Operating Systems for Sensor Netzworks

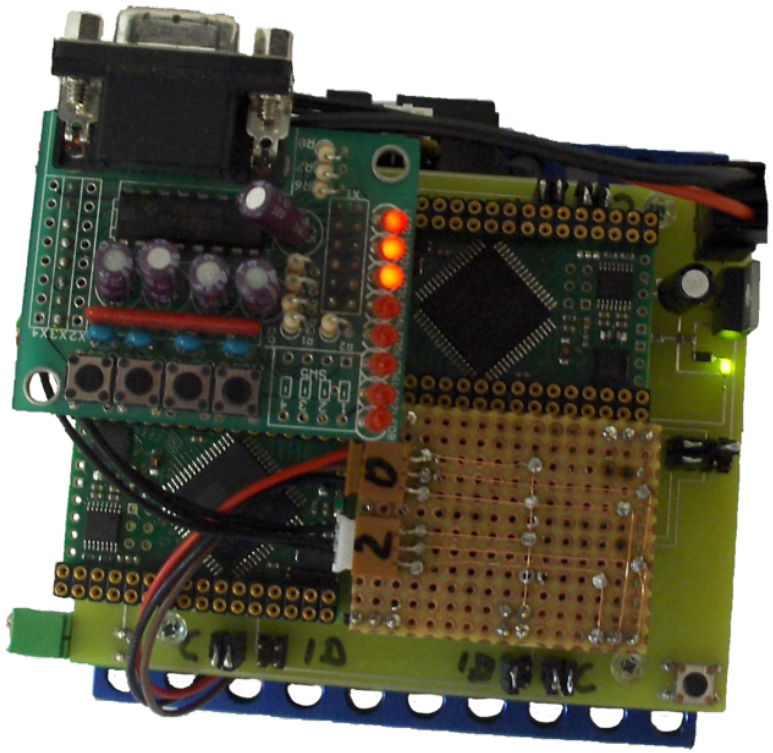
Michael Schulze

mschulze@ovgu.de

Department of Distributed Systems (IVS)
Embedded Systems and Operating Systems (EOS)
Otto-von-Guericke-Universität Magdeburg

Magdeburg, 09.07.2008

Outline



Sensor Network Applications

- Environmental monitoring
- Tracking of animals
- Inventory control (RFID)
- Home/office monitoring and automation
- Structural monitoring

- Defense industry

Hardware

- Heterogeneity
- Miniaturization
- Integration
- Energy management

Typical Hardware of a sensor node e.g. Chipcon CC2420DB

- AVR ATmega128L
- CC2420 communication modul (IEEE 802.15.4)
- Temperature sensor
- 4 LEDs
- Joystick
- Potentiometer

Motivation

Why an OS?

- Abstracts from hardware level
- Provide logical (device independent) interfaces
- Manage resources (CPU, memory, . . .)

Sensor network operating system challenges

- Extreme resource constrains (processing, memory, energy, . . .)
- Different hardware
- Event driven – reactive to sensor inputs and communication
- Concurrently data processing
- Special operating systems needed

Examples of sensor network operating systems

- Tiny OS – `http://www.tinyos.net/`
- Contiki OS – `http://www.sics.se/contiki/`
- Mantis OS – `http://mantis.cs.colorado.edu`

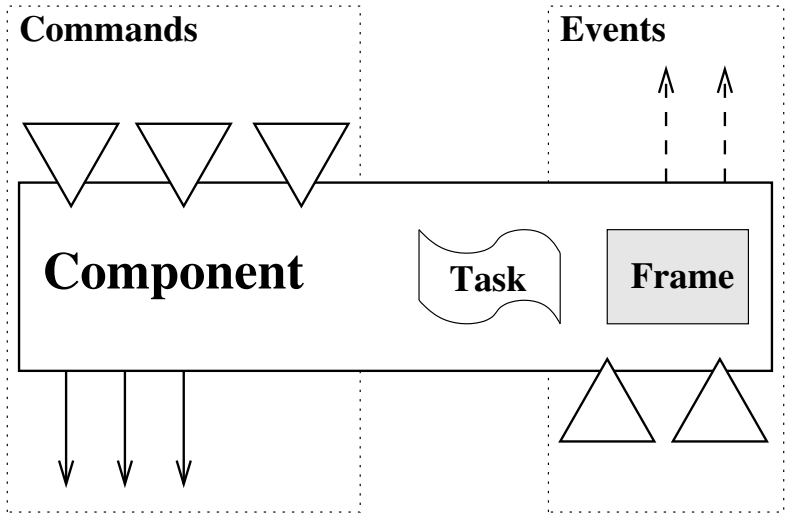
TinyOS

- Development since 2000 at the University of Berkley
- Open Source
- Objectives:
  - Minimal hardware requirements
  - Efficient parallel processing of data streams
  - Software modularity
- General structure
  - Scheduler
    - Schedules task in FIFO order
    - If task queue empty, puts the CPU into sleep mode
  - Components
- Programming language: NesC
  - C like syntax, ...
  - ...but more set up to the TinyOS model

