Integrating Custom Hardware into Sensor Web

Maria Porcius
Carolina Fortuna
Gorazd Kandus
Mihael Mohorcic
OUTLINE

1. Introduction
2. State of the art
3. System architecture - main components
   - 3.1 Hardware support
   - 3.2 Operating system
   - 3.3 Communication protocol
   - 3.4 Application protocol
4. Case study analysis
5. Conclusions
1. INTRODUCTION

- **Wireless sensor networks - WSNs**
  - More and more present in our daily activities

- **Sensor Web**
  - Heterogeneous wireless sensors sharing information
2. STATE OF THE ART

- **Sensor nodes** constraints
  - *Hardware*: microcontroller, memory, wired and/or wireless modules, power, etc
  - *Software*: drivers, protocol stack implementations, operating system

- **Web** today
  - Web services, web applications
  - HTTP, TCP, SOAP – not for limited resources devices

- New solution needed for integration of sensor nodes into sensor web
  - Partial solutions
    - *Hardware designs*
    - *Upper layer (application) solutions*
3. SYSTEM ARCHITECTURE - MAIN COMPONENTS

- Hardware support
- Operating system
- Compatible communication protocols and stack
- Application protocol – web integration
3.1 HARDWARE SUPPORT

- **Versatile Sensor Node – VSN**
  - Developed at Communication Systems department, JSI
  - ARM Cortex-M3 32-bit microcontroller
  - 512 kb Flash, 64 kb RAM
  - Several interfaces: RTC, I2C, UART, SPI, USB..
3.1 HARDWARE SUPPORT

- **Versatile Sensor Node – VSN**

  - Radio modules operating in 300 MHz, 868 MHz and 2.4 GHz ISM frequency bands

  - Communication modules: ZigBee, Bluetooth, 2G, 3G, Ethernet, 6LoWPAN

  - Sensors: temperature, humidity, color, pH, ultrasonic transmitter and receiver, speaker, microphone, camera, GPS, luminance, etc.
### 3.2 OPERATING SYSTEM

- **Requirements for integration of VSN into Sensor Web**
  - Support for IP (provided by Contiki OS)
  - Integration with web services protocols (provided by CoAP)

- **VSN runs *Contiki* operating system**
  - OS for limited resources devices
  - Adapted for VSN board
  - Support for dynamic loading of application and services
    - Reduce the power consumption and loading time
  - Simultaneously loading of multiple communication stacks and runtime replacement of parts of the stack
  - Few stack implementations: lwIP, μIP and μIPv6 (integration into 6LoWPAN networks)
3.3 COMMUNICATION PROTOCOLS AND STACKS

- Several *standards and technologies* for WSNs
  - IEEE802.15.4, Zigbee, 6LoWPAN, Bluetooth, ISA100.11a, WirelessHART

- **IP**-based communication standards: 6LoWPAN

- **IP** approach for WSNs
  - Advantages: extensive use and improvement
  - Disadvantages: build for different set of constraints and goals
3.3 COMMUNICATION PROTOCOL AND STACKS

- Disadvantages eliminated by developments of protocol stacks for constrained devices - taking into account
  - Limited memory requirements
  - Operating system and microcontroller compatibility
  - Code portability

<table>
<thead>
<tr>
<th>Name</th>
<th>Memory</th>
<th>OS</th>
<th>Microcontroller</th>
</tr>
</thead>
<tbody>
<tr>
<td>lwIP</td>
<td>40kb ROM 20kb RAM</td>
<td>With/without OS</td>
<td>8- and 16-bit</td>
</tr>
<tr>
<td>µIP</td>
<td>6 kb ROM 1 kb RAM</td>
<td>Contiki</td>
<td>8- and 16-bit</td>
</tr>
<tr>
<td>µIPv6</td>
<td>11.5kb ROM 1.8kb RAM</td>
<td>Contiki</td>
<td>8-bit</td>
</tr>
<tr>
<td>NanoStack</td>
<td>4-8kB RAM 32-64kB ROM</td>
<td>FreeRTOS</td>
<td>16-bit</td>
</tr>
<tr>
<td>Matus</td>
<td>4kb RAM</td>
<td>TinyOS</td>
<td>8-bit</td>
</tr>
<tr>
<td>BLIP</td>
<td>4kb RAM</td>
<td>TinyOS</td>
<td>8-bit</td>
</tr>
</tbody>
</table>
3.4 APPLICATION PROTOCOL – WEB INTEGRATION

- Existing web applications mostly based on HTTP
  - HTTP – not appropriate for WSNs: synchronous, keep-alive option

- Solution: *Constrained application Protocol (CoAP)*

- **CoAP**
  - Designed by CoRE (Constrained RESTful Environments) working group for constrained networks and nodes
  - First draft from Internet Engineering Task Force (IETF)
  - Provides
    - Communication over the Internet
    - Integration of small and limited devices into embedded web
3.4 APPLICATION PROTOCOL – WEB INTEGRATION
4. CASE STUDY ANALYSIS

- *Empty parking space detection system*

- **Nodes:**
  - VSN with presence detection sensors
  - Contiki OS
  - IPv6 protocol stack
  - CoAP

- Sending periodic measurements
  - Request/reply interaction model
  - Additional subscribe/notify model

- Agent - remotely checks the availability of parking spaces
  - through web services
4. CASE STUDY ANALYSIS

- Server for storing data – inside or outside the local network
  - CoAP *maps* to HTTP – through proxy

- **Caching** - Optimize bandwidth and power consumption
  - Client and proxy level
    - Client – resources (*cache refresh after lifetime*)
    - Proxy – resources and subscriptions (deal with sleeping nodes and avoid traffic overload)
5. CONCLUSIONS

- Goal: *Integration of Custom Hardware into Sensor Web*

- State of the art

- Our approach
  - WSN of *Versatile Sensor Nodes*
  - VSN with *Contiki OS*
  - *IP* - based protocol stack implementation to provide communication
  - *CoAP* – application layer protocol for web integration

- Case study
  - Empty parking space detection system
Thank You!