Smart Sensors, Measurement and Instrumentation 9

Subhas Chandra Mukhopadhyay *Editor*

Internet of Things

Challenges and Opportunities

Springer

Smart Sensors, Measurement and Instrumentation

Volume 9

Series editor

S. C. Mukhopadhyay, Palmerston North, New Zealand

For further volumes:
http://www.springer.com/series/10617

Subhas Chandra Mukhopadhyay
Editor

Internet of Things

Challenges and Opportunities

Springer

Editor
Subhas Chandra Mukhopadhyay
School of Engineering and Advanced
Technology
Massey University (Turitea Campus)
Palmerston North
New Zealand

The book editor is "Guest Editor".

ISSN 2194-8402 ISSN 2194-8410 (electronic)
ISBN 978-3-319-04222-0 ISBN 978-3-319-04223-7 (eBook)
DOI 10.1007/978-3-319-04223-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014930229

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This special issue titled "Internet of Things: Challenges and Opportunities" in the book series of "Smart Sensors, Measurement and Instrumentation" contains the invited papers from renowned experts working in the field. A total of 10 chapters have described the advancement in the area of Internet of Things, Wireless Sensors and Sensor Networks, protocols, topologies, Instrumentation architectures, Measurement techniques, Energy harvesting and scavenging, Signal processing, Design and prototyping in recent times.

In recent times, Internet of Things (IoT) has aroused great interest in the research, scientific and technological communities. Although wireless sensor networks have been in place for more than a few decades now, the smart wireless sensors, miniaturization, RFID have opened up a whole new application space of sensors. Internet of Things is different from traditional wireless sensor networks as well as computer networks and therefore poses more challenges to solve such as limited energy, restricted lifetime, etc.

Advancement in sensor technology, smart instrumentation, wireless sensor networks, miniaturization, RFID and information processing is helping towards the realization of Internet of Things. IoTs are finding applications in various areas of applications including environmental monitoring, intelligent buildings, smart grids and so on. This book provides design challenges to IoT, theory, various protocols, implementation issues and a few case studies.

We sincerely hope that the readers will find this special issue interesting and useful in their research as well as in practical engineering work in the area of biomedical sensing technology. We are happy to be able to offer the readers such a diverse special issue, both in terms of its topical coverage and geographic representation.

Finally, we would like to wholeheartedly thank all the authors for their contribution to this special issue.

Subhas Chandra Mukhopadhyay

Contents

Internet of Things: Challenges and Opportunities

S. C. Mukhopadhyay and N. K. Suryadevara

Abstract The term Internet of things (IoT) is used to describe embedded devices (things) with Internet connectivitsy, allowing them to interact with each other, services, and people on a global scale. This level of connectivity can increase reliability, sustainability, and efficiency by improved access to information. Environmental monitoring, home and building automation and smart grids could be interconnected, allowing information to be shared between systems that affect each other. Giving these systems better awareness can improve their efficiency, reliability and sustainability. Due to the large number of applications the IoT has the potential to replace people as the largest consumer and producer of information on the Internet. Low powered wireless embedded devices are cost effective and require little infrastructure, however the Internet and its protocols are unsuitable for such devices due to a lack of resources. IPv6 over low-power wireless area networks (6LoWPAN) was created for this purpose by the Internet Engineering Task Force (IETF). The IETF created the standards the Internet operates on. 6LowPAN allows low powered wireless devices to behave like any other Internet connected device with some restrictions. This chapter will give an introduction of the status of IoT along with the challenges and opportunities of making the IoT.

1 Introduction and Literature Review

The use of internet is increasing steadily over the years. As per the statistics [1], the number of internet users at the end of 2011 exceeded 2.2 billion and the details of the users based on geographical distribution as shown in Fig. 1. Providing internet facility is a financially costly resources and it may be extremely valuable proposition to think

S. C. Mukhopadhyay (✉) · N. K. Suryadevara
School of Engineering and Advanced Technology, Massey University,
Palmerston North, New Zealand
e-mail: S.C.Mukhopadhyay@massey.ac.nz

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors,
Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_1,

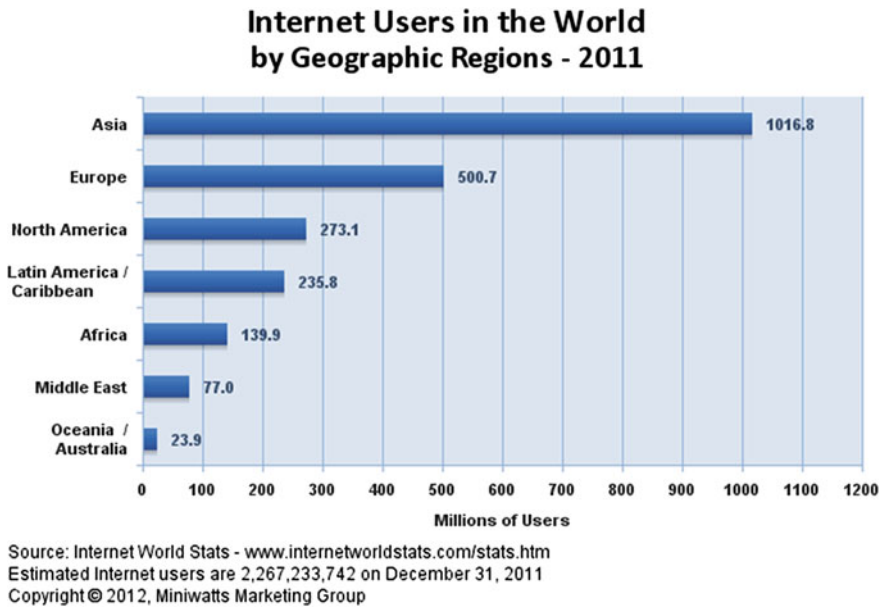


Fig. 1 The internet users in the world by geographic regions [1]

of internet for other applications. The internet can be used to transmit the sensing data collected from widely distributed regions such as measurement of environmental parameters.

