Implementing Software on Resource-Constrained Mobile Sensors
Experience with Impala and ZebraNet

T. Liu, C. Sadler, P. Zhang, and M. Martonosi, MobiSys ‘04

*Presented by Fabián E. Bustamante (based on the authors’ presentation)*
Tracking wildlife

• Focus of biology and biocomplexity research
  ▫ Gather data for different species on interaction and mutual influence
    • E.g. How does human development impact indigenous species?

• Current tracking technology
  ▫ Most common VHF transmitter collar with triangulation
    • Researchers drive through/fly over with a receiver antenna to listen for pings and then track animals
    • Infrequent data collection, mostly during the day

  ▫ Small GPS-based trackers
    • Commonly must retrieve collar to get data
    • Few observation points, no recharge
    • Satellite uploads – high energy and low bitrate
ZebraNet

- Collaboration between CS & wildlife biologist researchers
- Biologist goals
  - Detail positioning information (GPS position every 3’)
  - Operation over a large area (1,000km²), for long unattended periods (1 year)
  - Weight bet/ 3-5lbs
  - No fixed base stations, antennas, etc.
  - Delay tolerant
- Implications
  - Delivery of position data should ~100%
  - Weight limits restrict power availability
  - Large areas means store-and-forward
  - Long-term use in the wild means on-the-fly software updates
ZebraNet

Store-and-forward communications

Tracking node with CPU, FLASH, radio and GPS

Base station (car or plane)

<table>
<thead>
<tr>
<th>WSN Attributes</th>
<th>ZebraNet</th>
<th>Other WSNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node mobility</td>
<td>High</td>
<td>Static or moderate</td>
</tr>
<tr>
<td>Communication range</td>
<td>Miles</td>
<td>Meters</td>
</tr>
<tr>
<td>Sensing frequency</td>
<td>Constant</td>
<td>Sporadic</td>
</tr>
<tr>
<td>Sensing device power</td>
<td>Hundreds of mW</td>
<td>Tens of mW</td>
</tr>
</tbody>
</table>
Software design rationales

System layering

Monolithic approach: Software modularity

Middle ground

Layered approach: Layering overhead

Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Software design rationales

Middleware weight

Limited services: application simplicity

Middle ground

Overloaded services: Middleware overhead

Mini-Middleware

Impala

Super-middleware

Hardware

Applications

Hardware

Applications
Impala overview

Application modularity, simplicity, adaptivity and repairability

- Software adaptation for sensor network performance
- Software update for sensor network deployment
- Operation scheduling for regular operations
- Event handling for irregular events
- Network support for sensor network communications
Impala overview

Application modularity, simplicity, adaptivity and repairability

Prototyped on iPAQs

This Paper: on ZebraNet node
Hardware design and constraints

- Memory constraint: 2KB for data, 60KB for program
- Energy constraint: 3 days, 50hrs to recharge
- Device access constraint: GPS and radio interference, e.g.
- Radio packet size constraint: 64B
- GPS sensing time constraint
- FLASH storage constraint

### Power Measurement (4.0V applied)

<table>
<thead>
<tr>
<th>System Mode</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU at 32KHz</td>
<td>9.6 mW</td>
</tr>
<tr>
<td>CPU at 8MHz</td>
<td>19.32 mW</td>
</tr>
<tr>
<td>8MHz w/ GPS</td>
<td>568 mW</td>
</tr>
<tr>
<td>8MHz w/ radio transmit</td>
<td>780 mW</td>
</tr>
<tr>
<td>8MHz w/ radio receive</td>
<td>312.4 mW</td>
</tr>
</tbody>
</table>

### Microcontroller
- TI MSP430F149
  - 16-bit RISC
  - 2KB RAM
  - 60KB ROM
  - 8MHz/32KHz dual clock

### FLASH
- ATMEL AT45DB041B
  - 4Mbit
  - 78 days data capacity

### Radio
- MaxStream 9Xstream
  - 902-928MHz
  - 19.2Kbps over-the-air
  - 0.5-1mile transmit range

### GPS
- µ-blox GPS-MS1E
  - 10-20s position fix time

### Power Supplies
- 5V
- 3.3V

### Charging Circuits

### Solar Modules
Layers & I/F for complex hardware

- Asynchronous Network Transmission
- Protected FLASH Access
- Application Timer Control
- System Clock Time Read

