t-Kernel: Providing Reliable OS Support to Wireless Sensor Networks

SANDEEP KONDUBHATLA
Introduction

• What is the paper all about???
• Challenges in developing a reliable large-scale wireless sensor network (WSN):
  Resource Constraints
  Energy Budget
  Demanding Application requirements
Introduction

• Features that can improve the reliability of WSN software:
  
  OS protection
  
  Virtual Memory
  
  Preemptive Scheduling
Introduction

- Problem with traditional softwares:
  Lack of hardware support for privileged execution and address translation
- Need to design a new software Kernel...
t-kernel
Introduction

- t-kernel can perform extensive code modification at load time
- The modified code and the OS work in a collaborative way supporting the three essential features used to improve the reliability of the WSN systems
Introduction

• t-kernel’s performance is evaluated by measuring the **overhead** and **execution speed**.

• The t-kernel enhances developers’ ability to design reliable and sophisticated sensor networks, and includes several new design techniques.
Motivation

• Scenario 1: OS Control

• Project Name: ExScal - Extreme Scaling in Wireless Sensor Networking

• In December 2004, the OSU DARPA-NEST team headed by Anish Arora completed the first demonstration and experiments of ExScal. This demonstration covered an area 1.3km by 300m with about 1000 sensor nodes and around 200 backbone nodes making it the largest wireless sensor network assembled to date
Motivation

• ExScal's demonstration is also the largest ad hoc 802.11 network thus far created

• The motivation for the DARPA Extreme Scaling project, code-named "ExScal," was to investigate the challenges in scaling to a network of 10,000 sensor nodes.

• http://cast.cse.ohio-state.edu/exscal/
Motivation

• Such a network can be deployed by aircraft in an inaccessible area and operate in an unattended manner

• For maintenance, it is essential to guarantee that the node always responds to wireless control requests, such as reprogramming commands
Motivation

• However, such a guarantee turns out to be difficult to accomplish because the OS and application share the same memory space and have the same privilege

• A faulty application can grab the CPU and prevent the OS from processing any control requests

• Scenario 2: Memory Shortage
Assumptions

- Assumption R: Reprogrammable
- Assumption E: External nonvolatile storage
- Assumption M: Memory
- REM Computers
Figure 1. The host node and \textit{t-kernel}
Design Overview

- CPU Control and OS Data Integrity are two aspects of OS protection
- The Naturalization guarantees OS control
-Traditionally, the CPU control is guaranteed by privilege support and clock interrupts
Design Overview

• However, many micro controllers used by sensor nodes do not have privilege support.

• The application can disable interrupts and occupy the CPU for an arbitrary long time.

• The t-kernel’s approach is to modify the application program so that naturalized program yields CPU to kernel frequently.
Design Overview

• The t-kernel provides a virtual memory space much larger than the physical data memory to the application

• The virtual-physical memory address translation, boundary check, and memory swapping are handled by natin without virtual memory and exception hardware
Design Overview

• It is a challenge to support fast virtual memory accesses without hardware support
• Most virtual memory systems provide flat memory space, and virtual addresses in the space are translated by the virtual memory hardware
• REM computers do not assume such hardware
Design Overview

• To efficiently support a large virtual address space, the t-kernel defines three types of memory accesses to these areas to make the common case fast

• Heap memory, Stack memory, Physical address sensitive memory

• The virtual memory in the t-kernel is known as Differentiated Virtual Memory (DVM)
Implementation

- The t-kernel is implemented for the ATmega128L microcontroller, which is broadly used in many WSN platforms.
# Implementation

<table>
<thead>
<tr>
<th>Hardware parameters</th>
<th>Data RAM</th>
<th>External flash</th>
<th>Program memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>4KB</td>
<td>512KB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>128KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OS parameters</th>
<th>Virtual memory</th>
<th>64KB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data frame</td>
<td>64 frames</td>
</tr>
<tr>
<td></td>
<td>Look-aside buffer</td>
<td>64 entries</td>
</tr>
<tr>
<td></td>
<td>2-associative VPC</td>
<td>256 entries</td>
</tr>
<tr>
<td></td>
<td>System stack</td>
<td>1KB</td>
</tr>
<tr>
<td></td>
<td>I/O buffer</td>
<td>516 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Implementation details</th>
<th>Code size (source)</th>
<th>10,056 lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Code (binary)</td>
<td>28,950 bytes</td>
</tr>
</tbody>
</table>

Table 1. System characteristics
Performance Evaluation

- Performance results of t-kernel on the Berkeley MICA2 platform are presented.
- The overhead of Kernel transitions and that of DVM are calculated.
Performance Evaluation

- To study the overhead of kernel transitions, programs are written in assembly
- The execution of the kernel transition was found to be around 20
- Overall CPU utilization
Implementation

Figure 13. Execution time of insertion sorting