Networking issues for the Internet of Things
Giacomo Morabito
University of Catania

2015 – IEEE SPS – Italy Chapter
Summer School on Signal Processing
Acknowledgements

• The following are the results of the work carried out with many colleagues and reported in the following papers:
Outline

- Great, however...
- The need of a network operating system for the IoT
- Network operating systems
  - Open Network Operating System (ONOS)
- SDN for wireless sensor and actor networks
- Prototype implementation
- Conclusions
Great, however...

The “next” big thing in communications? General public

- 50 billion IoT devices by 2020 (Cisco Systems)
- Google trends (http://www.google.com/trends)
The “next” big thing in communications? Scientific community

Most downloaded papers:

- **IEEE (4th):**
  - Top 10 for almost 1 year

- **ACM-SIGCOMM:**
  - BTW: 4th and 5th are about SDN and ONOS. We’ll talk later about these...

- **Elsevier – Computer science:**
  - BTW: the second is “Internet of Things (IoT): A vision, architectural elements, and future directions”
The “next” big thing in communications? Industry

From techcruch.com
# The standards

<table>
<thead>
<tr>
<th>I GENERATION</th>
<th>Tagged objects</th>
<th>EPCglobal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor networks</td>
<td>IEEE 802.15.4</td>
<td></td>
</tr>
<tr>
<td>Object description</td>
<td>IEEE 1451</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>II GENERATION</th>
<th>Internetworking</th>
<th>IETF 6LoWPAN</th>
<th>IETF ROLL RPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web of Things</td>
<td>IETF CoAP</td>
<td>OASIS DPWS</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>ITU-T FS M2M</td>
<td>oneM2M</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>III GENERATION</th>
<th>Cloud computing</th>
<th>TIA TR50</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Social networking</td>
<td>Missing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Future Internet</td>
<td>IETF ICNRG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantics</td>
<td>W3C SSN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://sdn-wise.dieei.unict.it  
2015 – IEEE - SPS Italy Chapter
Intranets of Things instead of Internet of Things

- Several IoT platforms developed independently without a clear reference architecture → **Fragmented technological landscape**
  - Low interoperability
  - Low expandability
  - Low reusability

- Where would you test your new algorithm for in-network data processing?
Existence of heterogeneous platforms: not a new problem
Existence of heterogeneous platforms: how it was addressed?

Driver
Operating system
Printer abstraction
Printing service

Word Adobe PPT

http://sdn-wise.gieei.unict.it
2015 – IEEE
Existing operating systems for the IoT

- Contiki
- RIOT
- CCN-Lite
Contiki

- Open source OS for the IoT
- Supports:
  - IPv6 and IPv4
  - 6LOWPAN
  - RPL
  - CoAP
- Active community
- Industrial interest
- Network simulator: Cooja
Contiki: Architecture

Contiki Operating System

- App. 1
- App. 2
- Node management
  - Neighbor discovery
  - Function installer
  - Sensor configuration

Forwarding
Loader
Proto threads

Radio
MCU
Sensors
...

