Wireless sensor networks abstractions: from the middleware approach to the layered architecture of the Internet of Things

PhD defense

PhD candidate: Daniele Alessandrelli

Tutor
Prof. Giorgio Buttazzo

Supervisor:
Dr. Paolo Pagano

October the 29th, 2013
Outline

1. Background and motivation
2. Contributions
3. In-network processing in IoT-based WSNs
4. Conclusions
Outline

1. Background and motivation
2. Contributions
3. In-network processing in IoT-based WSNs
4. Conclusions
Wireless Sensor Networks

Networks of *smart* sensors and actuators, called *motes*

**Smartness:**
- local data processing
- cooperation

*Motes* are tiny, cheap, and self-powered embedded devices
Wireless Sensor Networks

Networks of *smart* sensors and actuators, called *motes*

**Smartness:**
- Local data processing
- Cooperation

*Motes* are tiny, cheap, and self-powered embedded devices

Cheap and pervasive technology for monitoring and controlling the physical world
Problem statement

The adoption of WSNs is hindered by the complexity of application development:

• many issues
  – distributed architecture, wireless communication, interaction with the physical world, etc.

• limited resources
  – computational power, memory, battery, network bandwidth, etc.
Problem statement

The adoption of WSNs is hindered by the complexity of application development:

- many issues – distributed architecture, wireless communication, interaction with the physical world, etc.
- limited resources – computational power, memory, battery, network bandwidth, etc.

Programming abstractions are needed to simplify the development of WSN applications
The traditional approach: middleware for WSNs

A **Middleware** system is **vertically-integrated** software infrastructure that manages the entire WSN and provides application developers with **high-level services**

Example: the *database-like abstraction*

```sql
SELECT SENSOR_ID, ACC_X
FROM ACCELERATION
```

<table>
<thead>
<tr>
<th>Sensor ID</th>
<th>Acc X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.3</td>
</tr>
<tr>
<td>2</td>
<td>+2.1</td>
</tr>
<tr>
<td>3</td>
<td>-0.1</td>
</tr>
<tr>
<td>4</td>
<td>+0.7</td>
</tr>
</tbody>
</table>
The middleware approach: pros and cons

Pros

Full control over the WSN

• High efficiency

• Quality of service (QoS) can be easily provided
  — Including real-time

Cons

Lack of generality

• Middleware solutions targets specific classes of applications
  — Not possible to envision all the services that applications may need
  — Not possible to add new services to existing middleware
The Internet of Things (IoT) approach

Adapt the Internet protocol suite to the WSN domain

- **Layered** networking model
- **Standard** protocols

<table>
<thead>
<tr>
<th>Layer</th>
<th>CoAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Observe, Blockwise Transfer, GroupComm, etc.</td>
</tr>
<tr>
<td>Transport</td>
<td>UDP, DTLS</td>
</tr>
<tr>
<td>Network</td>
<td>IPv6 - 6LoWPAN, IPsec, MobileIP</td>
</tr>
<tr>
<td>Link</td>
<td>IEEE 802.15.4, (RDC protocols)</td>
</tr>
<tr>
<td></td>
<td>ContikiMac, B-MAC, etc.</td>
</tr>
</tbody>
</table>

Each layer provides a different level of abstraction
IoT-based WSNs

Every node has:

- An IP address

IPv6 address
aaaa::1
IoT-based WSNs

Every node has:
- An IP address
- A set of resources

IPv6 address
aaaa::1

/temperature
/led
/location
IoT-based WSNs

Every node has:

- An IP address
- A set of resources
  - sensors

IPv6 address
aaaa::1

/temperature
/led
/location
IoT-based WSNs

Every node has:
• An IP address
• A set of resources
  – sensors
  – actuators

IPv6 address
aaaa::1

/temperature
/led
/location
IoT-based WSNs

Every node has:

- An IP address
- A set of resources
  - sensors
  - actuators
  - other kind of data

IPv6 address: `aaaa::1`
IoT-based WSNs

Resources are similar to web pages

- Accessed with an HTTP-like protocol: CoAP
- Identified by a URI
  - e.g., coap://[aaaa::1]/led

IPv6 address

aaaa::1

/temperature
/led
/location
IoT-based WSNs

GET

/temperature
/led
/location

IPv6 address aaaa::1

A resource can be retrieved
IoT-based WSNs

A resource can be updated
IoT-based WSNs

IPv6 address
aaaa::1

A resource can be created/deleted

GET/PUT

POST/DELETE

/temperature
/led
/location
IoT-based WSNs: pros and cons

Pros

General purpose
• Applications are built on top of well-defined abstractions

Extensibility
• New abstractions can be built leveraging existing ones

Cons

Lack of full control over the WSN
• Sub-optimal use of resources
• QoS is challenging
  – Especially real-time
Outline

1. Background and motivation
2. Contributions
3. In-network processing in IoT-based WSNs
4. Conclusions
High-level abstraction for WSNs: contributions

The middleware approach
- Mirtes
  - Real-time features
- ScanTraffic
  - Smart camera networks

