A Practical Introduction to 6LoWPAN

Programming IPv6 Wireless Sensor Networks with Contiki

Anuj Sehgal

s.anuj@jacobs-university.de

Jacobs University Bremen, Germany

4th International Summer School on Network and Service Management (ISSNSM 2010)

JACOBS UNIVERSITY

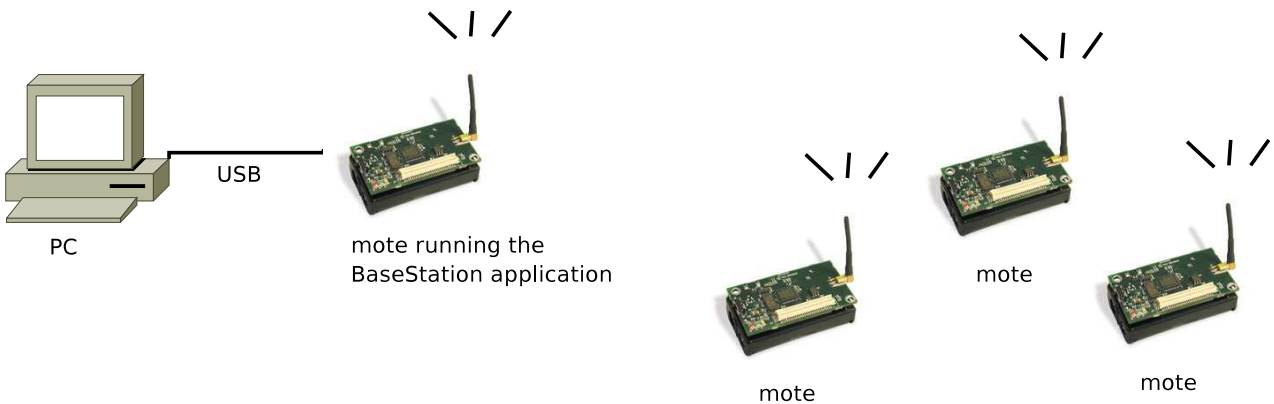
Outline

Outline

Wireless Sensor Networks

Definition

A wireless sensor network is a **wireless network** consisting of **spatially distributed autonomous devices** using **sensors** to **cooperatively monitor** physical or environmental conditions, such as temperature, pressure, gases or motion.



Wireless Sensor Networks

Characteristics

- Spatially distributed and possibly mobile
- Miniature hardware with sensors, radio and battery
- Long-term large area deployments for unattended operation
- Limited power and disruptive communications

Applications

- Environmental monitoring (seismic detection)
- Health and medical systems (patient monitoring)
- Inventory tracking and logistics (shipping containers)
- Smart spaces (home/office scenarios)
- Smart grids

Wireless Sensor Networks

Standards and Specifications

- **6LoWPAN** - IPv6 networking for embedded devices, over IEEE 802.15.4
- **ZigBee** - networking specification for embedded devices, also over IEEE 802.15.4
- **IEEE 1451** - standardized smart sensor devices
- **EnOcean** - wireless energy harvesting for building automation systems
- **WirelessHART** - designed for industrial applications like process monitoring and control