Core
Drivers
Hardware
<table>
<thead>
<tr>
<th>MCU/SoC</th>
<th>Radio</th>
<th>Platforms</th>
<th>Cooja simulation support</th>
</tr>
</thead>
<tbody>
<tr>
<td>RL78</td>
<td>ADF7023</td>
<td>EVAL-ADF7023DB1</td>
<td>-</td>
</tr>
<tr>
<td>TI CC2538</td>
<td>Integrated</td>
<td>cc2538dk</td>
<td>-</td>
</tr>
<tr>
<td>TI MSP430x</td>
<td>TI CC2420</td>
<td>exp5438, z1</td>
<td>Yes</td>
</tr>
<tr>
<td>TI MSP430x</td>
<td>TI CC2520</td>
<td>wismote</td>
<td>Yes</td>
</tr>
<tr>
<td>Atmel AVR</td>
<td>Atmel RF230</td>
<td>avr-raven, avr-rcb, avr-zigbit, iris</td>
<td>-</td>
</tr>
<tr>
<td>Atmel AVR</td>
<td>TI CC2420</td>
<td>micaz</td>
<td>Yes</td>
</tr>
<tr>
<td>Freescale MC1322x</td>
<td>Integrated</td>
<td>redbee-dev, redbee-econotag</td>
<td>-</td>
</tr>
<tr>
<td>ST STM32w</td>
<td>Integrated</td>
<td>mb851, mbxxx</td>
<td>-</td>
</tr>
<tr>
<td>TI MSP430</td>
<td>TI CC2420</td>
<td>sky, jcreate, sentilla-usb</td>
<td>Yes</td>
</tr>
<tr>
<td>TI MSP430</td>
<td>TI CC1020</td>
<td>msb430</td>
<td>-</td>
</tr>
<tr>
<td>TI MSP430</td>
<td>RFM TR1001</td>
<td>esb</td>
<td>Yes</td>
</tr>
<tr>
<td>Atmel Atmega128 RFA1</td>
<td>Integrated</td>
<td>avr-atmega128rfa</td>
<td>-</td>
</tr>
<tr>
<td>Microchip pic32mx795f512l</td>
<td>Integrated</td>
<td>seed-eye</td>
<td>-</td>
</tr>
<tr>
<td>TI CC2530</td>
<td>Integrated</td>
<td>cc2530dk</td>
<td>-</td>
</tr>
<tr>
<td>RC2300/RC2301</td>
<td>Integrated</td>
<td>sensinode</td>
<td>-</td>
</tr>
<tr>
<td>6502</td>
<td>-</td>
<td>apple2enh, atari, c128, c64</td>
<td>-</td>
</tr>
<tr>
<td>Native</td>
<td>-</td>
<td>native, minimal-net, cooj a</td>
<td>Yes</td>
</tr>
</tbody>
</table>
RIOT

- HW Support: MSP430, ARM7, CORTEXM0-4, X86
- Drivers for many transceivers and sensors
- SW Support:
  - AODVv2
  - 6LoWPAN
  - RPL
  - TCP with header compression for 6LoWPAN
  - CCN-lite
  - OpenWSN
  - CoAP, CBOR, and UBJSON

http://sdn-wise.dieei.unict.it  2015 – IEEE - SPS Italy Chapter
RIOT

- **Modular**: When programming a node you can decide which modules to be loaded
- Written in C/C++
- Has multithreading and RealTime operations
- Same memory space requirements as TinyOS
RIOT Stack
# RIOT and the others

<table>
<thead>
<tr>
<th>OS</th>
<th>Min RAM</th>
<th>Min ROM</th>
<th>C Support</th>
<th>C++ Support</th>
<th>Multi-Threading</th>
<th>MCU w/o MMU</th>
<th>Modularity</th>
<th>Real-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contiki</td>
<td>&lt; 2kB</td>
<td>&lt; 30kB</td>
<td>•</td>
<td>×</td>
<td>•</td>
<td>✓</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Tiny OS</td>
<td>&lt; 1kB</td>
<td>&lt; 4kB</td>
<td>×</td>
<td>×</td>
<td>•</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Linux</td>
<td>~ 1MB</td>
<td>~ 1MB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>RIOT</td>
<td>~ 1.5kB</td>
<td>~ 5kB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **OS:** Operating System
- **Min RAM:** Minimum RAM required
- **Min ROM:** Minimum ROM required
- **C Support:** Support for C programming language
- **C++ Support:** Support for C++ programming language
- **Multi-Threading:** Support for multi-threading
- **MCU w/o MMU:** Support for microcontrollers without memory management unit
- **Modularity:** Modularity support
- **Real-Time:** Real-time support

*Note: Full support is denoted with a checkmark (✓), partial support with a dot (•), and no support with an X (×).*
CCN-LITE

- CCN-lite has been included in the RIOT operating system for the Internet of Things (IoT): [http://www.riot-os.org/](http://www.riot-os.org/)
- Objective of CCN-lite has been to make the most popular implementation of a content-centric networking client (CCNx) runnable in a IoT device
Great! However...

