BTnodes

A Distributed Environment for Prototyping Ad Hoc Networks

Jan Beutel
Outline

BTnode - Ad hoc networking prototyping platform

Constructing network topologies using Bluetooth

Implementation of a robust, self-healing tree topology

Outlook on our current work: Bluetooth v1.2 and BTnode rev3
Initial projects

**Smart everyday objects**

by attaching sensor nodes:
- self aware
- context sensitive
- cooperative
- integration into computing environment

**Ad hoc networking scenarios**

- integrated application protocols
- scalable multi-hop routing

**Wearable Computing**

**Ubiquitous Computing**
Consumer Electronics Integration

- Mobile Phone
- BTnode
- PDA
- Camera
- BTnode
- PC Peripherals
Backend Connectivity

Connectivity to
- application servers
- other networks

Clusters of mobile networks
- using GSM
- using SMS services
- Wireless LAN
- interfacing to other sensor networks
BTnode architecture

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

Bluetooth has the advantage of
  – availability today for experimentation
  – compatibility to interface to consumer appliances
  – an abstract, standardized high level digital interface
Bluetooth prototyping platform

**Integrated hardware features**
- 8-Bit RISC, max. 8 MIPS, 128 kB Flash, 64 kB SRAM, 180 kB data cache
- operating from 3 cell batteries
- generic sensor interfaces

**Event-driven lightweight OS**
- standard C language
- system software available as library

<table>
<thead>
<tr>
<th>Current bill of material</th>
<th>50 parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parts</td>
<td>60 USD</td>
</tr>
<tr>
<td>Assembly</td>
<td>5 USD</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>45 USD</td>
</tr>
</tbody>
</table>

Unit cost @ 200 units 110 USD
Large ad hoc network topologies

How to construct an ad hoc network topology with Bluetooth
- large network, many devices
- all devices connected, supporting transparent multihop transport

Understanding the limits and benefits of Bluetooth

XHOP prototype

TreeNet topology
BTnode networking – definitions

Four states
- IDLE
- MASTER
- SLAVE
- MASTERSLAVE

Useful operations
- inquiry() – find other nodes
- connect() – open connection
- roleSwitch() – change MS relation
- sendData() – data transport

Hardware limitations on the BTnodes/Bluetooth
- max. 7 active slaves in one Piconet
- while in inquiry() and connect() a node is not visible
- while in SLAVE or MASTERSLAVE a node is not visible
- while in SLAVE or MS a node cannot do inquiry() or connect()
- inquiry() and connect() have long delays and no a priori guarantee

Bluetooth only defines single hop Master-Slave data transport
Distributed Bluetooth Piconets

Distributed `inquiry()` and `connect()` is a problem

- nodes are uncoordinated
- limited visibility
- asymmetry: inquired node doesn’t notice

`Inquiry()` and `connect()` have long delays

- state change in remote node goes unnoticed
- average delay in seconds [Kasten2001]
- no a priori guarantee for success

`Inquiry()` and `connect()` are highly nondeterministic (both in timing and function)
Purpose of this study:

How can we construct ‘arbitrarily’ large trees in a robust and distributed way?
TreeNet simple tree construction

Every node executes algorithm
  – until single tree is reached

Formation of large topologies
  – robustness
  – simplicity
  – redundancy
  – distribution
  – self-healing

Services and applications can break up trees later
  – forming other topologies
  – optimizing topology

loop {
  inquiry();
  forall (nodes_found) do {
    while (not_max_degree)
      connect();
  }
}
TreeNet Demo at Monte Verita

Shown at MICS Annual Review

- **ba** – Blink All – all nodes flash once
- **b** – Blink Levels – starting from the root, all levels flash once
TreeNet discussion

Nodes must all be in visible range

Might not fully connect if multiple max_degree roots form
- rebuilding of partial trees necessary if nodes cannot connect at root

Simple greedy algorithm reduces inquiry() and connect()
- better performance by caching and time-stamping inquiry() and connect()
- try to connect() to node-last-seen first
- exchange of topology data and adaptive connect() retries
BTnode rev3 ongoing work

**Bluetooth v1.2 frontend**
- faster connections
- Adaptive Frequency Hopping (AFH)
- BT v1.2 Scatternets
- class 1 operation supported

**Need for more devices**
- design shrink, physical integration
- lower power

**Make BTnodes available as a platform kit for partners**

**Rethinking of BTnode System Software**
After over 3 years of BTnodes

200 units with 16 research groups
- smart objects
- networking
- wearable computing
- perceptual computing
- operating systems

Good community interaction

Open Source
- BTnode System Software
- examples
- BTnode design data
- development tools
To probe further...

BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks

Welcome to the BTnode Platform

Overview

The BTnode is an autonomous wireless communication and computing platform based on a Bluetooth radio and a microcontroller. It serves as a demonstration platform for research in mobile and ad hoc connected networks (MANETs) and distributed sensor networks. The BTnode has been jointly developed by the Computer Engineering and Networks Laboratory (TIK) and the Research Group for Distributed Systems at ETH Zurich. Currently the BTnode is primarily used in two major research projects: NCCR-MICS and SmartIt.

BTnode features at a glance

- Microcontroller: Atmel ATmega 128L (8 MHz @ 6 MIPS)
- Memory: 64 kbyte RAM, 128 kbyte FLASH ROM, 4 kbyte EEPROM
- Bluetooth radio module, Ericsson ROK 101 007
- External Interfaces (SPI, UART, SPI, I2C, GPIO, ADC, …)
- 4 LEDs
- Standard C Programming

Quickstart

To get going is quite straightforward. Before you can start off developing applications for the BTnode you need to

- Download and install the development tools (compilers). See the tools section.
- Download and install the BTnode System Software. See Installing the BTnode System Software.
- Buy a hardware programmer. We recommend the Atmel STK 500 programmer.
- Build your own programming cable (one per programmer). See Hardware.
- Get BTnodes or serial Bluetooth devices that can be used in emulation mode.
- Compile and download your first example application.

http://www.btnode.ethz.ch

Copyright (c) 2000-2003 BTnode Project