IEEE 802.15.4-2003 Standard

Overview

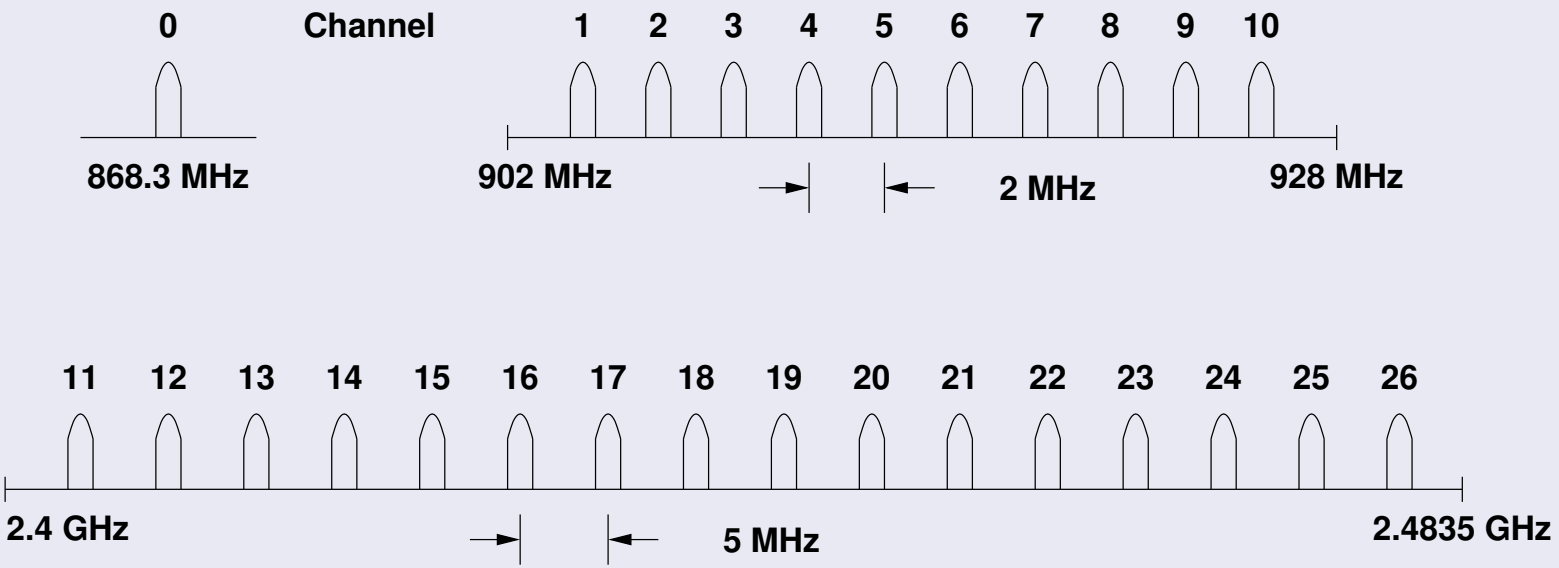
IEEE 802.15.4 is a standard which specifies the **physical layer** and **media access control** for **low-rate wireless personal area networks** (LR-WPANs). Basis for ZigBee, WirelessHART and 6LoWPAN.

Characteristics

- 20-250 kbps (2.4 GHz ISM band)
- Direct Sequence Spread Spectrum (DSSS)
- CSMA-CA medium access control
- Link encryption (AES) (no key management)
- Full / reduced function devices

IEEE 802.15.4 Radio Characteristics

Frequencies and Data Rates



Frequency	Channels	Region	Data Rate	Baud Rate
868-868.6 MHz	0	Europe	20 kbit/s	20 kBaud
902-928 MHz	1-10	USA	40 kbit/s	40 kBaud
2400-2483.5 MHz	11-26	global	250 kbit/s	62.5 kBaud

IEEE 802.15.4 Device Classes

Full Function Device (FFD)

- Any topology
- PAN coordinator capable
- Talks to any other device
- Implements complete protocol set

Reduced Function Device (RFD)

- Reduced protocol set
- Very simple implementation
- Cannot become a PAN coordinator
- Limited to leaf nodes in more complex topologies

IEEE 802.15.4 Network Elements

Device

An RFD or FFD implementation containing an IEEE 802.15.4 medium access control and physical interface to the wireless medium.

Coordinator

An FFD with network device functionality that provides coordination and other services to the network.

PAN Coordinator

A coordinator that is the principal controller of the PAN. A network has exactly one PAN coordinator.

IEEE 802.15.4 Frame Formats

General Frame Format

octets: 2	1	0/2	0/2/8	0/2	0/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	Frame sequence check

bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame type	Security enabled	Frame pending	Ack. requested	Intra PAN	Reserved	Dst addr mode	Reserved	Src addr mode

- 16-bit "short" addresses (unique within a PAN)
- Optional 16-bit source / destination PAN identifiers
- max. frame size 127 octets; max. frame header 25 octets

IEEE 802.15.4 Frame Types

Beacon Frames

- Broadcast by the coordinator to organize the network

Command Frames

- Used for association, disassociation, data and beacon requests, conflict notification and etc.