TinyOS-Components I

Each component consists of four parts:

- Command handler
- Event handler
- Frame with static size
- one or more tasks

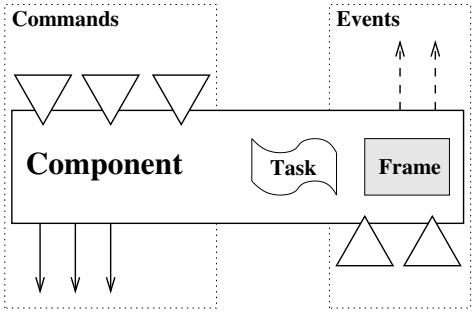


and can be realized in hardware as well as software.

TinyOS Components II

Commands

- Send by higher level components
- Return value shows failure or success



Events

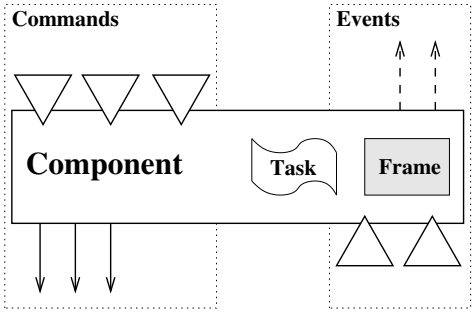
- Triggered by hardware interrupts on the lowest level
- May trigger events on higher level components

Commands and events are always yield small and fast actions without waiting times.

TinyOS Components III

Frames

- Data memory of the component
- Size is static and known at compile time



Tasks

- Doing the work
- Started by events, commands or other tasks of same component
- One at a time, only events may interrupt shortly
- Run to completion
  - \+ One stack is required for all tasks
  - – Blocking or busy waiting not allowed

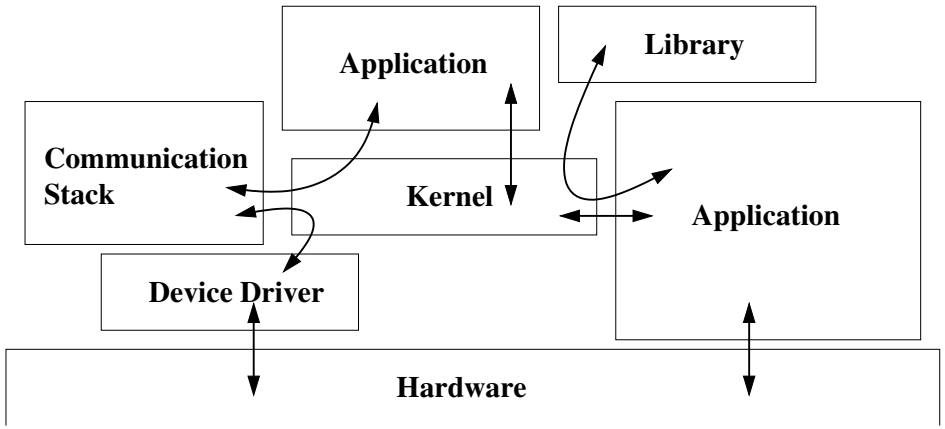
Contiki OS

- Adam Dunkel (Swedish Institute of Computer Science)
- Open Source
- Since March 2003 ported on 20 platforms
- System consists of a kernel and libraries
- Kernel:
  - Based on processes and events
  - Schedules and dispatches events to running processes
  - Provide CPU-multiplexing and message passing
  - Only platform independent code
  - Not multi-thread
- Libraries:
  - On top of kernel
  - Provide additional functionality often platform dependent code