The percentage of internet users based on the geographic regions is shown in Fig. 2. From Fig. 2 it is seen that a huge percentage of populations in many countries are still not enjoying the internet and there is a strong possibility of growth of internet in near future.

In recent times an enormous amount of research and development works are carried out in different parts of the world to make Internet of Things to be feasible. Every THING in this world will be connected to each other via INTERNET so that we can know anything we want to know.

Though the term "Internet of Things" was proposed by Kevin Ashton [2, 3] in 1999 but 'The Internet of Things' is a concept originally coined and introduced by MIT, Auto-ID Center and intimately linked to RFID and electronic product code (EPC) [4, 3]. The IoT literally means, "… all about physical items talking to each other..". Machine-to-machine communications and person-to-computer communications will be extended to things. Technologies that will drive the future Internet of Things: Sensor technologies including RFID, smart things, nanotechnology and miniaturization. The concept of the Internet of Things is now being influenced strongly by developments in computing and network ubiquity and developments in the next generation Internet—and considered at all levels including United Nations [5].

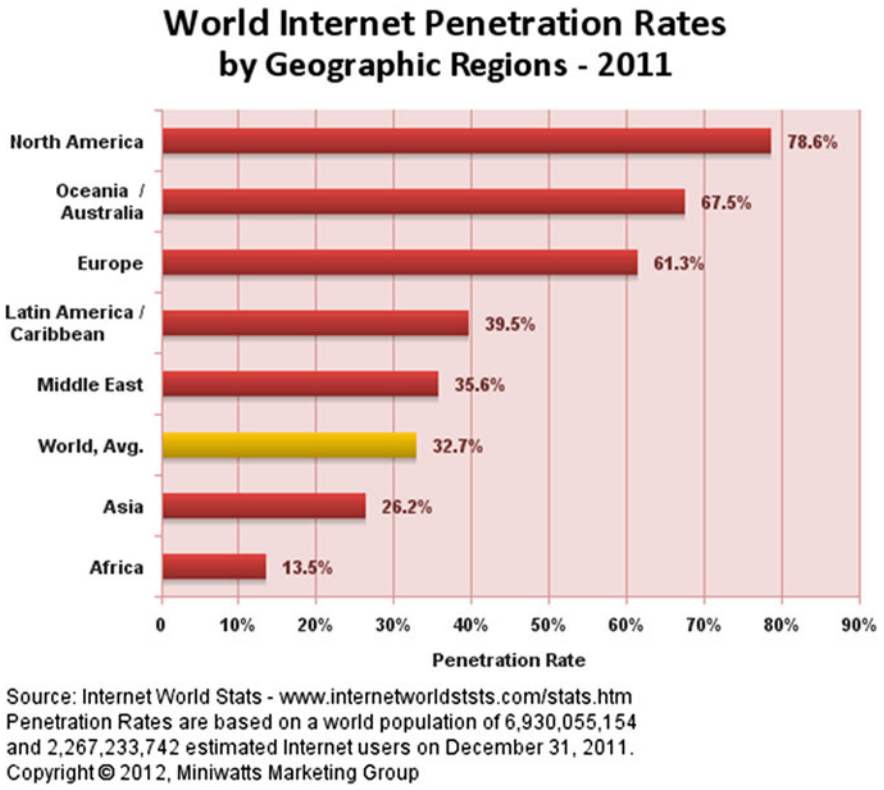


Fig. 2 The percentage internet users in the world by geographic regions [1]

"We are heading into a new era of ubiquity, where the users of the Internet will be counted in billions, and where humans may become the minority as generators and receivers of traffic. Changes brought about by the Internet will be dwarfed by those prompted by the networking of everyday objects" [3]. The importance of IoT in future can be perfectly visualized with the help of Fig. 3 [6]. It shows that there will be many times connected devices that the population in another few years' time. In future, it will use the Internet as a scaffold to support and transmit its sensations. It may consist of millions of embedded electronic measuring devices: thermostats, pressure gauges, pollution detectors, cameras, microphones, glucose sensors, ECGs, electroencephalographs. These will probe and monitor cities and endangered species, the atmosphere, our ships, highways and fleets of trucks, our conversations, our bodies-even our dreams.

A significant amount of research papers, both in journal and conferences are reported in the last few years. A special issue has recently been published by IEEE Sensors journal, Vol. 13. No. 10, October 2013 in which 35 papers have reported different works on IoT architecture, protocols, services and different applications [7].

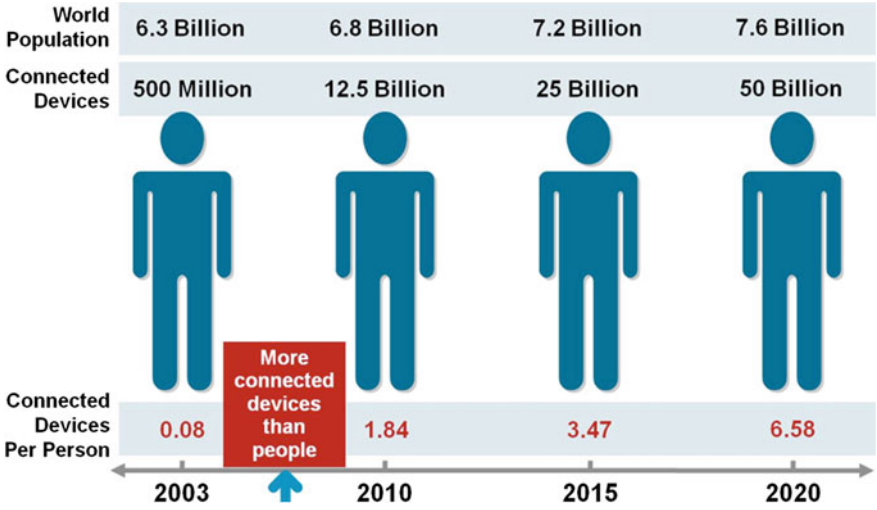


Fig. 3 The comparison of connected devices with human population [6]

A survey on IoT and future visions are reported in [8, 9]. CERP-IoT, cluster of European Research Project on the Internet of Things described the vision and challenges for realizing the Inter of Things [10]. In IEEE Xplore with the search words "Internet Of Things" gives 2071 articles, mostly published in different international conferences held in recent times (2010–2013).