Data Frames

- Carrying user data — this is what we are interested in

IEEE 802.15.4 Media Access Control

Carrier Sense Multiple Access / Collision Avoidance

- Wait until the channel is idle
- Once the channel is free, start sending data frame after some random back-off interval
- Receiver acknowledges the correct reception of a data frame
- If the sender does not receive an acknowledgement, retry the data transmission

IEEE 802.15.4 Security

Security Services

Security Suite	Description
Null	No security (default)
AES-CTR	Encryption only, CTR Mode
AES-CBC-MAC-128/64/32	128/64/32 bit MAC
AES-CCM-128/64/32	Encryption and 128/64/32 bit MAC

- Key management must be provided by higher layers
- Implementations must support AES-CCM-64 and Null

6LoWPAN: Motivation

Benefits of IP over 802.15.4

- The pervasive nature of IP networks allows use of existing infrastructure.
- IP-based technologies already exist, are well-known, and proven to be working.
- Open and freely available specifications vs. closed proprietary solutions.
- Tools for diagnostics, management, and commissioning of IP networks already exist.
- IP-based devices can be connected readily to other IP-based networks, without the need for intermediate entities like translation gateways or proxies.

6LoWPAN: Challenges

Header Size Calculation

- IPv6 header is 40 octets, UDP header is 8 octets
- 802.15.4 MAC header can be up to 25 octets (null security) or 25+21=46 octets (AES-CCM-128)
- With the 802.15.4 frame size of 127 octets, we have only following space left for application data!
  - 127-25-40-8 = 54 octets (null security)
  - 127-46-40-8 = 33 octets (AES-CCM-128)

IPv6 MTU Requirements

- IPv6 requires that links support an MTU of 1280 octets
- Link-layer fragmentation / reassembly is needed

6LoWPAN: Overview

Overview

- The 6LowPAN protocol is an adaptation layer allowing to transport IPv6 packets over 802.15.4 links
- Uses 802.15.4 in unslotted CSMA/CA mode (strongly suggests beacons for link-layer device discovery)
- Based on IEEE standard 802.15.4-2003
- Fragmentation / reassembly of IPv6 packets
- Compression of IPv6 and UDP/ICMP headers
- Mesh routing support (mesh under)
- Low processing / storage costs

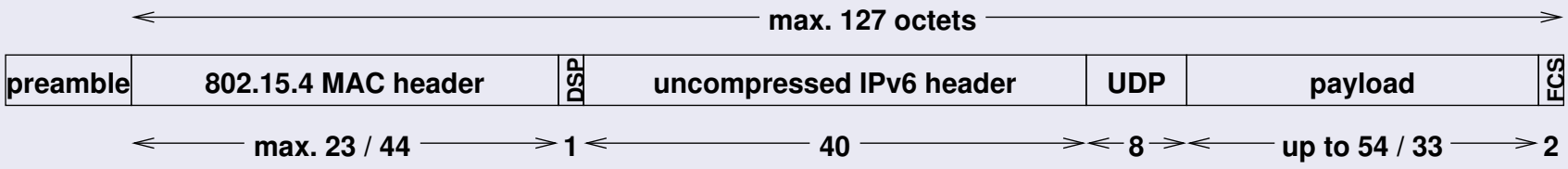
6LoWPAN: Dispatch Codes

- All LoWPAN encapsulated datagrams are prefixed by an encapsulation header stack.
- Each header in the stack starts with a header type field followed by zero or more header fields.