Contiki OS–Libraries

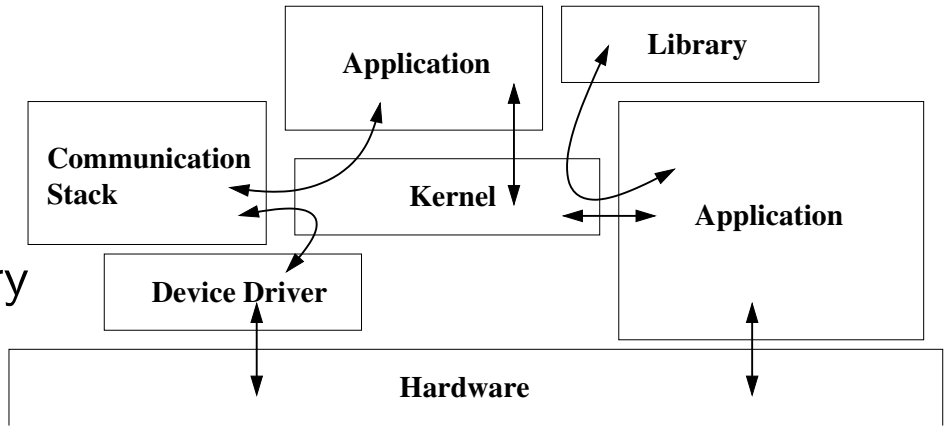
- Optionally linked to programs
- Run-time loading and linking possible
- Full access to underlying hardware
- Contain all platform dependent code

- Examples:
  - Multi-threading
  - Device driver
  - Network stack (TCP/IP)
  - Web browser
  - GUI
  - ...



Contiki OS–Processes

- Defined by an event handler function and . . .
- . . . an optional poll handler function
- Not preemptable and run to completion
- Fast context switches
- Extensions
  - Proto-threads
  - Multi-threading as a library



Contiki OS–Events

- Processes use events as a mean for IPC
- Events may be send to one or all processes

- Synchron:
  - Target process is immediately scheduled
  - After finishing, the caller gets control back
- Asynchron:
  - Event is queued
  - Target process is scheduled later

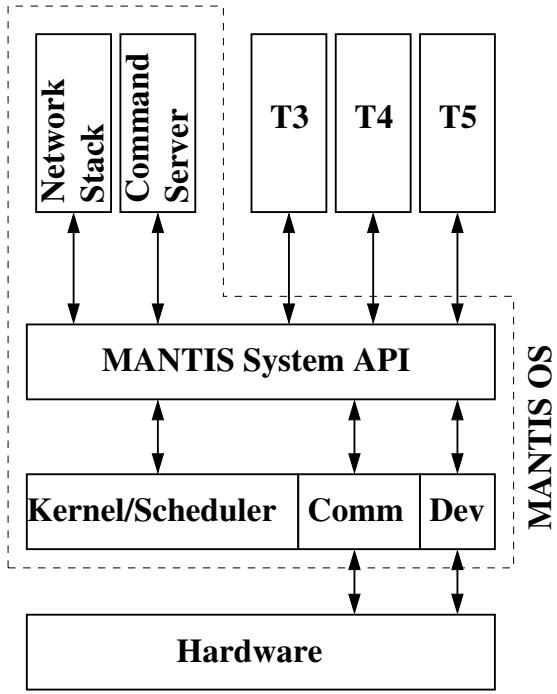
- Energy management has to be done by the application itself
  - Kernel provides only the fill level of the event queue

MANTIS OS – MultimodAl NeTworks of In-situ Sensors

- Development since 2003 at the University of Colorado
- Open Source

- Objective: as simple as possible use and programming

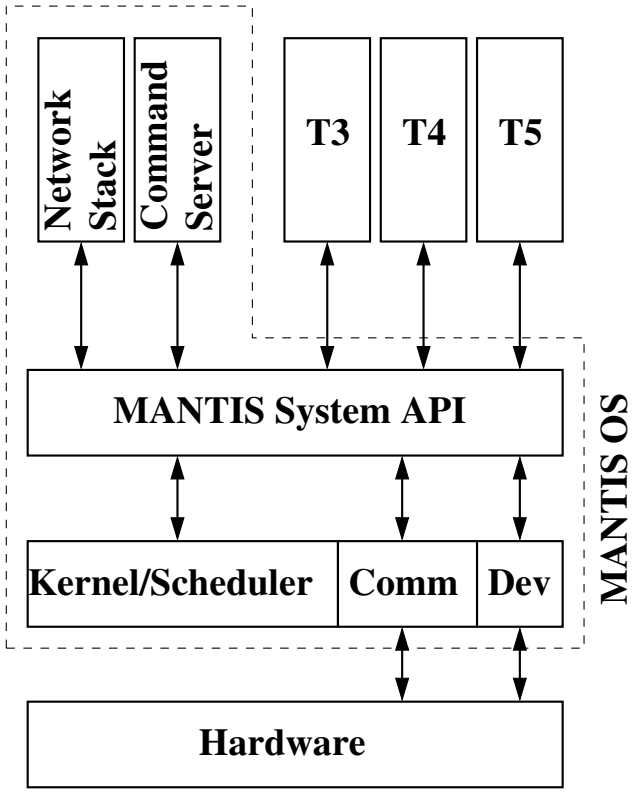
- Features:
  - Classical layered architecture
  - Traditional stack based preemptive multi-threading model
  - I/O synchronisation via MUTEX
  - Network stack
  - Device driver
  - Kernel and API plain C