The IoT approach
- PyoT
  - Macro-programming
- T-Res
  - In-network processing

All solutions implemented for mote-class hardware (16-128 kB RAM and 128-512 kB ROM)
Mirtes: real-time middleware

• Database-like abstraction
• Novelty: real-time features
  – Periodic queries with deadline
  – Jitter-free periodic sampling
• Especially suited for time-discrete control applications
  – System output is measured exactly at instant $t_K$
  – Measured value is available before instant $t_{K+1}$
ScanTraffic: middleware for SCNs

Middleware managing smart camera networks (SCNs) for traffic monitoring

- Mirtes adapted to SCNs
- Remote configuration
- Remote code-update
- **Real deployment** at the Pisa Airport
  - 21 smart cameras monitoring
    - 83 parking spaces
    - 8 traffic lanes
PyoT: a macro-programming framework for IoT-based WSNs

Python-based macro-programming
• resources are abstracted as Python objects
• communication details are completely hidden

Other features
• cache
• resource discovery
• federation of multiple WSNs

```python
temps = Resource.objects.filter(uri='temp')
results = []
for temp in temps:
    results.append(temp.GET())
avg = sum(results)/len(results)
if avg > 24:
    Resource.objects.get(ip='2001::1',
                          uri='fan').PUT('on')
```
PyoT: a macro-programming framework for IoT-based WSNs

Python-based macro-programming
• resources are abstracted as Python objects
• communication details are completely hidden

Other features
• cache
• resource discovery
• federation of multiple WSNs

Main limitation: no in-network processing

```python
temps = Resource.objects.filter(uri='temp')
results = []
for temp in temps:
    results.append(temp.GET())
avg = sum(results)/len(results)
if avg > 24:
    Resource.objects.get(ip='2001::1', uri='fan').PUT('on')
```
Outline

1. Background and motivation
2. Contributions
3. In-network processing in IoT-based WSNs
4. Conclusions
Motivations and core idea

Motivation
• Lack of a common way for changing the interactions among IoT nodes and the data processing they perform

Core Idea
• Use resources to represent in-network processing tasks: the task resource abstraction (T-Res)
Challenges

Design a **platform-independent solution**
- Native code cannot be used for defining the data-processing of IoT nodes

Design a **feasible solution**
- Mote-class devices must be able to support it
- Resulting power consumption and application execution time must be acceptable
Just simple tasks

Limit the scope of T-Res to *simple data-processing tasks* that:

1. Monitor one or more resources (input sources)
2. Perform some data-processing on their values
3. Send the resulting output to other resources (output destinations)
A RESTful interface for tasks

Represent each task as a CoAP resource

- With a well-defined RESTful interface

<table>
<thead>
<tr>
<th>Resource</th>
<th>Content</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tasks</td>
<td>List of installed tasks</td>
<td>GET</td>
</tr>
<tr>
<td>[task-name]</td>
<td>A specific task</td>
<td>GET/PUT/DELETE</td>
</tr>
<tr>
<td>/is</td>
<td>URLs of the input sources</td>
<td>GET/PUT/POST</td>
</tr>
<tr>
<td>/od</td>
<td>URL of the output destination</td>
<td>GET/PUT/POST</td>
</tr>
<tr>
<td>/pf</td>
<td>Processing function</td>
<td>GET/PUT</td>
</tr>
<tr>
<td>/lo</td>
<td>Last output produced</td>
<td>GET (observe)</td>
</tr>
</tbody>
</table>
Processing Function

- Processing functions are written in Python
  - More precisely, a processing function is a Python script compiled into Python bytecode

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getInput()</td>
<td>Get the current input</td>
</tr>
<tr>
<td>getInputTag()</td>
<td>Get the tag of the current input</td>
</tr>
<tr>
<td>setOutput()</td>
<td>Set the output</td>
</tr>
<tr>
<td>getState()</td>
<td>Get the last saved state</td>
</tr>
<tr>
<td>saveState()</td>
<td>Save the state</td>
</tr>
</tbody>
</table>
Example: heat control application

Keep the temperature in the range 19–21°C
Example: heat control application

aaaa::1
/sens/temperature

aaaa::2
/act/heater
/tasks/heatcontrol
Example: heat control application

```
coap://[aaaa::1]/sens/temperature
```

```
/act/heater
/tasks/heatcontrol/is
/tasks/heatcontrol/od
/tasks/heatcontrol/pf
/tasks/heatcontrol/lo
```
Example: heat control application

```
Example: heat control application

aaaa::1
/sens/temperature

coop://[aaaa::2]/act/heater

aaaa::2
/act/heater
/tasks/heatcontrol/is
/tasks/heatcontrol/od
/tasks/heatcontrol/pf
/tasks/heatcontrol/lo
```
Example: heat control application

```python
from tres import *

t = getInput()
if t < 19:
    setOutput("on")
if t > 21:
    setOutput("off")
```
Implementation

