Scaling it up
Networking using the BTnode Platform

Jan Beutel
Outline

Wireless Sensor Networks – visions and current status

Example: Constructing network topologies using Bluetooth
  – BTnode – Ad hoc networking prototyping platform
  – robust, self-healing tree topology TreeNet Algorithm
  – implementation requirements and issues

Lessons learned

Open issues for large scale deployment
Wireless Sensor Networks visions

Large scale of proposed systems
- centralized, decentralized, clustered
- very few, many, massive amounts
- functionally rich, constrained
- homo-, heterogeneous
- self-configuring, managed
- failure tolerant, QoS

Smart Dust [Kahn1999]
Paintable Computing [Butera1999]
Picoradio [Rabaey1999]
Terminodes [Hubaux1999]
Amorphous Computing [Abelson2001]
Specnet [Arvind2003]
Diffusion [Estrin2000]
WINS [Pottie2000]
Wireless Sensor Network systems today

Sub mm scale, super high density all the way to layered, semi infrastructure dependant iPAQ/PC architecture nodes.
BTnode prototyping platform

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 architecture details

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

Current bill of material | 50 parts
---|---
Parts | 60 USD
Assembly | 5 USD
Bluetooth | 45 USD

Unit cost @ 200 units | 110 USD
Bluetooth architecture details

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

Current bill of material

<table>
<thead>
<tr>
<th>Part</th>
<th>Unit Cost</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

BTnode rev3 out soon
Many successful BTnode applications

- The Lighthouse location system [Roemer2003]
- Smart product monitoring [Siegemund2002]
- Bluetooth enabled appliances [Siegemund2003]
- Smart It’s friends [Siegemund2003]
- XHOP/R-DSR multihop prototype [Beutel2002]
- Distributed positioning – TERRAIN implementation [Frey2003]
- Physical activity detection network [Junker2003]
- Better avalanche rescue through sensors [Michahelles2002]
- Wearable unit with reconfigurable modules [Plessl2003]
- Undergrad projects with Lego Mindstorms [Blum2003]
- …

Mostly relying on simple point to point data links
Constructing large network topologies

How to construct an ad hoc network topology with Bluetooth

- large network, many devices
- all devices connected, supporting transparent multihop transport
Every node executes algorithm
  – until a single tree is reached

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

Demonstrated with 40 nodes at NCCR-MICS annual review

```
loop {
inquiry();
forall (nodes_found) do {
  while (not_max_degree)
    connect();
}
```
Lessons Learned

A. A 7 line high level algorithm leads to about 2000 lines of code.

B. It is very difficult to test, debug, deploy and evaluate a large amount of devices.
A. Code size and complexity

Lockup issues
- might not fully connect if multiple max_degree roots form
- distributed inquiry() and connect() problem

Performance issues
- simple greedy algorithm reduces inquiry() and connect()
- highly non-deterministic behavior

Basic underlying infrastructure
- data storage and exchange
- timing and time-stamping
- connection/link management

Leads to about 2000 lines of additional code!
B. Large scale distributed deployment

So why do we actually need even more lines of code?

- additional system software + debugging + visualization + monitoring
- stepwise testing and deployment
- result in an ~87 kB program (un-optimized)

Other problems we had with deployment

- cables
- batteries
- mounting/casing
- (re-)programming
- debugging of a distributed concurrent system
- developing for stepwise deployment
- visualization/analysis
- online access to nodes
- …
It is hard to deploy anywhere beyond 10-20 nodes today.

Coordinated methods and tools are missing today.
Motivation for future work

A. Models and methods for the design.
   - How do we deal with unreliable links?
   - Do we need a model that enables formal verification and optimization of parameters (e.g. time outs)?
   - How do we integrate this into OS and deployment concepts?

B. Deployment.
   - Is there a methodology for a stepwise refinement?
   - How do we debug and quantify?
   - Situation today:
Acknowledgements

**BTnode/TreeNet collaborators**
- Oliver Kasten, Friedemann Mattern, Matthias Ringwald, Kay Römer, Frank Siegemund
- Regina Bischoff, Roger Wattenhofer, Aaron Zollinger
- Jan Beutel, Matthias Dyer, Lennart Meier, Martin Hinz, Lothar Thiele

**Related publications**

**Thanx for material to**
- Deborah Estrin, Jeremy Elson, Ivo Locher, Mani Srivastava, UCLA
- David Culler, Rob Szewczyk, Paul Lewis, Jan Rabaey, Brian Boser, UC Berkeley
- Ralph Kling, Intel Research