A report titled "The Internet of Things Business Index: A quiet revolution gathers pace" [11], also found that 30 % of business leaders feel that the IoT will unlock new revenue opportunities, while 29 % believe it will inspire new working practices, and 23 % believe it will eventually change the model of how they operate [11]. The study found that European businesses are ahead of their global counterparts in the research and planning phases of implementing IoT [12]. Meanwhile, manufacturing is the leading sector when it comes to research and implementation of IoT technologies, driven in part by the need for real-time information to optimize productivity [12].

According to the report [11], the top five concerns that companies have around the IoT are: a lack of employee skills/knowledge; a lack of senior management knowledge and commitment; products or services that don't have an obvious IoT element to them; immaturity of industry standards around IoT; and high costs of required investment in IoT infrastructure [11]. The 779 respondents came from 71 countries across Europe (29 %), North America (29 %), Asia-Pacific (30 %) and rest of the world (12 %) [11]. However, a few steps need to be taken if the IoT revolution is to really take off [11]. The report suggests that data silos need to be removed and common standards need to be established in order to allow the IoT to scale to a size that will allow it to operate across all markets successfully [11].

In June 2013 The Economist Intelligence Unit conducted a global survey of 779 executives [13]. Some of the responses to the survey are highlighted in Fig. 4.

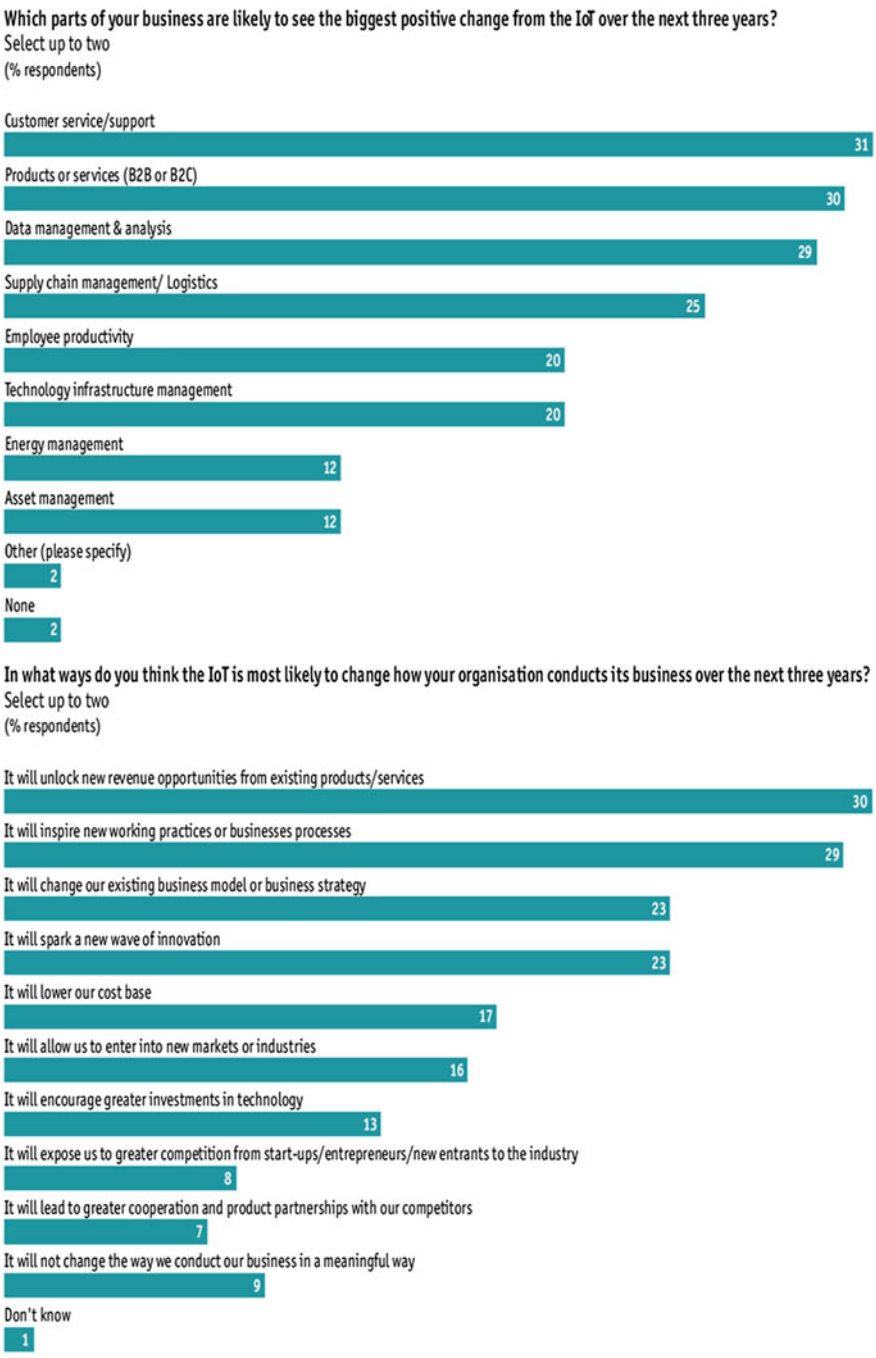


Fig. 4 Changes in the business perspective for the next 3 years [11]