GPS Data Event Handler
Application Timer Event Handler
Network Packet Event Handler
Network Send Done Event Handler

- GPS Data Event
- Radio Packet Event
- Timer Event

Access and Control to All Devices

Adapter
Updater
Operation Scheduler
Event Filter

Network Support

CPU
Radio
GPS
FLASH
Timer
WDT
Memory footprint of Impala layers

**Code Size**

- Firmware
- Impala
- Application
- Unused

**Data Size**

- Firmware
- Impala
- Application
- Unused

Network interface is the largest piece

Only one app
Regular operation scheduling

- GPS-aided operation synchronization
- Prohibited simultaneous device operations
- Non-trivial radio wake-up time
- Potentially long GPS sensing time (10-40 sec for fixing)
- Split transaction
- Stringent energy budget

1. CPU wake up/Radio, FLASH power on
2. Radio transmitting / receiving start
3. Network communication time
4. Radio power off/FLASH power off
5. GPS power on
6. GPS sensing time
7. GPS power off / FLASH power on
8. CPU sleep / FLASH power off
## Operation scheduling overhead

<table>
<thead>
<tr>
<th>Scheduling Type</th>
<th>Impala Activity</th>
<th>Time (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU Scheduling</strong></td>
<td>To put CPU on the full-speed clock</td>
<td>3127</td>
</tr>
<tr>
<td></td>
<td>To put CPU on the slow clock</td>
<td>38</td>
</tr>
<tr>
<td><strong>Radio and FLASH Scheduling</strong></td>
<td>To set up the first transmission time and turn on radio and FLASH</td>
<td>50 ms</td>
</tr>
<tr>
<td></td>
<td>To set up the next transmission time</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>To set up the network cleanup time</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>To clean up incomplete network messages, power off radio and FLASH, and set up</td>
<td>11 ms</td>
</tr>
<tr>
<td></td>
<td>the next networking period</td>
<td></td>
</tr>
<tr>
<td><strong>GPS Scheduling</strong></td>
<td>To initiate GPS sensing and set up its finish time</td>
<td>1247</td>
</tr>
<tr>
<td></td>
<td>To format GPS data, power off GPS, signal an GPS data event, and set up the next</td>
<td>2550</td>
</tr>
<tr>
<td></td>
<td>GPS sensing period</td>
<td></td>
</tr>
</tbody>
</table>
Handling irregular events

**Issue One: Event Abstraction**
- Abstract Application Events
- Miscellaneous Hardware Interrupts

**Issue Two: Concurrency**
- Short / atomic hardware interrupts
- Long / preemptive software events

**Issue Three: Event Prioritization**
- A priority ordered set of queues, one for each event type

![Diagram with event signals and priorities]
Special communication needs

- Peer discovery or data
- Session-based
  - Msg designated by the app to have network transaction semantics
  - 1 to 32KB
  - Header information in != packet
- One, more or unlimited destinations
- Data from FLASH or RAM
- Acknowledgment or not
- Connectionless

![Diagram showing communication types: Reliable Unicast, Reliable Multicast, Unreliable Broadcast]
Unified MAC & transport control

- **MAC**: round-robin timeslots (w/ GPS-aided time synchronization)
- **Transport control**: detect packet loss and retransmit
- **Unified**: reduce data copy & overhead across layers
  - Replace data buffering with indexing
  - Packet, rather than session buffering, in network reception – complicates apps
- **Asynchronous**

---

**Diagram:**

- A sends packet 1-8 to B, C, and D
- B acks packet 1-8 C acks packet 1-4 D silent
- B acks packet 1-8 C acks packet 1-4 D acks packet 1-8
- B gives up C and sends packet 9-16
Conclusions

• Propose and implement Impala middleware model
• Solutions for hardware constraints and application requirements
• Concrete experience with real system and application
• To-do: Refine collar design; switch to lower-energy GPS, merge Impala software update code into final collars