Piles réseau économes en énergie dans les Réseaux de Capteurs sans Fil

David Hauweele

Université de Mons

25 mars 2016
Internet of Things

Internet today

- Not only interconnected computers
- Mobile Internet (smartphones, tablets)
- Interconnected objects (sensors, automation, monitoring)
Internet of Things

Internet today

- Not only interconnected computers
- Mobile Internet (smartphones, tablets)
- Interconnected objects (sensors, automation, monitoring)

Definition

- The **Internet of Things** is a concept which seeks to integrate physical objects to Internet thanks to an addressing system which can uniquely identify them.
- These objects are generally equipped with *sensors* and *actuators* in order to interact with their environment and they have *limited* processing capabilities → *smart objects*. 
Wireless Sensor Network

Features

- Wireless nodes
- Interconnected
- Common task

Application domains

- Environmental monitoring [11]
- Industrial applications [14]
- Health care [7]
- ...

⇒ No single WSN design!

Source: Automated irrigation [9]
### Typical wireless node

<table>
<thead>
<tr>
<th></th>
<th>Node</th>
<th>Smartphone</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>20 MHz</td>
<td>1000 MHz</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>8 kB</td>
<td>1 GB</td>
</tr>
<tr>
<td><strong>Lifetime</strong></td>
<td>years</td>
<td>days</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>10 €</td>
<td>200 €</td>
</tr>
</tbody>
</table>

Source: [http://zolertia.sourceforge.net](http://zolertia.sourceforge.net)
Typical wireless node

<table>
<thead>
<tr>
<th></th>
<th>Node</th>
<th>Smartphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>20 MHz</td>
<td>1000 MHz</td>
</tr>
<tr>
<td>RAM</td>
<td>8 kB</td>
<td>1 GB</td>
</tr>
<tr>
<td>Lifetime</td>
<td>years</td>
<td>days</td>
</tr>
<tr>
<td>Cost</td>
<td>10 €</td>
<td>200 €</td>
</tr>
</tbody>
</table>

Problems

- Limited resources
- Energy constraints
- Single application

Need an optimized and modular network stack

Source: http://zolertia.sourceforge.net
How can we achieve years of longevity?

- Low-power micro-controllers and radio
- Energy harvesting techniques
- *Energy aware communication protocols*
Lifetime

How can we achieve years of longevity?

- Low-power micro-controllers and radio
- Energy harvesting techniques
- **Energy aware communication protocols**

  - **Link layer:** RDC, dedicated MAC [12, 3, 6]
  - **Network/transport layer:** 6LoWPAN [10], Rime [5]
  - **Routing:** RPL [15]
  - **Application layer:** CoAP [13]
Lifetime

How can we achieve years of longevity?

- Low-power micro-controllers and radio
- Energy harvesting techniques
- *Energy aware communication protocols*
  - **Link layer**: RDC, dedicated MAC [12, 3, 6]
  - **Network/transport layer**: 6LoWPAN [10], Rime [5]
  - **Routing**: RPL [15]
  - **Application layer**: CoAP [13]

- *Energy aware task scheduling*
Radio Duty Cycle

Radio dominates node power consumption

- Transmission and reception $\approx 25$ mA
- Battery 2500 mAh $\rightarrow$ only 4 days lifetime
- Sleep mode $\approx 1\ \mu$A
Radio Duty Cycle

Radio dominates node power consumption

- Transmission and reception $\approx 25$ mA
- Battery 2500 mAh $\rightarrow$ only 4 days lifetime
- Sleep mode $\approx 1$ µA

Solution : leverage the sleep mode $\rightarrow$ Radio Duty Cycle
How to talk with a neighbor that sleeps 99% of time?

Sender

Receiver 1

Receiver 2

time estimated from last reception (phase-lock)

not the destination (limit overhearing)

no signal detected (fast sleep)

Listening

Activity detected

Transmitting

Receiving

D. Hauweele (UMons)
Multi-hop Routing

Transmit messages over longer ranges
- Low-power radio → limited range
- Nodes can act as *relay* for other nodes
- Routing → compute lowest cost paths

What is the cost of a path in WSN?
Interoperability

How to connect these wireless nodes to Internet?

- Physical and link layer → IEEE 802.15.4
- Network layer → ZigBee or network IP (IPv6)
- Upper layers → dedicated framework, CoAP or TCP/UDP/ICMP

Source: http://compixels.com
Interoperability

How to connect these wireless nodes to Internet?

- Physical and link layer → IEEE 802.15.4
- Network layer → ZigBee or network IP (IPv6)
- Upper layers → dedicated framework, CoAP or TCP/UDP/ICMP

We use IPv6 on top of the IEEE 802.15.4 standard

Source: http://compixels.com
Network layer IPv6

- Very large number of addresses \((2^{128} \approx 3.4 \times 10^{38})\)
- Directly exposed to Internet

Problems

The IPv6 network is optimized for use with efficient links (Ethernet, Wi-Fi, ...). This poses harsh contraints on the nodes used within a wireless sensor network.
Network layer IPv6

- Very large number of addresses ($2^{128} \approx 3.4 \times 10^{38}$)
- Directly exposed to Internet

Problems

The IPv6 network is optimized for use with efficient links (Ethernet, Wi-Fi, ...). This poses harsh constraints on the nodes used within a wireless sensor network.