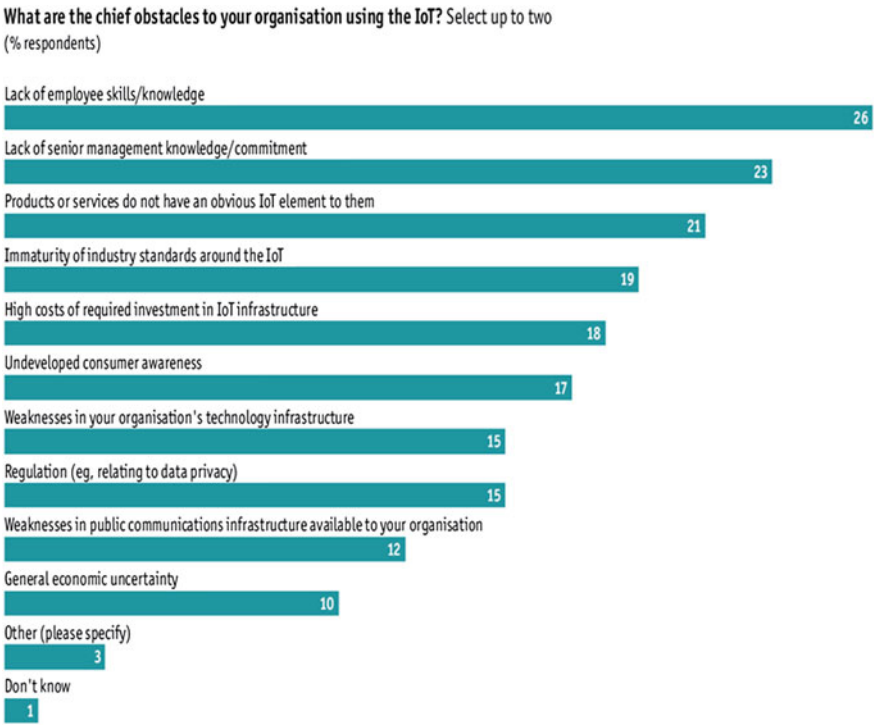


Fig. 5 Some of the main obstacles, businesses are facing in implementing IoT [11]

One of the biggest obstacles of using the IoT is the perception that products or services do not have any obvious IoT application (see Fig. 5) [11].

The full potential of the IoT will be unlocked when small networks of connected things, from cars to employee IDs, become one big network of connected things extending across industries and organizations [11]. Since many of the business models to emerge from the IoT will involve the sale of data, an important element of this will be the free flow of information across the network [11].

Another interesting fact is that the internet connectable consumer household devices will increase significantly in the next decade, with the computer network equipment that accounts for the majority of household devices, at about 75 % in 2010 and declining to 25 % by 2020 [10]. Figure 6 shows the share of Internet-Connectable consumer household devices by different types.

The IoT research needs including various technologies in terms of hardware and software are listed in the following Table 1.

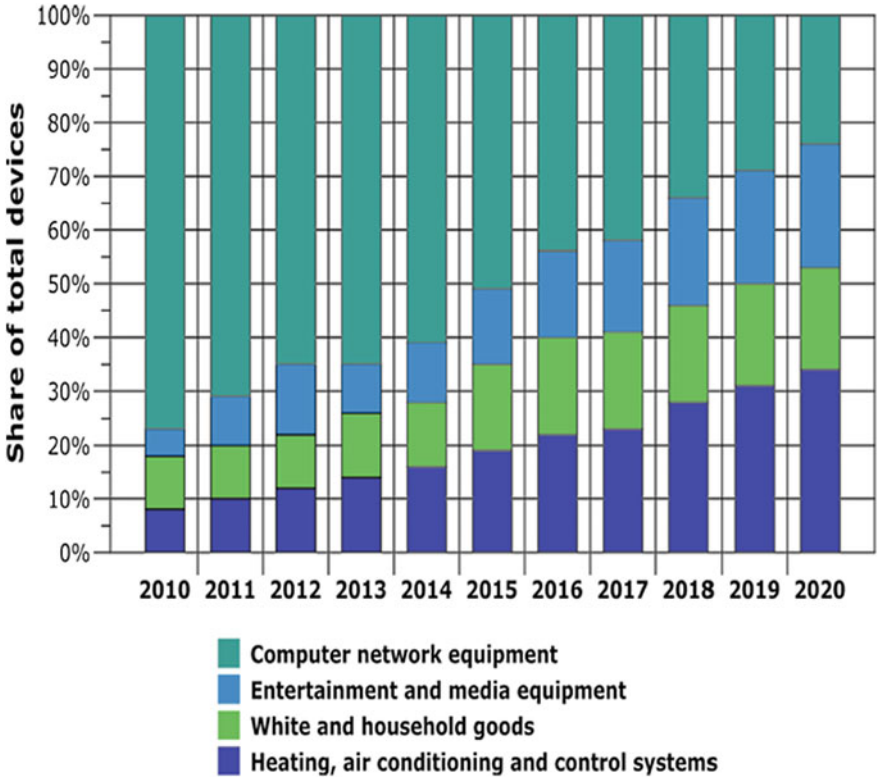


Fig. 6 Share of internet-connectable consumer household devices by type [10]

2 Challenges and Opportunities of IoT

Though development of many IoT based systems are reported there are many design challenges faced by the developers and engineers. Among many issues such as availability of internet, miniaturization, the IoT is entirely dependent on the development of Wireless Sensor Networks (WSN) and Radio Frequency Identification devices (RFID).