Bit Pattern	Description
01 000001	uncompressed IPv6 addresses
01 000010	HC1 Compressed IPv6 header
01 010000	BC0 Broadcast header
01 111111	Additional Dispatch octet follows
10 xxxxxx	Mesh routing header
11 000xxx	Fragmentation header (first)
11 100xxx	Fragmentation header (subsequent)

6LoWPAN: Frame Format

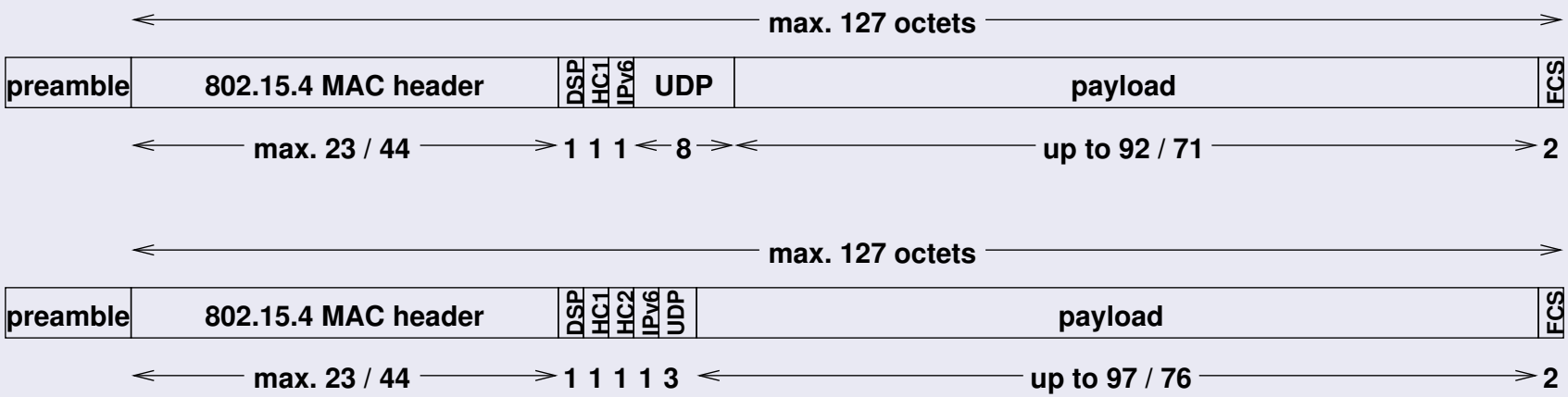
Uncompressed IPv6/UDP (worst case scenario)



- Dispatch code (01000001_2) indicates no compression
- Up to 54 / 33 octets left for payload with a max. size MAC header with null / AES-CCM-128 security
- The relationship of header information to application payload is obviously really bad

6LoWPAN: Frame Format

Compressed IPv6/UDP (best case scenario)



- Dispatch code (01000010_2) indicates HC1 compression
- HC1 compression may indicate HC2 compression follows
- This shows the maximum compression achievable for link-local addresses (does not work for global addresses)
- Any non-compressable header fields are carried after the HC1 or HC1/HC2 tags (partial compression)

6LoWPAN: Header Compression

Compression Principles (RFC 4944)

- Omit any header fields that can be calculated from the context, send the remaining fields unmodified
- Nodes do not have to maintain compression state (stateless compression)
- Support (almost) arbitrary combinations of compressed / uncompressed header fields

Note: Header compression approach is currently being revised.

