A System Architecture for Networked Sensors

Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kris Pister

http://tinyos.millennium.berkeley.edu
U.C. Berkeley
11/13/2000
Computing in a cubic millimeter:

- Advances in low power wireless communication technology and micro-electromechanical sensors (MEMS) transducers make this possible.
- How do you combine sensing, communication and computation into a complete architecture?
- What are the requirements of the software?
- How do you evaluate a given design?
Ad hoc sensing

- Autonomous nodes self assembling into a network of sensors
- Sensor information propagated to central collection point
- Intermediate nodes assist distant nodes to reach the base station
- Connectivity and error rates used to infer distance

http://tinyos.millennium.berkeley.edu
Organization

- The Vision
- Hardware of today
- Software Requirements
- TinyOS system architecture
- System evaluation

http://tinyos.millennium.berkeley.edu
Today’s Hardware

- Assembled from off-the-shelf components
- 4Mhz, 8bit MCU (ATMEL)
  - 512 bytes RAM, 8K ROM
- 900Mhz Radio (RF Monolithics)
  - 10-100 ft. range
- Temperature Sensor & Light Sensor
- LED outputs
- Serial Port

http://tinyos.millennium.berkeley.edu
No dedicated I/O controllers

- Bit by bit interaction with Radio
- Software must process bit every 100µs
- No buffering
  - missed deadline $\rightarrow$ lost data
- Must time share on granularity of 2x sampling rate

http://tinyos.millennium.berkeley.edu
Key Software Requirements

- Capable of fine grained concurrency
- Small physical size
- Efficient Resource Utilization
- Highly Modular
TinyOS system architecture
State Machine Programming Model

- System composed of state machines
- Command and event handlers transition a module from one state to another
  - Quick, low overhead, non-blocking state transitions
- Many independent modules allowed to efficiently share a single execution context

http://tinyos.millennium.berkeley.edu
Simple State Machine Logic

Bit_Arrival_Event_Handler
State: \{bit_cnt\}

- Don’t you have to do computational work eventually?
- “Tasks” used to perform computational work
TinyOS component model

- Component has:
  - Frame (storage)
  - Tasks (computation)
  - Command and Event Interface
- Constrained Storage Model allows compile time memory allocation
- Provides efficient modularity
- Explicit Interfaces help with robustness

http://tinyos.millennium.berkeley.edu
TinyOS – The Software

- Scheduler and graph of components
  - constrained two-level scheduling model: tasks + events
- Provides a component based model abstracting hardware specifics from application programmer
- Capable of maintaining fine grained concurrency
- Can interchange system components to get application specific functionality

http://tinyos.millennium.berkeley.edu
Composition into a Complete Application

Routing Layer

Messaging Layer

Radio Packet

Radio byte

RFM

photo

Temp

ADC

i2c

clocks

application

sensing application

Routing

messaging

packet

byte

bit

HW

SW

http://tinyos.millennium.berkeley.edu
The Application

- The single node application is just another state machine

```java
Message_Handler(incoming_message) {
    if(sender_is_better_parent()) {
        my_parent = sender();
    } else if(I_am_parent_of_sender()) {
        forward_message(my_parent, incoming_message);
    }
}
Clock_Event_Handler() {
    check_expire(my_parent);
    if(my_parent != null) {
        send_data(my_parent);
    }
}
```

http://tinyos.millennium.berkeley.edu
Analysis

- Let’s take apart Space, Power and Time
Space Breakdown...

Code size for ad hoc networking application

- Interrupts
- Message Dispatch
- Initialization
- C-Runtime
- Light Sensor
- Clock
- Scheduler
- Led Control
- Messaging Layer
- Packet Layer
- Radio Interface
- Routing Application
- Radio Byte Encoder

Scheduler: 144 Bytes code
Totals: 3430 Bytes code
226 Bytes data

http://tinyos.millennium.berkeley.edu
Power Breakdown…

<table>
<thead>
<tr>
<th></th>
<th>Active</th>
<th>Idle</th>
<th>Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>5 mA</td>
<td>2 mA</td>
<td>5 μA</td>
</tr>
<tr>
<td>Radio</td>
<td>7 mA (TX)</td>
<td>4.5 mA (RX)</td>
<td>5 μA</td>
</tr>
<tr>
<td>EE-Prom</td>
<td>3 mA</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LED’s</td>
<td>4 mA</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Photo Diode</td>
<td>200 μA</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Temperature</td>
<td>200 μA</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- But what does this mean?
  - Lithium Battery runs for 35 hours at peak load and years at minimum load!
    - That’s three orders of magnitude difference!
  - A one byte transmission uses the same energy as approx 11000 cycles of computation.

Panasonic CR2354 560 mAh

http://tinyos.millennium.berkeley.edu
Time Breakdown...