• Built on top of the Contiki Operating System, extended with
  – IPv6 loopback communication
  – Client-side support for CoAP observe

• PyMite used for running Python bytecode
  – PyMite is a reduced Python interpreter for embedded systems
    • By Dean Hall
  – I integrated PyMite in Contiki
Development Platform

The WiSMote Platform

TI MSP430F5437x

CPU Frequency 16 MHz
RAM 16 kB
Flash Memory 256 kB
T-Res memory requirements

The WiSMote Platform

<table>
<thead>
<tr>
<th></th>
<th>RAM</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15 kB (92%)</td>
<td>95 kB (37%)</td>
</tr>
</tbody>
</table>
Evaluation setup

Cooja used to simulate a multi-hop IoT-based WSN

• The sensor produces data periodically (every second)
• The border router connects the WSN to the Internet
• The actuator can receive PUT requests
• Node 2 and 4 just forward packets
• Every node is also a T-Res node
Evaluation setup

• Two different applications are considered
  – A threshold-based event detector
  – A PID controller
• Each application is implemented twice:
  – Using the external logic approach
  – Using T-Res
• The performance of T-Res is compared with that of the external logic approach
Topologies

Topology #0

Ext. Logic

Sensor

Actuator

Topology #1

Ext. Logic

Sensor

Actuator

Topology #2

Ext. Logic

Sensor

Actuator

Topology #3

Ext. Logic

Sensor

Actuator
Topologies

Topology #0

Ext. Logic

3

2

Sensor

4

Actuator

5

1

Topology #1

Ext. Logic

3

2

Sensor

4

Actuator

5

1

Topology #2

Ext. Logic

3

2

Sensor

4

Actuator

5

1

Topology #3

Ext. Logic

3

2

Sensor

4

Actuator

5

1
Network Traffic

- **PID control**
- **Event Detection**

- Network Traffic [bytes]

- Topology

- Ext. Logic
- T-Res

Daniele Alessandrelli
Application Delay

PID control

Event Detection

Daniele Alessandrelli
Energy Consumption

PID Control

- Ext. Logic (avg) - T-Res (avg)
- Ext. Logic (max) - T-Res (max)

Event Detection

- Ext. Logic (avg) - T-Res (avg)
- Ext. Logic (max) - T-Res (max)
T-Res final remarks

- T-Res enables **reconfigurable in-network processing** in IoT-based WSNs
- T-Res **simplifies the development of IoT applications**
- T-Res can run on constrained devices
- T-Res reduces network traffic and energy consumption
  - While maintaining an application delay similar to that of the external logic approach
- T-Res **complements the external logic approach used by PyoT**
Outline

1. Background and motivation
2. Contributions
3. In-network processing in IoT-based WSNs
4. Conclusions
Conclusions

Two approaches for easing the development of applications for WSNs

• The middleware approach
  – Suitable for real-time applications (see Mirtes) and applications requiring high-efficiency (see ScanTraffic)

• The IoT approach
  – Promising for achieving general-purpose WSNs
  – PyoT and T-Res are two steps in such a direction
The IoT App vision

Sensor network applications that can be downloaded from an App Store and installed in our domestic sensor networks with just a click.
THANK YOU!
List of publications (1/2)

• **D. Alessandrelli, P. Pagano, C. Nastasi, M. Petracca, and A.F. Dragoni**
  
  **MIRTES: middleware for real-time transactions in embedded systems**
  *Proceedings of the 3rd IEEE International Conference on Human-System Interaction (HSI), May 2010.*

• **D. Alessandrelli, A. Azzarà, M. Petracca, C. Nastasi, and P. Pagano**
  
  **ScanTraffic: Smart Camera Network for Traffic Information Collection**
  *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN), Springer-Verlag, February 2012.*

• **A. Azzarà, D. Alessandrelli, S. Bocchino, M. Petracca, and P. Pagano**
  
  **Architecture, functional requirements, and early implementation of an instrumentation grid for the IoT**
  *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC), June 2012.*

• **D. Alessandrelli, M. Petracca, and P. Pagano**
  
  **T-Res: enabling reconfigurable in-network processing in IoT-based WSNs**
List of publications (2/2)

• P. Pagano, M. Petracca, D. Alessandrelli, and C. Salvadori
  *Is ICT mature for an EU-wide intelligent transport system?*

• D. Alessandrelli, L. Mainetti, L. Patrono, G. Pellerano, M. Petracca, and M.L. Stefanizzi
  *Performance evaluation of an energy-efficient MAC scheduler by using a test-bed approach*

• D. Alessandrelli, L. Mainetti, L. Patrono, G. Pellerano, M. Petracca, and M. L. Stefanizzi
  *Implementation and validation of an energy-efficient MAC scheduler for WSNs by a test bed approach*
  *Proceedings of IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCom), September 2012*

• B. Abid, M. Petracca, P. Pagano, S. Bocchino, and D. Alessandrelli
  *Experimental validation of wireless localization techniques in IEEE 802.15.4 networks*
  *Proceedings of 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), September 2011.*