The wireless sensors are the extension of smart sensors with communication along with adaptation and learning capability. There are many wireless devices available around us with different functionality. The wireless devices offer many advantages in terms of cost, flexibility, power options, ease of installation and replacement. Figure 7 shows the normal situation in a house where there are so many chargers for different appliances and the wires get tangled with each other. It becomes very annoying to find the right devise at the right time so a labeling is required. The problem may be easily avoided if all the devices are made wireless eliminating the need of wired chargers. So the wireless devices may be designed with some form of

Table 1 Internet of Things scenario [14]

IoT-research needs	2011–2015	2015–2020	Beyond 2020
Identification technology	• Convergence of IP and IDs and addressing scheme • Unique ID • Multiple IDs for specific cases • Extend the ID concept (more than ID number) • Electromagnetic Identification (EMID)	Beyond EMID	Multi methods-one ID
IoT architecture	• Extranet (extranet of things) • Partner to partner applications, basic interoperability, billions-of-things	• Internet (Internet of Things) (global scale applications, global interoperability, many trillions of things)	
SOA software services for IoT	• Composed IoT services (IoT services composed of other services, single domain, single administrative entity)	• Process IoT services (IoT Services implementing whole processes, multi/cross domain, multi administrative • entities, totally heterogeneous service infrastructures)	
Internet of things architecture technology	• Adaptation of symmetric encryption and public key algorithms from active tags into passive tags • Universal authentication of objects • Graceful recovery of tags following power loss • More memory • Less energy consumption • 3-D real time location/position embedded systems • IoT governance scheme	• Code in tags to be executed in the tag or in trusted readers • Global applications • Adaptive coverage • Context awareness • Object intelligence • Context awareness	• Intelligent and collaborative functions

(continued)

Table 1 (continued)

IoT-research needs	2011–2015	2015–2020	Beyond 2020
Communication technology	• Long range (higher frequencies—tenth of GHz)	• On chip networks and multi standard RF architectures	• Self configuring, protocol seamless networks
Network technology	• Grid/Cloud network • Hybrid networks • Adhoc network formation • Self-organizing wireless mesh networks • Multi authentication • Sensor RFID-based systems • Networked RFID-based systems—interface with other networks—hybrid systems/networks	• Service based network • Integrated/Universal authentication • Brokering of data through market mechanisms	• Need based network internet of everything robust security based on combination of ID metrics autonomous systems for nonstop information technology service
Software and algorithms	• Self-management and control • Micro operating systems • Context aware business event generation • Interoperable ontologies of business events • Scalable autonomous software • Software for coordinated emergence • (Enhanced) Probabilistic and non-probabilistic track and trace algorithms, run directly by individual "things" • Software and data distribution systems	• Evolving software • Self-reusable software autonomous things: • Self configurable • Self-healing • Self-management • Platform for object intelligence	• Self-generating "molecular" software • Context aware software

(continued)

Table 1 (continued)

IoT-research needs	2011–2015	2015–2020	Beyond 2020
Hardware devices	• Paper thin electronic display with RFID • Ultra low power EPROM/FRAM	• Polymer based memory • Molecular sensors	• Biodegradable antennas • Autonomous "bee" type devices
Power and energy storage technologies	• Printed batteries • Photovoltaic cells • Super capacitors • Energy conversion devices • Grid power generation • Multiple power sources	• Paper based batteries • Wireless power everywhere, anytime. • Power generation for harsh environments	• Biodegradable batteries
Security and privacy technologies	• Adaptation of symmetric encryption and public key algorithms from active tags into passive tags • Low cost, secure and high performance identification/authentication devices	• Context based security activation algorithms • Service triggered security • Context-aware devices • Object intelligence	Cognitive security systems
Standardization	• Privacy and security cantered standards • Adoption of standards for "intelligent" IoT devices • Language for object interaction	• Dynamic standards • Adoption of standards for interacting devices	• Evolutionary standards • Adoption of standards for personalized devices

Fig. 7 A few practical wired devices at home

energy harvesting scheme and that need to be sustainable and cost-effective. This is a challenge for the designer but at the same time it provides an opportunity to design electronics which will not be power hungry.

The challenges of IoT can be summarized as:

- Availability of internet at everywhere and at no cost
- Security issues
- Low-cost smart sensing system development
- Energy
- Computational ability
- Scalability
- Fault Tolerance
- Power Consumption
- Acceptability among the society

The success of IoT is entirely dependent on the availability of internet at everywhere. It is seen from Fig. 2 that except North America, Europe and Oceania, over 50 % of population of the rest of the world still do not have access of internet. Figure 8 shows the availability of internet in the New Zealand in relation to an internet service provider.

There is a huge need of internet growth to make it available to everyone in the world. Moreover, the availability of internet to people does not mean that the internet is available to every remote corner of the world. In order to make it available to every parts of the world there is a need of huge investment of providing infrastructure and resources. The investment from private companies is not expected unless there is a clear indication of making profit so the government needs to come in picture. Under the current economic situation it will not be a quick decision for any government to go for it. But with time it is expected to happen.