<table>
<thead>
<tr>
<th>Components</th>
<th>Packet reception work breakdown</th>
<th>CPU Utilization</th>
<th>Energy (nj/Bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>0.05%</td>
<td>0.20%</td>
<td>0.33</td>
</tr>
<tr>
<td>Packet</td>
<td>1.12%</td>
<td>0.51%</td>
<td>7.58</td>
</tr>
<tr>
<td>Ratio handler</td>
<td>26.87%</td>
<td>12.16%</td>
<td>182.38</td>
</tr>
<tr>
<td>Radio decode thread</td>
<td>5.48%</td>
<td>2.48%</td>
<td>37.2</td>
</tr>
<tr>
<td>RFM</td>
<td>66.48%</td>
<td>30.08%</td>
<td>451.17</td>
</tr>
<tr>
<td>Radio Reception</td>
<td>-</td>
<td>-</td>
<td>1350</td>
</tr>
<tr>
<td>Idle</td>
<td>-</td>
<td>54.75%</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>100.00%</td>
<td>100.00%</td>
<td>2028.66</td>
</tr>
</tbody>
</table>

- 50 cycle thread overhead (6 byte copies)
- 10 cycle event overhead (1.25 byte copies)

http://tinyos.millennium.berkeley.edu
How well did we meet the requirements?

✓ Capable of fine grained concurrency
✓ Small physical size
✓ Efficient Resource Utilization
✓ Highly Modular

http://tinyos.millennium.berkeley.edu
Conclusions

- People are working on shrinking sensors and communication, we need to focus on what brings them together
- TinyOS is a highly modular software environment tailored to the requirements of Network Sensors, stressing efficiency, modularity and concurrency
- We now have a whole new set of tradeoffs need to investigate

http://tinyos.millennium.berkeley.edu
Hardware Kits

- Two Board Sandwich
  - Main CPU board with Radio Communication
  - Secondary Sensor Board
- Allows for expansion and customization
- Current sensors include: Acceleration, Magnetic Field, Temperature, Pressure, Humidity, Light, and RF Signal Strength
- Can control RF transmission strength & Sense Reception Strength

http://tinyos.millennium.berkeley.edu
How to get more information:

http://tinyos.millennium.berkeley.edu
Real time operating systems

<table>
<thead>
<tr>
<th>Name</th>
<th>Code Size</th>
<th>Target CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>pOSEK</td>
<td>2K</td>
<td>Microcontrollers</td>
</tr>
<tr>
<td>pSOSSystem</td>
<td></td>
<td>PII-&gt;ARM Thumb</td>
</tr>
<tr>
<td>VxWorks</td>
<td>286K</td>
<td>Pentium -&gt; Strong ARM</td>
</tr>
<tr>
<td>QNX Nutrino</td>
<td>&gt;100K</td>
<td>Pentium II -&gt; NEC</td>
</tr>
<tr>
<td>QNX RealTime</td>
<td>100K</td>
<td>Pentium II -&gt; SH4</td>
</tr>
<tr>
<td>OS-9</td>
<td></td>
<td>Pentium -&gt; SH4</td>
</tr>
<tr>
<td>Chorus OS</td>
<td>10K</td>
<td>Pentium -&gt; Strong ARM</td>
</tr>
<tr>
<td>ARIEL</td>
<td>19K</td>
<td>SH2, ARM Thumb</td>
</tr>
<tr>
<td>Creem</td>
<td>560 bytes</td>
<td>ATMEL 8051</td>
</tr>
</tbody>
</table>

- QNX context switch = 2400 cycles on x86
- pOSEK context switch > 40 µs
- Creem -> no preemption
TOS Module AM

Accepts

char AM_SEND_MSG(char addr, char type, char* data);
void AM_POWER(char mode);
char AM_INIT();

Signals

char AM_MSG_REC(char type, char* data);
char AM_MSG_SEND_DONE(char success);

Handles

char AM_TX_PACKET_DONE(char success);
char AM_RX_PACKET_DONE(char* packet);

Uses

char AM_SUB_TX_PACKET(char* data);
void AM_SUB_POWER(char mode);
char AM_SUB_INIT();

http://tinyos.millenium.berkeley.edu
Ad hoc networking

- Each node needs to determine its parent and its depth in the tree.
- Each node broadcasts out <identity, depth, data> when parent is known.
- At start, Base Station knows it is at depth 0.
  - It sends out <Base ID, 0, **>.
- Individuals listen for minimum depth parent.

http://tinyos.millennium.berkeley.edu
Easy Migration of the Hardware Software Boundary

- TinyOS component models hardware abstractions in software
- Component model allows migration of software components into hardware
- Example:
  - Bit level radio processing component could be implemented as specialized FIFO with complex pattern matching
  - Could reduce CPU utilization while sending by more than 50%

http://tinyos.millennium.berkeley.edu
Sample tradeoffs

Radio Receive Power Optimizations

Optimization Type

<table>
<thead>
<tr>
<th>Average Power Consumption (mA)</th>
<th>None</th>
<th>Micro</th>
<th>Macro</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender Overhead</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Receiver Radio</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Receiver CPU</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Battery Lifetime for sensor reporting every minute

<table>
<thead>
<tr>
<th>Duty Cycle</th>
<th>Estimated Battery Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Time Listen</td>
<td>3 Days</td>
</tr>
<tr>
<td>Full Time Low_Power Listen</td>
<td>6.54 Days</td>
</tr>
<tr>
<td>Periodic Multi-Hop Listening</td>
<td>65 Days</td>
</tr>
<tr>
<td>No Listen (no Multi-hop)</td>
<td>Years</td>
</tr>
</tbody>
</table>