Protocol Awareness:
A Step Towards Smarter Sensors

Hoel Iris, Francois Pacull

CEA-LETI MINATEC Campus, France
Francois.Pacull@cea.fr
Context

Building Automation Systems

Communication reliability
Energy consumption
Coordinated actions
Context

Communication reliability
Energy consumption
Coordinated actions

Most of the communications ensure only a best effort
Communication reliability
Energy consumption
Coordinated actions

Sensing rate is under the responsibility of the sensors
Context

Communication reliability
Energy consumption
Coordinated actions

Difficult to ensure the performance of a group of actions
Outline of the presentation

- The high level coordination protocol we rely on
- How we make the sensors aware of this protocol
- 2 examples as illustration
Middleware / coordination protocol

**Associative Memory**

**Production Rules**

**Precondition** based on the Rd()

“when these conditions are reached I would trigger something”

**Performance**

To verify the Rd() are still valid
To consult some resources Rd()
To consume some resources Get()
To produce some resources Put()

**Database record** (field1, field2, field3)

**Event** (evenid, type, tm, payload)

**Service** (in1, in2, out1, out2, out3)

**Sensor** (id, type, value)

**Actuator** (id, cmd, p1, p2, p3)

Rd(), Get() and Put() operations are performed as a sequence of transactions

\{ ... \} \{ ... \} \{ ....\}

each of the transaction into curly bracket enforces all-or-nothing
Protocol aware sensor

The Rd(), Get() and Put() are embedded in transactions

Initial approach

Communication Protocol

New approach

Coordination protocol

Gateways

Sensor

Actuator

Initial approach

New approach
Example of transaction committed

Coordinator

Phase 1

bag1.rd()

bag2.get()

bag3.put()

bag4.put()

Phase 2 commit

ok

ok

ok

ok

done
Example of transaction cancelled (processing)

Coordinator

Phase 1

ok
nok
ok
ok

Phase 2 abort
done

bag1.rd()

bag2.get()

bag3.put()

bag4.put()
Example of transaction cancelled (failure)
Platforms

OpenPicus Flyport + integrated Wifi (802.11 b/g/n)

16bits micro-controleur, 32MHz, 256Ko Flash, 16Ko Ram
26 I/O
Wifi (802.11 b/g/n)

Arduino - Xbee (802.15.4)

8bits micro-controleur, 8MHz, 32Ko Flash, 2Ko Ram
20 I/O
Xbee (802.15.4)
Wake up

Boards can be put in sleep mode
- communication
- micro-controller
Boards can be wake up by external events
- e.g. I/O pin set to high level

Signal from application to signal that we need to talk to the micro controller

Same mechanism to wake up the micro controller only when the physical sensor has something useful to say when the application needs to interrogate the sensor

open-contact
e.g. Detect the opening of a door

Physical sensor

Micro-controller

Radio

Battery
Signal from application to warn that we need to talk to the micro controller

This is out the scope of this paper and let to further investigation

We used infrared because it was the simpler

Several possibilities
low cost wireless signal
passive RFID
infrared, ...

Coordinator

Physical sensor

Micro controller

Radio

Battery
Coordination Protocol
Precondition (not transactional)

Coordinator

Smart sensor

Physical sensor

Invoked wake up signal

Invocation

reply

Interrogation of the sensor (immediate reading)
Coordination Protocol
Performance (transactional)

Coordinator

Smart sensor

Physical sensor or actuator

Infrared wake up signal

op ()

Phase 1

ok/nok

Phase 2 commit/abort

done

Verification that the operation can be actually performed

Performance or release initial state
Coordination Protocol
Get()

Coordinator

Smart sensor

Physical sensor

Infrared wake up signal

\( \text{bag.get}(a,b,c) \)

ok/nok/retry

commit/abort

done

Is \((a,b,c)\) still valid?
if no return “nok”
if yes

Is \((a,b,c)\) locked?
if no, lock it and return “ok”
if yes return “retry”

If commit
remove \((a,b,c)\), return “done”
If abort
release lock, return “done”
Coordination Protocol

Put ()

Verification that the operation can be actually performed

Performance or Release initial state

Coordinator

Smart sensor

Physical Actuator

Infrared wake up signal

Phase 1

ok/nok

Phase 2 commit/abort

done
Main interests

Precondition phase:
Interrogate the sensor only when needed by the application

→ impact on the power consumption

Performance phase:
Verify that the command sent to an actuator is physically possible

→ ease the management of group of actuators
Example 1

Algorithm using temperature sensors where the interrogation of the sensors is not predictable but relies on computation done by the previously read values.