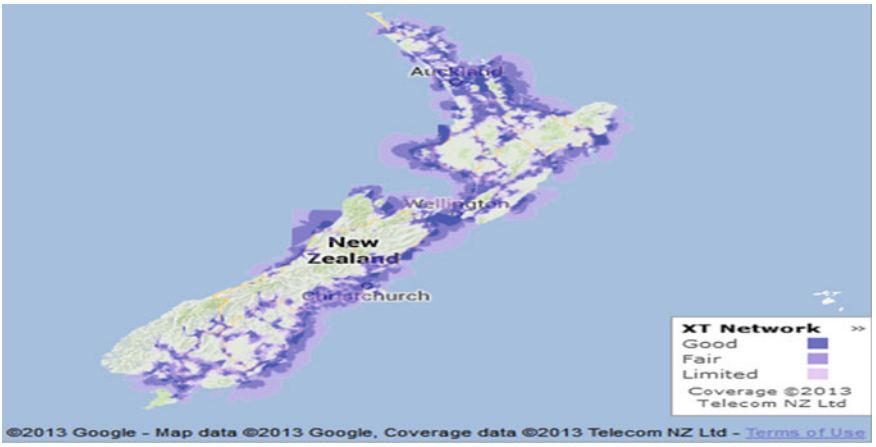


Fig. 8 Internet service availability in the New Zealand

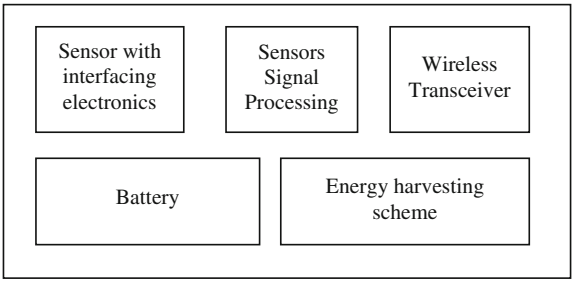


Fig. 9 The block diagram representation of wireless sensor node

A few countries such as Taiwan and China [15] are planning to have availability of internet at every parts of the country within a few years' time. It is expected that those action will inspire other countries to follow. Moreover, the internet needs to be available at free of cost and that may not be very easy to decide. It is not only the installation expenses, there is a running cost involved too. Along with the availability of internet there come the security issues. Under the current volatile situation where terrorism becomes a common word of people's life, accessibility of internet without any security may be a big ask. Then the design and development of internet without any security risk or threat may be huge challenge and that will gives an opportunity to the engineers. In relation to security to internet as well as in the wireless sensor network there may be malicious use. There can be use of sensor networks for illegal purposes, e.g. planting them in computers to extract private information. This is a challenge as this might happen at any time. So the challenge is to develop appropriate countermeasure.

The countermeasures could be to deploy sensor detectors to detect malicious sensor nets. It will not protect illegal sensor network deployment, but will make

attacks expensive. So that's an opportunity for the developers of IoT. Since IoT will have trillions of sensor nodes around us and in the environment, the most important challenge is to develop and fabricate them very cheap [16]. A sensor node consists of a few blocks as is shown in Fig. 9. It is possible to develop the node cheap if all the five blocks can be fabricated from one fabrication process but it has got huge challenges. At the moment it is not possible to fabricate everything in one chip but it offers a huge opportunity to the designers to achieve it. It is expected to have a very tiny sensor node with all the functionality as shown in Fig. 9 to be available for use.

One of the important challenges is the power supply for sensor nodes. Usually batteries are used to supply the necessary energy required for sensor signal processing and communication.

The following parameters should be considered for the selection of batteries:

- Type of battery (Alkaline, Lithium-Ion, NiCad, NiMH, Lead-Acid)
- Life-time (Ahr requirement)
- Environmental impact
- Cost
- Size
- Memory effect
- Safety issue.

The Opportunity here lies on the development of environmentally friendly new types of rechargeable batteries. The rechargeable batteries will need to be charged from different renewable energy sources. Figure 10 shows the established renewable energy sources, wind and solar. The choice of wind energy may not be limited for sensor nodes due to its initial investment and availability of wind. So there is a need of Accelerating the Implementation of Environmentally Friendly Renewable Energy Systems.

There are some challenges and consequently opportunities on the development of renewable energy sources for powering sensor nodes:

Challenge—The world looks increasingly towards implementing renewable energy systems → but the large potential for renewable energy has not been realized due to many barriers.

Opportunity—An important task is to identify the means to remove the economic, regulatory, and institutional disadvantages that make renewable energy less competitive than other conventional sources.

Most of the energy is consumed for data communication and different protocols are reported to address the issues [17]. Though there can be different strategies but from the design consideration and users' perspective it may be useful to follow a standard. In recent times there is a growing interest to follow IEEE protocols especially zigbee and Wi-Fi for different wireless applications [18–23]. A few projects have been developed based on zigbee protocols [24–27].

Fig. 10 Established renewable energy source

Fig. 11 Dumped computers

3 E-Waste

It may be good to have the environment surrounded by millions of sensors but we need to realize that the life-span of electronic devices is not long enough. Replacement needs to be done at regular interval in order to make the complete system active all the time. But it becomes a practical problem and it is a nuisance to find the place to dump the old electronic devices. Figure 11 shows the old computer stacked at the corridor of a university in New Zealand. Usually the computers of the academic staff are replaced at every four years. In the particular case the computers were lying on the corridor for over 3 months.

The designer must be very aware of the situation as the problem is severe: As per the UN report [28], a few statistical figures in US alone are:

Between 1997 and 2007, over 500 million personal computers have become obsolete-almost two computers for each person.

15,000,000 PCs become obsolete every year.

7,000,000 computers will end up stockpiled for at least 3 years.

750,000 computers will end up in landfills this year alone.

It seems "Every day nearly 1,000 computers and 1,400 cell phone sets are being sold in India amounting to 7,000 tons per year of e-waste and that too in major cities only". The report predicts that in South Africa and China, by 2020 e-waste from old computers will have jumped by 200–400 % from 2007 levels and by 500 % in India.

WSN contain hazardous and toxic materials that pose environmental risks if they are land filled or incinerated in future. Printed circuit boards contain primarily plastic and copper, and have small amounts of chromium, lead solder, nickel, and zinc.

In addition, WSN have batteries that often contain nickel, cadmium, and other heavy metals. Relays and switches, especially older ones, may contain mercury. Also, capacitors in some types that are now entering the waste stream may contain polychlorinated biphenyls (PCBs).

So guidelines to tackle the problems are [28]:

- Reuse is the environmentally preferable option for managing older electronic equipment. Extending the life of old products minimizes the pollution and resource consumption associated with making new products.
- Reuse also gives people who cannot afford new products access to electronic equipment at reduced or no cost.
- Electronic equipment which are too old and commercially and practically not viable for reuse or is broken beyond repair may be sent for disassembly i.e. salvaging parts, and selling reclaimed materials.
- Several electronic equipment, such as computers, monitors, printers, and scanners, contain materials suitable for reclamation and use in new products. These may include plastic, glass, steel, aluminum, copper, gold, silver, and other metals.

