oxygen saturation**
- **Air pocket detection**
- **Orientation, mobility**

Demo App: Physical Activity Detection Network

- Use multiple motion sensors for context awareness
- Idea: Many sensors reveal "more context"
- Architecture required to combine those sensors
- Map hierarchical topology to human body

Demo App: WURM - Wearable Unit with RC Modules

WURM hardware architecture

- CPU for:
  - legacy C-code, binary only code
  - low-intensity, background tasks
- Reconfigurable unit for:
  - high-performance tasks
  - low-power tasks

Case study: Bluetooth/Ethernet-Bridge

- IP access point for WURM modules using Btnodes
- Soft CPU (LEON, 32bit SPARC)
  - HW minimal TCP/IP stack
  - HW Ethernet MAC

Demo App: XHOP/R-DSR Multihop Network

Bluetooth multihop source routing prototype
• integrated scalable application protocol
• based on Dynamic Source Routing (CMU)
• routing across piconet borders to support >8 nodes

Remote topology discovery
• Script like command language in the payload

xhop: sending of data packets over a predefined DSR route

remote_prog: sending of programs to distant networking nodes for execution

xhop(ABCDE)
inq()
return_result()
xhop(reverse_route)

xhop(ABHJK)
pos()
return_result()
xhop(reverse_route)
The road to 3G

• Shrinking the design by advanced packaging technology
  – Next generation packaging saves 30-50% on component area
  – I/O and antenna become the size constraints

• Check for feasibility of ROK 104001 module with embedded Arm and SDK?

• Power – drop Vcc to 3.0 and beyond
  • Implement adaptive power/frequency control in software
  • Remote programming, debugging and management functions

• Custom radio frontend?
• Multiple frontends?
Who is using BTnodes today?

Successful deployment of 200 units with 12 groups worldwide

- DSG, ETH Zurich
- PCCV, ETH Zurich
- Computing Department, Lancaster University, UK
- TecO, University of Karlsruhe, Germany
- PLAY, Interactive Institute, Sweden
- VTT, Finland
- TIK, ETH Zurich
- IFE Wearable Lab, ETH Zurich
- NTTDoCoMo, Munich, Germany (Anthony Terlano)
- Ptolemy Group, UC Berkeley, USA (Jörn Janneck)
- Art of Technology, Zurich, Switzerland (Michael Scheffler)
- DistLab, Diku, Copenhagen, Denmark (Philippe Bonnet)
Acknowledgements

The Btnodes have been developed in cooperation with the Smart Its Project that is part of the EU Disappearing Computer Initiative.

- Oliver Kasten (DSG)
- Matthias Ringwald (DSG)
- Matthias Fries (IFH)
- Frank Siegemund (DSG)
- Florian Michahelles (PCCV)
- Holger Junker (IFE)
- Christian Plessl (TIK)

- Our students
  - Urs Frey
  - Urs Bär
  - Peter Fercher, Egon Burgener
  - Lars Wernli, Riccardo Semadeni
  - Lukas Karrer, Thomas Moser
BTnode Resources

Web page
http://www.tik.ee.ethz.ch/~beutel/btnode.html
http://www.inf.ethz.ch/vs/res/proj/smart-its/btnode.html

Mailing list
http://www.ee.ethz.ch/~slist/btnode-development/