Technische Universität Braunschweig

INSTITUT FÜR BETRIEBSSYSTEME UND RECHNERVERBUND

Prof. Dr.-Ing. L. Wolf | Prof. Dr. S. Fekete

IBR

Recent Topics in Computer Networking: LiteOS

Martin Wegner

January 12th, 2012

Outline

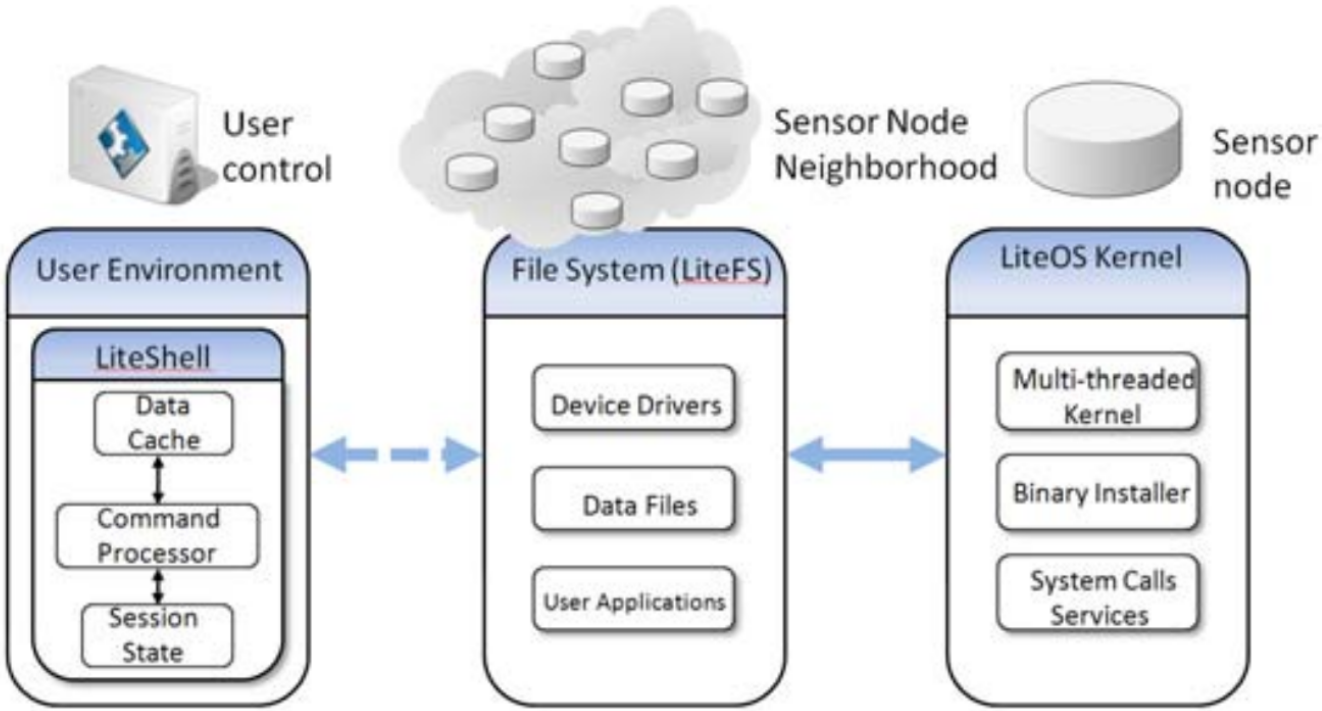
Design goals of LiteOS

- Provide Unix-like abstraction from wireless sensor nodes
  - Shell
  - Hierarchical filesystem
  - **Programming:** Threads, C
- Small resource requirements
  - Designed for MicaZ nodes
  - 8 MHz CPU, 128 KB program flash, 4 KB RAM

Assumptions on...

- *topology*: wireless sensor nodes with (powerful) computer as "base station"
- *environment*: trusted, no authentication implemented