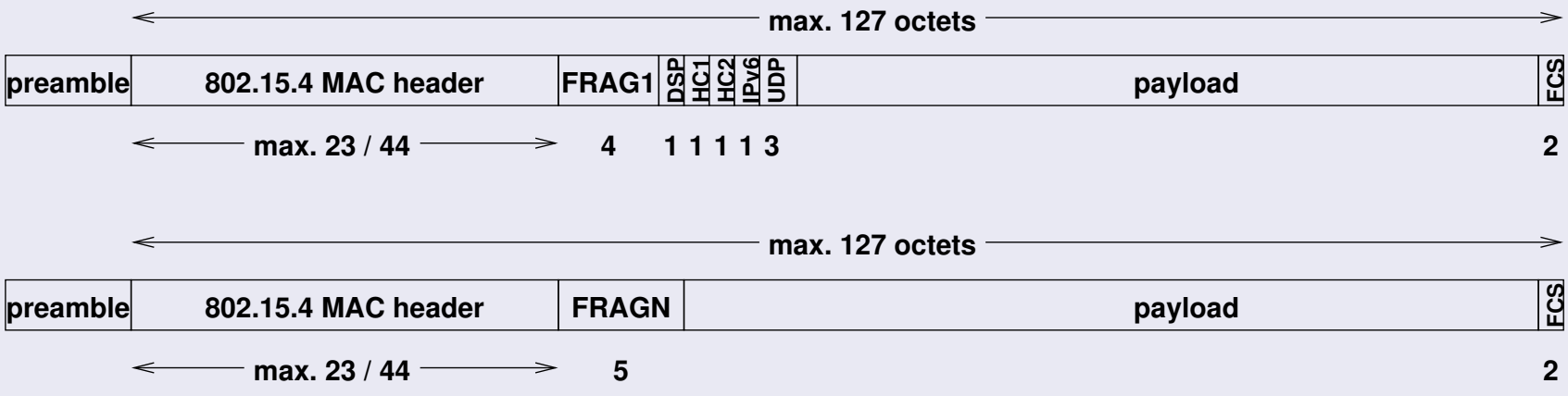
6LoWPAN: Fragmentation & Reassembly

Fragmentation Principles (RFC 4944)

- IPv6 packets to large to fit into a single 802.15.4 frame are fragmented
- A first fragment carries a header that includes the datagram size (11 bits) and a datagram tag (16 bits)
- Subsequent fragments carry a header that includes the datagram size, the datagram tag, and the offset (8 bits)
- Time limit for reassembly is 60 seconds

6LoWPAN: Fragmentation & Reassembly

Fragmentation Example (compressed link-local IPv6/UDP)



Outline

Typical WSN Hardware

Mote Class Devices

Motes are small **energy efficient** computers based on **microcontrollers** with a **wireless interface** and several **sensors**, able to operate on **battery power** for months or years.

Typical WSN Hardware

TelosB

- 8 MHz TI MSP430 (10kB RAM, 16kB ROM, 1MB flash)
- 802.15.4 2.4 GHz radio w/ antenna and integrated sensors

MicaZ

- 8 MHz Atmega128 AVR (4kB RAM, 128kB ROM, 512kB flash)
- 802.15.4 2.4 GHz radio w/ antenna
- Multiple sensors through daughter-boards

AVR Raven

- 8 MHz Atmega1284P AVR (16kB RAM, 128kB ROM)
- 8 MHz Atmega3290P AVR for LCD control
- 802.15.4 2.4 GHz radio w/ antenna

Operating Model

Target Machine-code

- This model compiles a program, which uses target hardware specific APIs and libraries, to machine code. This program is burned to node memory and cannot be altered without a physical connection.
- Programs are not portable to other platforms.

Operating Systems

- This model provides a kernel on top of which user programs are executed. The kernel compiles to target-machine code, but the user program may or may not be part of the application bundle to be burned in chip memory.
- Dynamic loading and unloading of programs and services is possible. Programs are portable to other platforms.

API Programming Model

Characteristics

- Embedded devices typically programmed by using appropriate APIs and libraries
- Cross-compiling of code necessary for target architecture
- Compiled code is target processor machine code
- Programmer needs to write drivers, configurations, memory management
- Close to assembly code
- *Cumbersome and error prone!*

API Programming Model

Example

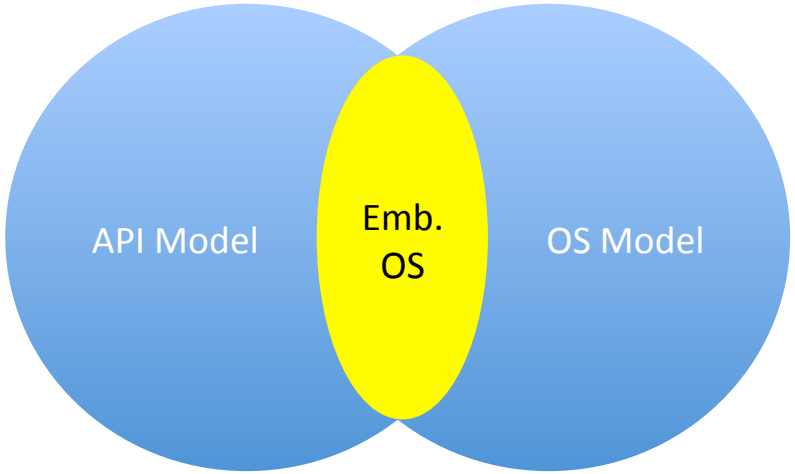
```
void main(){

   ADCSR |= (1<<ADPS2);
   ADCSR |= (1<<ADFR);
   ADMUX = currentChannel;
   ADCSR |= (1<<ADIE);
   ADCSR |= (1<<ADEN);
   ADCSR |= (1<<ADSC);

   ADC_read();
}
```