But re-use of old equipment will not solve the problem as it will just delay the process for a few years. This is a challenge for the engineering to come out with a solution how to deal with the e-waste situation.

We think here lays the opportunity for the engineers cum designers to design Biodegradable and Non-Hazardous sensor nodes. In future we need to explore materials which will not have any negative environmental impact and are suitable for fabricating sensor nodes.

More IoT-specific skills are needed for the next stage of development. A lack of IoT skills and knowledge among employees and management is viewed as the biggest obstacle to using the IoT more extensively [11]. To address these gaps, organisations are training staff and recruiting IoT talent, raising the potential for IoT talent wars. Others are hiring consultants and third-party experts, seeking to build knowledge and identify successful IoT business models. Moving executives and employees up the IoT learning curve should also help to ease the difficulty many firms experience in identifying IoT applications for existing products and services [11].

Big data, big privacy issues data, are thus a fundamental component of the IoT's future [11]. Fitting sensors to a potentially infinite number of "things" will generate untold amounts of new information [11]. For now, however, most business leaders

are confident that their organizations will be able to manage and analyze the data flowing from the predicted rapid expansion in IoT networks [11].

4 Conclusions

This chapter has briefly described different issues towards practical realization of Internet of Things. Though a lot of research going on towards technical implementation of IoT and will have positive impact on the society. At the same time many issues are arising which needs proper and appropriate attention to deal with them. The IoT should not be considered as a local development rather it is a global phenomenon and need to be treated with sincerity and priority.

References

1. http://www.internetworldstats.com/stats.htm. Accessed 10 Nov 2013
2. Kevin, A.: That 'Internet of Things' thing, in the real world things matter more than ideas. RFID J. (22 June 2009). Accessed 10 Nov 2013
3. http://en.wikipedia.org/wiki/Internet_of_Things. Accessed 10 Nov 2013
4. Analyst Geoff Johnson interviewed by Sue Bushell in Computerworld, on 24 July 2000 (M-commerce key to ubiquitous internet)
5. http://ec.europa.eu/information_society/policy/rfid/events/past/Info18022009/index_en.htm. Accessed 10 Nov 2013
6. Evans, D.: The Internet of Things: How the next evolution of the internet is changing everything, Cisco White paper. http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (2011). Accessed 10 Nov 2013
7. Guest editorial on special issue on internet of things (IoT): architecture, protocols and services. IEEE Sens. J. **13**(10), 3505–3510 (2013)
8. Atzori, L., Antonio Iera, b., Giacomo Morabito, C.: The internet of things: a survey. Comput. Netw. (2010). doi:10.1016/j.comnet.2010.05.010
9. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
10. Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S. (eds.): Vision and Challenges for Realising the Internet of Things. http://www.theinternetofthings.eu/sites/default/files/Rob%20van%20Kranenburg/Clusterbook%202009_0.pdf. Accessed March 2010
11. http://www.arm.com/files/pdf/EIU_Internet_Business_Index_WEB.PDF. Accessed 10 Nov 2013
12. http://news.techworld.com/networking/3476043/arm-report-businesses-look-make-money-through-internet-of-things-revolution/. Accessed 10 Nov 2013
13. http://www.economistinsights.com/search/node/internet%20things%20business%20index. Accessed 10 Nov 2013
14. Vermesan, O., Harrison, M., Vogt, H., Kalaboukas, K., Tomasella, M., et al. (eds.): The Internet of Things—Strategic Research Roadmap. Cluster of European Research Projects on the Internet of Things (CERP-IoT) (2009)
15. http://en.wuxi.gov.cn/web111/Events/InternetofThings/AboutInternetofThings/index.shtml. Accessed 10 Nov 2013

16. Mukhopadhyay, S.C.: Intelligent Sensing, Instrumentation and Measurements, Smart Sensors, Measurement and Instrumentation, vol. 5. ISBN: 978-3-642-37026-7 (Print) 978-3-642-37027-4 (Online) (2013)
17. Villalba, L.J.G., Orozco, A.L.S., Cabrera, A.T., Abbas, C.J.B.: Routing protocols in wireless sensor networks. Sensors **9**(1), 8399–8421 (2009). doi:10.3390/s91108399, ISSN 1424-8220, www.mdpi.com/journal/sensors
18. ZigBee Alliance: Understanding ZigBee gateway, ZigBee Document 095465r13, Sept 2010
19. Howitt, I., Gutierrez, J.A.: IEEE 802.15.4 low rate -wireless personal area network coexistence issuues. In: Proceedings of the IEEE-Wireless Communications and Networking Conference (WCNC 2003), vol. 3, No. 20, pp. 1481–1486 (New Orleans, 2003)
20. Angrisani, L., Bertocco, M., Foortin, D., Sona, A.: Assessing coexistence problems of IEEE 802.11b and IEEE 802.15.4 wireless networks through cross-layer measuremennts. In: IEEE Instrumentation and Measurement Technology Conference Proceedings (IMTC 2007), 1–3 May 2007, pp. 1–6
21. Khaleel, H., Pastrone, C., Penn a, F., Spirito, M.A., Garello, R.: Impact of Wi-Fi traffic on the IEEE 802.15.4 channels occupation in indoor environments. In: International Conferrence on Electromagneticcs in Advanced Applications (ICEAA '09), 14–18 Sept 2009, vol. 2, pp. 1042–1045
22. Khoshdelniat, R., Sinniah, G.R., Bakar, K.A, Shaharil, M.H.M, Suryady, Z., Sarwar, U.: Performance evaluation of IEEE802.15.4 6LoWPAN gateway. In: Proceedings of the 17th Asia-Pacific Conference on Communications (APCC), pp. 253–258 (2011)
23. Kushalnagar, N., Montenegro, G., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944 (2007)
24. Suryadevara, N.K., Mukhopadhyay, S.C., Wang, R., Rayudu, R.K.: Forecasting the behavior of an elderly using wireless sensors data in a smart home. Engineering Applications of Artificial Intelligence **26**(10), 2641–2652 (2013)
25. Kelly, S.D.T., Suryadevara, N.K., Mukhopadhyay, S.C.: Towards the implementation of IoT for environmental condition monitoring in homes. IEEE Sensors Journal **13**(10), 3846–3853 (2013)
26. Suryadevara, N.K., Mukhopadhyay, S.C.: Wireless sensor network based home monitoring system for wellness determination of elderly. IEEE Sensors Journal **12**(6), 1965–1972 (2012)
27. Suryadevara, N.K., Gaddam, A., Rayudu, R.K., Mukhopadhyay, S.C.: Wireless sensors network based safe home to care elderly people: behaviour detection. Sens. Actuators A: Phys. **186**, 277–283 (2012)
28. http://www.thegreenitreview.com/2010/02/un-reports-on-rocketing-e-waste-problem.html. Accessed 10 Nov 2013