MANTIS OS

Threads

- Subset of POSIX-Threads
- Kernel manages RAM dynamically and allocate space on thread creation
- Threads are managed in a table with max. 12 entries
- Idle-thread is used for energy management if all other threads are blocked



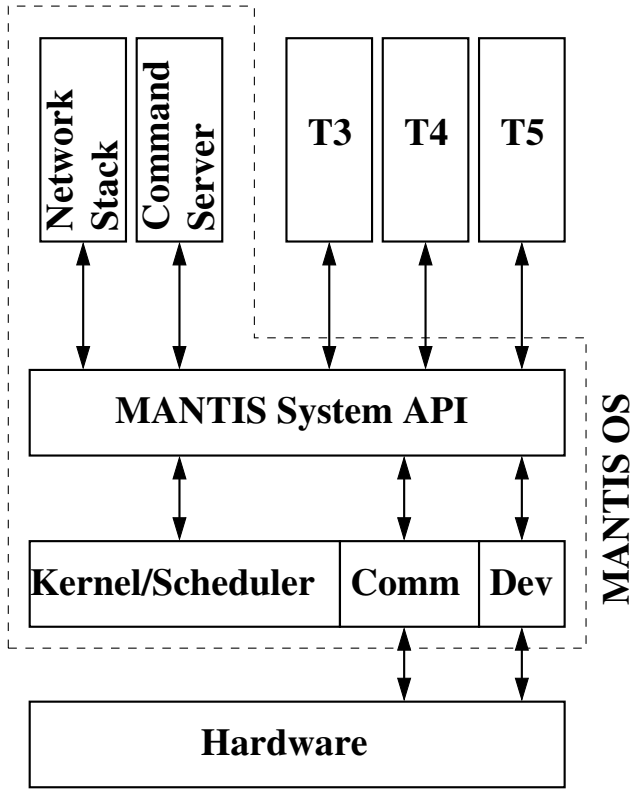
Scheduler

- UNIX-Style
- Priority based round-robin thread scheduling

MANTIS OS

Network Stack

- Four layer architecture:
  - Application
  - Network
  - MAC
  - Physical layer
- Layers are implemented as one or more threads
- Standard API between layers
- Flexible concept
- No TCP/IP



Comparison of the OS – I

	TinyOS	Contiki	MANTIS OS
Target Platform	Sensor node	MiniOS not only for sensor nodes	Sensor node
Structure	Components	Micro-Kernel, Libraries	Classical layer structure
Programming	nesC	C	C
Scheduling	FIFO	Event based, Proto-threads, Multi-Threading via libraries	Priority based round robin
Communication	Via self developed components	TCP/IP	Via self developed protocols
Energy saving	Scheduler	Application	Idle-thread

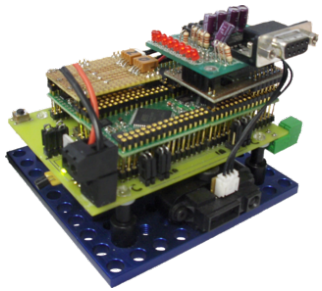
Comparison of the OS – II

- Sizes of the operating systems are dependent on actual configuration.
- Values describe different applications in combination with the operating system and should give an impression what is achievable.

	TinyOS	Contiki	MANTIS OS
RAM Usage	226 Bytes	250 Bytes	500 Bytes
Flash Usage	3.5 kBytes	4 kBytes	14 kBytes

Conclusion

- Three typical representatives of sensor network OS presented
- There are others like FreeRTOS, SOS, ...
- Deployment application context dependent



References I

Adam Dunkels and Björn Grönvall and Thiemo Voigt.
Contiki - a lightweight and flexible operating system for tiny networked sensors.
In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

Shah Bhatti, James Carlson, Hui Dai, Jing Deng, Jeff Rose, Anmol Sheth, Brian Shucker, Charles Gruenwald, Adam Torgerson, and Richard Han.
Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms.
Mob. Netw. Appl., 10(4):563–579, 2005.

CHIPCON.
`http://www.chipcon.com`.

Crossbow.
`http://www.xbow.com`.

Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali.
Protothreads: Simplifying event-driven programming of memory-constrained embedded systems.
In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006.

References II


Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister.
System architecture directions for networked sensors.
SIGPLAN Not., 35(11):93–104, 2000.


SOWNET.
`http://www.sownet.nl`.