Operating System Model

Characteristics

- Kernel
- Memory Management
- Services and APIs
- Program Execution
- Hardware Drivers
- Interaction
- *Abstraction*

Embedded Operating System Model



Characteristics

- Kernel and Memory Management
- Drivers, Services and APIs
- Program Execution and Interaction
- Cross Compilation Required
- Burns an OS + application bundle

Outline

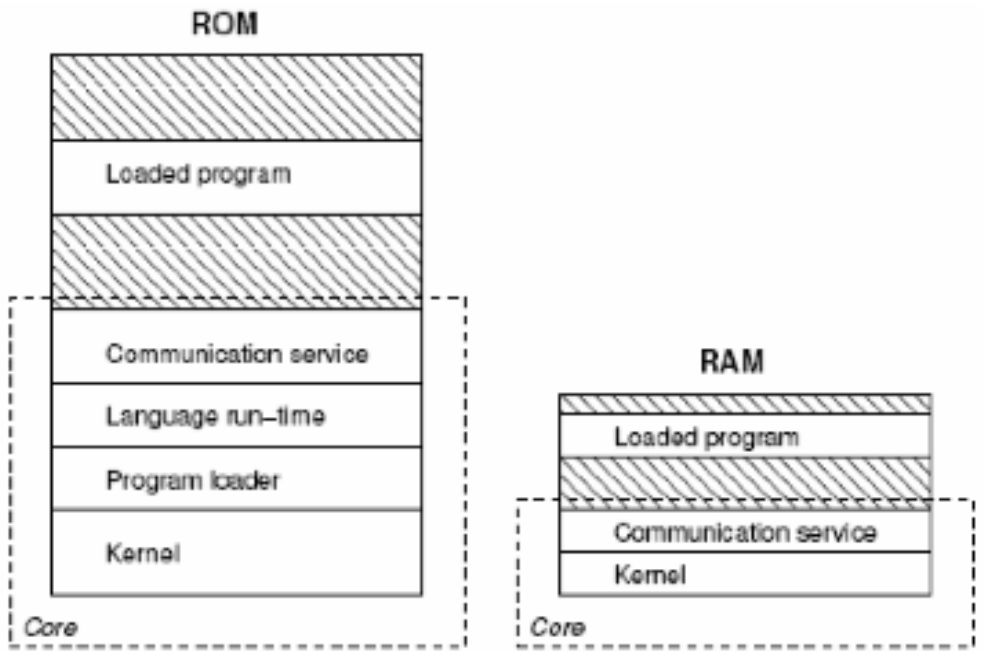
About Contiki

- An OS for sensor nodes
- Ported to many hardware platforms
  - MSP430, AVR, HC12, Z80, x86
- Plenty of hardware drivers
- Network communication stack (including IP)
- Event driven kernel
- Protothreads

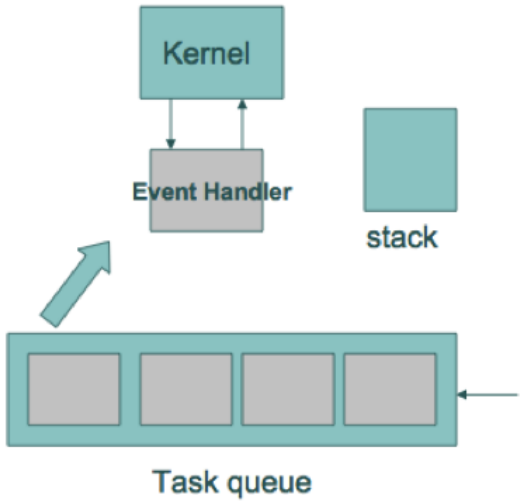
System Overview

Overview

- Contiki consists of:
  - Kernel, libraries, program loader, set of processes
- Dynamically loadable services and applications
- Inter-process communication through events