Solution → 6LoWPAN [10]

- Adaptation layer between IPv6 and IEEE 802.15.4
- Adjust the IPv6 network to the performances of smart objects
The nodes of a wireless sensor network are programmed using specialized Real Time Operating Systems (RTOS) among which the most popular, Contiki [1] and TinyOS [2].

Example: Contiki

- Support IPv4, IPv6, 6LoWPAN, several MACs
- Lightweight network layer for IoT, RIME
- Event-driven cooperative scheduling
- Modularity
- Implemented in C
The Contiki network stack

Problems
- Limited Modularity
- Modules developed separately
- Complex code (low-level)
The Contiki network stack

Problems
- Limited Modularity
- Modules developed separately
- Complex code (low-level)

How can we optimize?
Complexity: the RIME layer

**Figure:** The RIME callgraph
Higher-level approach

Need an inside view of the RTOS

- Monitor events in the RTOS
- Record the time of each event
- Correlation between events in different modules
- Better understanding of the network stack

Two approaches:

- **Hardware**: monitor devices changes in MSPSIM [4, 8]
- **Software**: monitor the firmware (and the netstack)
Monitor device

- Reference clock
- Minimize in-firmware time
- Find exact moment when an event occurred
WiredMon and RegMon

Two implementations on MSP430 emulator MSPSIM

WiredMon

- Implemented like a real device
- 2-wires interface to monitoring device
- Quite slow
WiredMon and RegMon

Two implementations on MSP430 emulator MSPSIM

**WiredMon**
- Implemented like a real device
- 2-wires interface to monitoring device
- Quite slow

**RegMon**
- Hack of the MSP430f1611
- 4 extra memory registers
- Really fast
Speed comparison

Number of cycles (MSP430)

- RegMon: 27 cycles
- WiredMon: 1590 cycles
- UART: 1620 cycles
- UDP: 1310 cycles
Example: delayed sleep

Receiving a packet in ContikiMAC

- Radio layer set poll flag on the stack process
- Poll flag checked after currently running process relinquishes CPU
- Radio stays on while currently running process still running
Example: delayed sleep

Receiving a packet in ContikiMAC

- Radio layer set poll flag on the stack process
- Poll flag checked after currently running process relinquishes CPU
- Radio stays on while currently running process still running

Solution ⇒ cross layer optimization (RADIO/RDC layers)
Experiment

Delayed sleep experiment

- Two nodes: transmitter and receiver
- Transmitter sends one packet per second
- Receiver does some processing with process $P1$

Sleep delayed in average by half the working time of $P1$
Other optimizations

Pack short packets
- ContikiMAC imposes a lower bound on the frame size
- Short packet → padding
- Queue short packets and pack them together
- Reduce the number of transmissions

Optimistic timer
- Scheduler not aware of duty-cycle
- Trigger events at appropriate time
- Schedule timers with min/max time
- Allow the node to sleep for longer periods
Higher level approach

Monitor and visualisation tools $\rightarrow$ extract higher level model

network stack implementation (Contiki, TinyOS, ...)

extract higher level model from implementation

channel clear

channel activity

packet pending

max silence max time

radio off

max CCA

schedule next powercycle
Higher level approach

Monitor and visualisation tools → extract higher level model

network stack implementation (Contiki, TinyOS, ...)

extract higher level model from implementation

generate implementation from higher level model

Can we generate the implementation from a higher level model?
Conclusion

WSN
- RTOS complex and low-level
- Energy constraints
- Need a higher-level approach for netstack optimization

Monitor
- Hardware and software approach
- MSP430 emulation in MSPSIM
- RegMon approach really fast
- WiredMon for real nodes but slow
- Visualisation tools
Conclusion

Optimizations

- Possible optimizations derived from the visualisation tools
- Delayed sleep
- Short packets
- Optimistic timer

Further work

- Generate implementation from higher level model
- Derive implementations from DSLs
Conclusion

Internet of Things

- Also data mining, security, ...
- Still many technical challenges
- 50 billions devices in 2020

Chances are that you will be developing code for the Internet of Things in the future!
Questions?

- **E-Mail:**
  david.hauweele@umons.ac.be

- **Site:**
  http://www.hauweele.net/~gawen

- **Slides:**
Contiki.

TinyOS.

M. Buettner, G. V. Yee, E. Anderson, and R. Han.

A. Dunkels.
RIME — a lightweight layered communication stack for sensor networks.

A. Dunkels.
SICS.

E. Egbogah and A. Fapojuwo.
A survey of system architecture requirements for health care-based wireless sensor networks.


Wireless sensor networks: a survey on environmental monitoring. 

J. Polastre, J. Hill, and D. Culler. 
Versatile low power media access for wireless sensor networks. 

Z. Shelby, K. Hartke, C. Bormann, and B. Frank. 
Constrained application protocol (CoAP), draft-ietf-core-coap-13. 
IETF, 2012.

Wireless sensor networks for industrial applications. 
RPL : IPv6 routing protocol for low power and lossy networks.