*Example*

- e.g. accelerate the pace when temperature delta increases quickly

**Classical approach**
The sensor send the temperature every 5 minutes

\[24 \times 12 = 288\text{ measures}\]

**Application driven approach**
The application interrogates the sensor when required

let say that 50 measures are enough
Example 1

<table>
<thead>
<tr>
<th>Micro + Radio</th>
<th>Idle</th>
<th>running</th>
<th>wakeup + request + sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flyport + Wifi</td>
<td>97µA</td>
<td>127.5mA</td>
<td>1s</td>
</tr>
<tr>
<td>Arduino + Xbee</td>
<td>206µA</td>
<td>57.1mA</td>
<td>0.04s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Micro + Radio</th>
<th>classical</th>
<th>Application driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flyport + Wifi</td>
<td>0.33%</td>
<td>0.058%</td>
</tr>
<tr>
<td>Arduino + Xbee</td>
<td>0.0133%</td>
<td>0.00231%</td>
</tr>
</tbody>
</table>

R = \frac{\text{Running Time}}{\text{Total Time}}

Cons = R \cdot C_{\text{Running}} + (1 - R) \cdot C_{\text{idle}}

Autonomy = \frac{\text{Cons}}{1300\, \mu\text{Ah}}

<table>
<thead>
<tr>
<th>Micro + Radio</th>
<th>classical</th>
<th>App. driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flyport + Wifi</td>
<td>105 days</td>
<td>328 days</td>
</tr>
<tr>
<td>Arduino + Xbee</td>
<td>253 days</td>
<td>261 days</td>
</tr>
</tbody>
</table>

More important to save on idle state than on running state

Costly but simpler to deploy wireless protocol is affordable
Example 2

We want to coordinate 2 servo-motors such that their combined moves allow to turn from 0 to 360 degrees while they can only turn 180 degrees each.

Transaction will fail if servo-motor receive out of range order

```
["Application", "Angle"]=rd(angle) &
::
{
["Application", "Angle"]=get(angle) ;
["Flyport", "Actuator"]=put("position", angle) ;
["Arduino", "Actuator"]=put("position", "180") ;
}
{
["Application", "Angle"]=get(angle) ;
["Flyport", "Actuator"]=put("position", "180") ;
["Arduino", "Actuator"]=put("position", angle) ;
}.
```

Fail if angle not in 0-180

Fail if angle not available

Fail if angle not in 180-360
Example 2

angle = 70°

"Application", "Angle"].rd("70") &
::
{
  "Application", "Angle"].get("70") ;
  ["Flyport", "Actuator"].put("position", "70") ;
  ["Arduino", "Actuator"].put("position", "180") ;
}
{
  "Application", "Angle"].get(angle) ;
  ["Flyport", "Actuator"].put("position", "180") ;
  ["Arduino", "Actuator"].put("position", "70") ;
}.

angle = 270°

"Application", "Angle"].rd("270") &
::
{
  "Application", "Angle"].get("270") ;
  ["Flyport", "Actuator"].put("position", "270") ;
  ["Arduino", "Actuator"].put("position", "180") ;
}
{
  "Application", "Angle"].get(angle) ;
  ["Flyport", "Actuator"].put("position", "180") ;
  ["Arduino", "Actuator"].put("position", "270") ;
}.
Conclusion

- High level coordination protocol on micro-controllers
- Better usage of application knowledge has a significant impact of the consumption.
  - Saving on running mode is not enough
  - “more costly” wireless protocol, easier to deploy is not always a bad idea.
- Embedded distributed actions into transaction
  - Use the 1st phase to verify the action is actually possible
  - Ensure all-or-nothing property
Future work

- Work on the wake up signal
  Involve other teams of CEA-Leti

- More complex scenario
  - Abandoned sensors
    First sensor waked up by alarm, others sensors by application
    (we are not very far from our 1\textsuperscript{st} example)

  - Robot with motorized camera
    Tracking an object by moving either the camera or the robot
    But the camera can be at the end of the range and the robot blocked by
    an obstacle.
    (we are not very far from our 2\textsuperscript{nd} example)