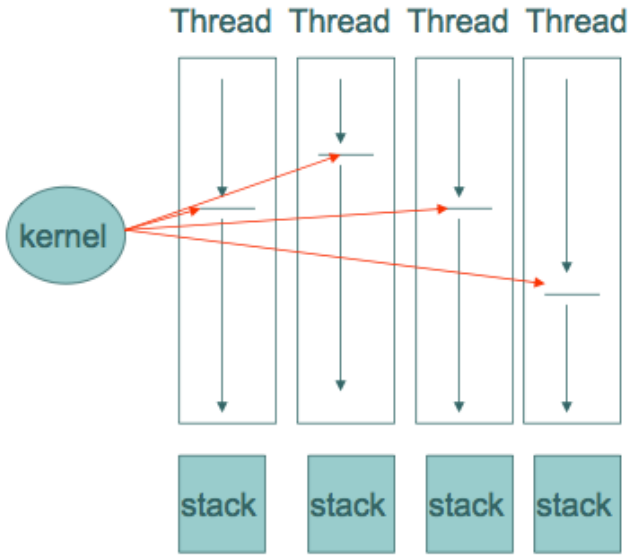
Traditional Kernel Architectures



Event Driven

- Processes do not run without events
- Single event can grab CPU resources
- Suitable for resource constrained applications and reactive systems