Components: Overview



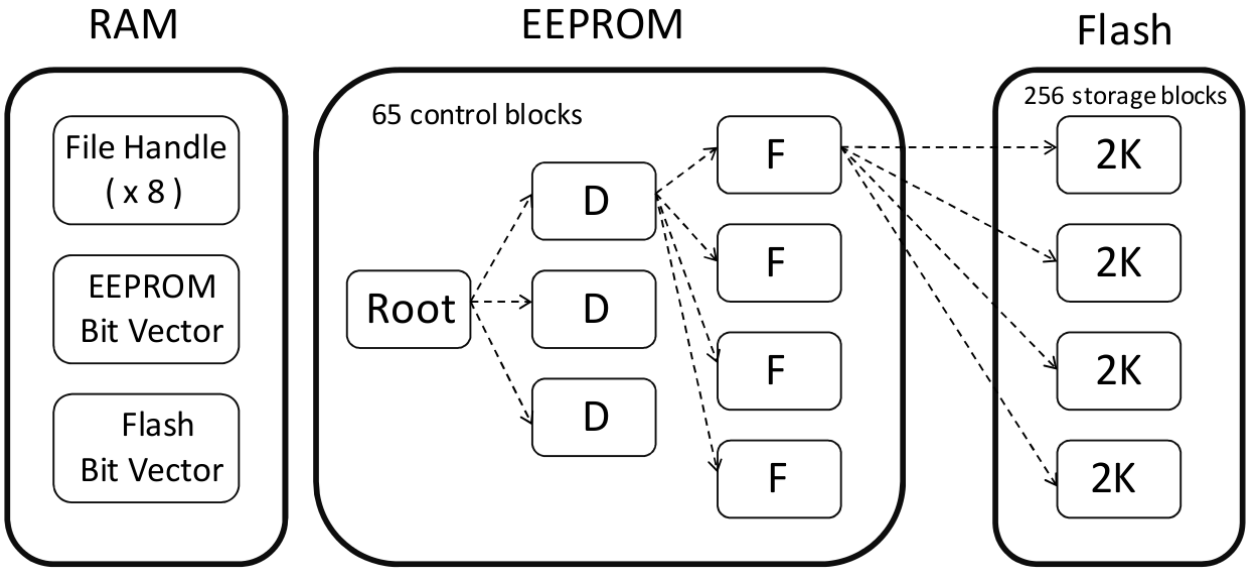
- LiteShell
- LiteFS
- LiteOS kernel

LiteShell

- Runs on "base station"
  - Shell state is *only* on base station
  - Commands translated to internal messages
- Provided commands for...
  - *dirs/files*: `ls`, `mkdir`, `cd`, `cp`, etc.
  - *processes*: `ps`, `kill`, `exec`
  - *debugging*: `breakpoint`, `continue`, `snapshot`, `restore`, etc.
  - *environment* (history, etc.) and *device commands* (`./<deviceName`)

LiteFS

- Hierarchical filesystem with directories and files (applications, devices, data)
- Each node's filesystem "mounted" to a root directory on base station
- Access control: user levels 1 - 3 with permissions `rwx` each
- Internal representation:



LiteFS API

Common API for LiteFS exists:

- `fopen()`, `fclose()`, `fread()`, `fwrite()`,...
- `fcreatedir()`, `fcopy()`, `fmove()`,...
- `fsearch()`, `finfonode()`,...

LiteOS kernel

- Supports multithreading
- *Priority-based* or *round-robin* scheduling
- Dynamic (un)loading of applications
- Event handling
  - internal events: e. g. sending of packet succeeded ⇒ threads
  - external events: e. g. packet received ⇒ callbacks
- Dynamic memory allocation (`malloc()`, `free()`)

LiteOS kernel: Dynamic (un)loading

- (Un)Loading of applications involves relocating memory access (start address, allocated memory address, stack top)
- Two approaches:
  1. Application's source code available:
     Recompile application with new memory locations
  2. Source not available:
     - Based on application's assembler derive a *model* describing how addresses change when relocated
     - Upload application and model to node

LiteOS: Event handling example [CASH08]

```
 1  void application() {
 2    bool wakeup = FALSE;
 3    uint8_t currentThread;
 4    currentThread = getCurrentThreadIndex();
 5    registerRadioEvent( MYPORT, msg, length, packetReceived
          );
 6    sleepThread( T_timeout );
 7    unregisterRadioEvent( MYPORT );
 8    if( wakeup == TRUE ) { /* ... */ }
 9    else { /* ... */ }
10  }
11
12  void packetReceived() {
13    _atomic_start();
14    wakeup = TRUE;
15    wakeupThread( currentThread );
16    _atomic_end();
17  }
```

Summary

- Unix-like environment with shell, hierarchical “network filesystem”, C programming
- Multithreaded, event handling via threads or callbacks
- Small hardware requirements on nodes
- Dynamic (un)loading of applications
- Shell integrated debugging

Thank you for your attention.

Martin Wegner, m.wegner@tu-bs.de

Literature

Cao, Qing ; Abdelzaher, Tarek ; Stankovic, John ; He, Tian:

The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks.
In: *Proceedings of the 7th international conference on Information processing in sensor networks*.
Washington, DC, USA : IEEE Computer Society, 2008 (IPSN '08).
–
ISBN 978–0–7695–3157–1, 233–244