- IoT applications can be developed without considering the specific features of the hardware platform
- Only/mostly for 6LOWPAN nodes
- For what concerns networking:
  - Is it possible to deploy new routing algorithms application specific?
  - Is it possible to decide the path according to the values measured by a sensor?
  - Is it possible to change the network topology depending on the characteristics of the flow?
- For DSP researchers:
  - If you develop a new scheme which requires packets to follow specific routes, how would you test it?
Great! However, we also need...

- A **Network Operating System**...
Network operating systems
Network Operating Systems (NOS)

- Holistic network resource management
- Access network resources through dedicated services
- Integrate heterogeneous network elements through drivers that implement NOS functionality leveraging device-specific technology
- Support third-party network services deployment on top of them (e.g. Routing as a Service)
Open Network Laboratory

ON.LAB

- No profit established in 2012
- Partners (it might be outdated):

1. BLACKDUCK
2. cnit
3. CREATE-NET
4. AT&T
5. Ciena
6. CISCO
7. ETRI
8. Consortium
9. GARR
10. happiest minds
11. ERICSSON
12. FUJITSU
13. HUAWEI
14. Infoblox
15. INTERNET
16. KREN
17. intel
18. NEC
19. NSF
20. KiSTi
21. SRI International
22. SK telecom
Open Network Operating System (ONOS)

- Open Source
- Java-based
- OSGi deployment
- Modular architecture
- Extensible components
- Originally designed for OpenFlow
ONOS Overall Architecture

Apps

NB (Consumer) API

Core
(Device, Host, Link, Topology, Path, Flow, Intent, Network, ...)

SB (Provider) API

Providers
(Device, Host, Link, Flow)

Protocols

Network Elements
ONOS Functional Organization

- Subsystems consisting of elementary services deployed in several layers

- Communication between layers is established through system-wide APIs:
  - Northbound API provides network applications and services with access to ONOS subsystems
  - Southbound API enables the deployment of core services regardless the device-specific implementation details
SDN for Network Management

- Separation of control and data plane
- Control plane remotely managed by dedicated services – controllers
- OpenFlow is the de facto standard for communication between controllers and network elements
- However: not all network elements can/do support OpenFlow
A few facts about wireless sensor networks

- Mature technology since early 2000s
- Challenging communication & networking environment
- Requirements extremely application specific

The bottom-line...

There is nothing like a one-fits-all solution

<table>
<thead>
<tr>
<th>Upsides:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Large number of solutions proposed</td>
</tr>
<tr>
<td>- Deep understanding of the WSN domain</td>
</tr>
<tr>
<td>- Zillions of papers, citations, academic promotions, projects</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Downsides:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- High solution specialization</td>
</tr>
<tr>
<td>- Market fragmentation</td>
</tr>
<tr>
<td>- Burden on application developers</td>
</tr>
<tr>
<td>- Low reusability</td>
</tr>
</tbody>
</table>
The consequence...

It’s not taking off!

http://sdn-wise.dieei.unict.it
Related work

SDN-WISE
Software Defined Networking (SDN) clearly separates:

- **Data plane**: run by network Switches
- **Control plane**: implemented by a software program running on a server (the Controller)

Modifying the behavior of the network as easy as it is installing a new piece of software on a PC

**OpenFlow** is the most popular implementation of the SDN paradigm
**SDN in WSNs**

- Few attempts to extend SDN to WSNs:
  - Software Defined Wireless Networks (SDWN), 2012
  - Sensor OpenFlow, 2012

- Different requirements:
  - **Traditional wired networks**
    - Velocity
  - **WSNs**
    - Efficiency
    - Flexibility
    - Memory occupancy
SDN-WISE: Basic concepts