Traditional Kernel Architectures



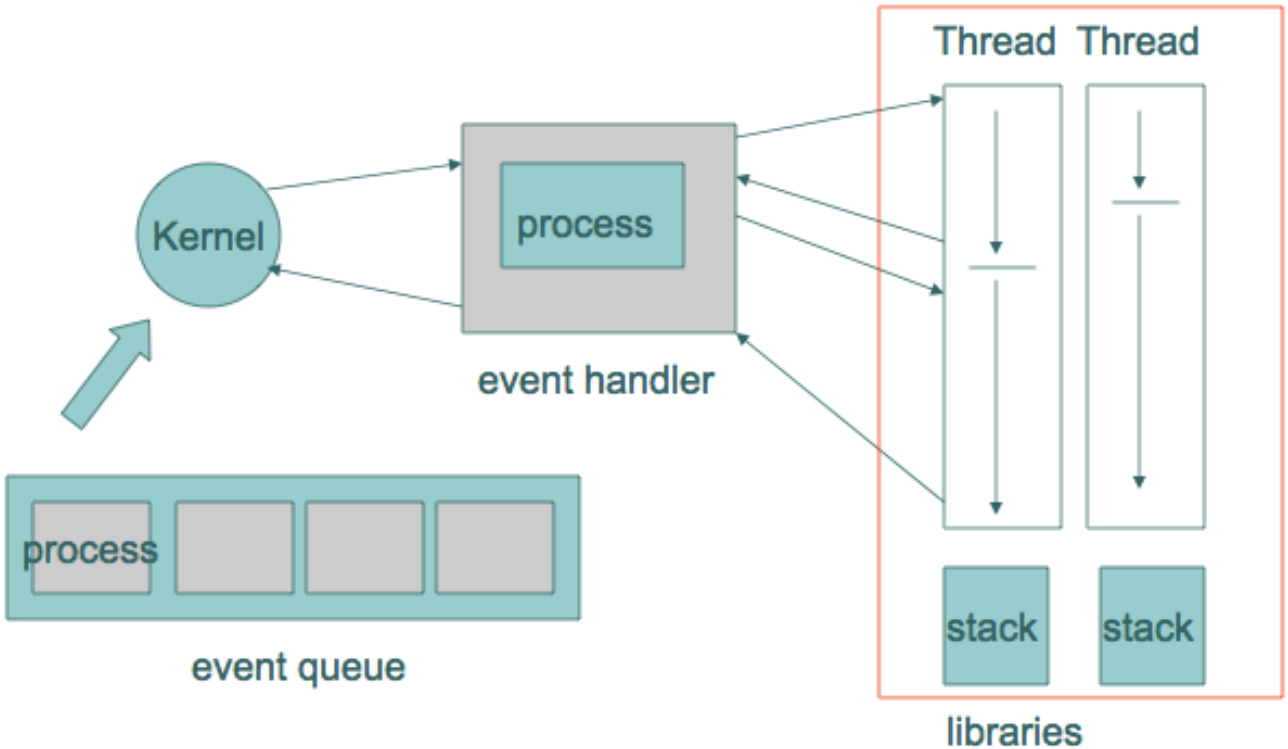
Multi-threaded

- Thread runs till blocking statement
- Preemption
- Large memory usage (threads need stacks)
- Race conditions
- Suitable for long running computations

Contiki Kernel Architecture

Contiki Kernel

- Contiki implements *Optional Multitasking*
- Protothreads – stackless small memory thread design
- Blocking and preemption on top of event-based kernel



Communication Support

Contiki Networking

- Contiki is a WSN OS and provides good communication support
- The µIP stack provides limited IP communications:
  - ARP, IP, ICMP, UDP
- IPv6 support built into µIPv6

Outline

Environment Setup

Virtual Machine

- Recommended: Use the VMWare Virtual Machine provided since it has all packages pre-installed.
- Alternate: Download the latest version from http://www.sics.se/contiki/

Programming Exercises

Concepts

- Cross compilation
- Timers and Analog-Digital-Conversions
- Hardware interaction (LEDs, sensors)
- Terminal output (mote-to-PC)
- IPv6 network setup
- Implementing UDP protocols
- Memory management
- Multi-threading using protothreads

Exercise 1

Provided

You are provided with a sample program that turns on and off the LEDs on your TelosB motes. The program toggles every LED within the same protothread.

Goal

Your task is to modify the program, such that, each LED blinks at its own individual schedule, independent of all the other LEDs. Since there are three LEDs available, make them blink at 1 Hz, 2 Hz and 4 Hz respectively.

Note: You may use the handout sheet for Exercise 1 to help you along the way.

Exercise 2

Provided

You are provided with a sample program that reads values from the light sensor on your TelosB motes. The program also outputs the value read from the sensor to a console over the USB connection.

Goal

Your task is to modify the program, such that, each sensor on the TelosB mote is sampled. Please note that the values returned by the sensors are raw voltage values since these are just digital representations of the analog voltages the sensor senses. It is also your task to convert these values into human readable formats. You will find helpful formulas in the handout for Exercise 2.

Note: You may use the handout sheet for Exercise 2 to help you along the way. For obtaining ADC value conversion formulas on other hardware, please refer to the datasheet of the manufacturer.

Exercise 3

Background

Having completed exercises to work with hardware devices, you are now familiar with concepts necessary to build more complex WSN systems.

Goal

Using the programs from Exercise 1 and 2 as the basis, convert your TelosB mote in to a light sensing device.The light intensity should be classified into dark, light, medium, and strong light. A mote should use the three LEDs to indicate whether it is dark (no LED turned on), there is low light (one LED turned on), medium light (two LEDs turned on) or strong light (all LED turned on).

Note: The concepts from this exercise may be used to build a system that reacts to environmental conditions and dispatches messages over a network.

Exercise 4

Goal

- The goal of this exercise is to setup an IPv6 network of motes that can be reached from your computers. Please follow the exercise as explained in the hand-outs, or as demonstrated.
- At the end of the exercise you will be able to connect to the motes from your laptops, using IPv6 networking and also be able to capture packets using Wireshark to observe the packet fragmentation as it occurs in 6lowpan networks.

Exercise 5

Provided

You are provided with a sample program that shows simple communication using the UDP protocol over IPv6 networking. The program sends a "Hello!" to the mote it connects to and responds with a simple "Hello!" to the system that connects to it.

Goal

You must modify the program such that you can send commands to each mote to turn on/off each individual LED as per requirement. You must also be able to send a command to retrieve the light sensor readings from the mote. A handout is provided to assist you with the exercise.

UDP Based Demos

Demo 1

- Command Line Interface on a mote

Demo 2

- Remote light sensor

Note: Sample code for these demos is provided.