- Directly derived by OpenFlow
- Separation (even physical) between
  - **data plane** (executed by sensor nodes)
  - **control plane** (executed by the Controller)
- When an event (e.g., the arrival of a packet) occurs sensor nodes behave as specified in the WISE Table
- If there is no relevant information in the WISE Table → Ask the Controller
- The Controller replies sending a new entry for the WISE Table
- A simple protocol defined to allow nodes to:
  - Learn the shortest path towards the (closest) sink(s)
  - Discover the neighboring nodes
  - Periodically report local information to the Controller (through the sink)
## WISE Table

<table>
<thead>
<tr>
<th>Matching Rule</th>
<th>Matching Rule</th>
<th>Matching Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>=</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>=</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>=</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>=</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>M</td>
</tr>
<tr>
<td>Modify</td>
<td>1</td>
</tr>
<tr>
<td>Modify</td>
<td>1</td>
</tr>
<tr>
<td>Forward</td>
<td>0</td>
</tr>
<tr>
<td>Drop</td>
<td>0</td>
</tr>
<tr>
<td>Forward</td>
<td>0</td>
</tr>
</tbody>
</table>
SDN-WISE Architecture

WISE-VISOR
TOPOLOGY MANAGER

CONTROL-LER 1

CONTROL-LER K

RMI

SOAP

APPLICATION
MCU

INPP
FWD
MAC
PHY

TCP/IP

ADAPTATION
FOR REAL NETWORK

ADAPTATION
FOR SIMULATOR

Real Sink

Simulated Sink

Sensor Node

Embedded system

Emulated Node

IEEE 802.15.4

OMNET++

TD SIM

INPP SIM

FWD SIM

MAC SIM

PHY SIM
Major features (compared to OpenFlow)

1. Statefulness → SDN-WISE nodes are Turing complete
2. Flexible definition of rules
3. Support of duty cycles
4. Support of multitenancy (beyond slicing)
5. Lots of deployment options and programming languages
6. Integration with simulation environments (OMNET++ & OPNET)
IoT Integration – ONOS

Level

- Protocols layer: Implementation of the SDN-WISE driver
- Providers layer: Translation of SDN-WISE-specific details to ONOS low-level abstractions for network resources
- SB API: Unchanged
- Core: Introduction of new services for maintaining WSN-specific information
- NB API: Provision of WSN-specific abstractions, such as SensorNode API
An Integrated NOS for the IoT

Applications

Northbound

Distributed Core

Providers

Protocols

Packet Forwarding APP

SensorNode API
FlowRule API
Path API
Host API
Device API

SensorNode Service
FlowRule Service
Topology Service
Host Service
Packet Service

Device Provider
Link Provider
Host Provider
Flow Provider
Packet Provider

SensorNode Controller

OpenFlow Controller

http://sdn-wise.dieei.unict.it 2015 – IEEE - SPS Italy Chapter
And from a User Point of View
Basic Advantages

- Holistic view of the topology in a device-level rather than a specific protocol level
- Re-use of ONOS components originally designed to support OpenFlow functionality:
  - FlowRules API has been extended to also support SDN-WISE; however the FlowRules service has remained the same
- All information regarding sensors are kept in the Core and any third-party application can access it through ONOS extended NB API
Conclusions and current work
Conclusions

- The IoT needs a network operating system to overcome fragmentation
- Network operating systems have attracted large attention by the R&D community
- However, so far focus has been on wired networks: other IoT components have specific features
- We have taken a few steps in this direction:
  - We have extended ONOS to integrate wireless sensor and actor networks
  - We have developed and tested SDN-WISE
  - We are experimenting the extended ONOS
A challenge for you

- Up to now *networking* has been a bunch of protocols
- Current trend: overcome this approach and create abstractions of network functionalities
- **Is the same abstraction process possible for DSP?**
  - Can you identify a (quasi) complete set of building blocks?
  - Can you define standard APIs?
  - Can you describe complex schemes as a sequence of entries in a table?
Q&A

http://sdn-wise.dieei.unict.it/