Exploring Major Architectural Aspects of the Web of Things

Iván Corredor Pérez and Ana M. Bernardos Barbolla

Abstract A number of technological research lines around the feasibility and applicability of the Internet of Things are currently under discussion. Some of those research issues are dealing with the deployment of friendly smart spaces made of smart objects which are digitally augmented by means of RFID tags or embedded wireless sensor and actuator devices. The advantages of deploying heterogeneous ecosystems of smart things through web technologies have lead to the so called Web of Things paradigm, that relays on the Internet of Things principles. The opportunity of developing to develop new services and applications has driven research towards proposals that integrate isolate islands of networks based on the Internet of Things into the Web. In this context, the contribution of this chapter is twofold. On the one hand, it aims at detailing a Web of Things Open Platform for deploying and integrating smart things networks into the Web. The purpose of this platform is to expose the functionalities of sensor and actuator devices and their involved information model as a set of RESTful services to be retrieved from the Web. On the other hand,the Chapter describes a Resource-Oriented and Ontology-Driven development methodology, which enables easier the development and deployment of smart spaces. The feasibility of this holistic approach is demonstrated through a fully-implemented case study.

1 Introduction: Towards Smart Spaces for Human Beings

The first decade of XXI century have been very prolific in research focusing on making real Weiser's vision [1] on Pervasive Computing. Those early works led to

I. Corredor Pérez (✉) · A. M. Bernardos Barbolla
Grupo de Procesado de Datos y Simulación, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain
e-mail: ivan.corredor@grpss.ssr.upm.es

A. M. Bernardos Barbolla
e-mail: abernardos@grpss.ssr.upm.es

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_2,

a number of exciting research results which have contributed to get more and more tiny devices connected to each other and to the Internet. Different technologies of very diverse fields have been involved on enhancing the communication capabilities of smart objects, e.g. RFID/NFC tags, wireless sensor and actuator embedded devices, low-energy communication protocols, etc. All these technologies are subsumed behind a concept that encompasses a lot of research subjects: the Internet of Things, which was conceived with the promising idea of improving the connectivity of constrained and embedded devices to the Internet.

Originally, the application of Internet of Things (IoT) was focused on managing information through RFID tags [2, 3] such as goods tracking, management of everyday objects, automatic payments in markets and military applications. The evolution of Micro-Electro-Mechanical (MEMS) technologies have fostered the evolution of the Internet of Things by means of the miniaturization of hardware components, i.e. wireless transceivers, sensors, actuators, microcontrollers, etc. Those achievements have motivated many proposals for improving the software of embedded devices in several aspects, e.g. protocols, operating systems, architectures, etc., which have been essential in order to optimize Machine-to-Machine (M2M) interactions, providing a much more proactive character to IoT-based applications. In this sense, current paradigmatic IoT applications are designed to create smart spaces which are made of thousands with of smart things autonomous capabilities.

While the implementation of the Internet of Things is exponentially growing, the Internet of Services is already a reality. One the pillars of the Future Internet is focused on how to integrate *real-world* services provided by the Internet of Things into mash-ups of traditional services. This new Internet order will be addressed under two viewpoints specifying two interaction dimensions, *horizontal* and *vertical* approaches, involving Machine-to-Machine and Human-to-Machine (H2M) interaction, respectively. The second one will also involve interactions among Machines and processes which could be running on external entities connected to other networks. Recently, concerns have raised on how to optimize the management of such vertical interactions which will generate enormous amounts of information in terms of *heterogeneity*, *reusability*, *scalability* or *accessibility*.

The major problem that hinders to deal with those issues is the wide use of *ad hoc* and monolithic designs that difficult the convergence of different applications through a generic and open solution. To make feasible and reusable IoT-based technologies, an open infrastructure is needed, which allows accessing heterogeneous sensors and actuators devices following the Web and the Internet standards. Future applications for IoT-based networks will require a significant improvement in reusability of deployment resources in order to be available for many high-level entities, e.g. smart phones apps, Web sites, expert systems, etc.

Solid proposals to deal with the above mentioned challenges have arisen behind the *open platform* concept. This concept involves a set of specific issues that can fairly facilitate rapid prototyping and deployment of large IoT-based networks. Although its major feature is the openness of interfaces (i.e. well documented and public interfaces), it involves other ideas that facilitate reusability, extension and reimplementation of functionalities for which the platform has not been initially designed.

It is important to highlight that this concept does not necessarily imply open source results, but a set of public operations that are specified in an API (Application Programming Interface). Multiple implementations of a same API can be offered to be used for different technological platforms (e.g. different mobile operating systems) or programming languages (e.g. JAVA, C, C++, etc.). A new trend in open platforms is to provide some interesting tools to integrate IoT networks with well-known Web technologies by means of a category of cloud computing services: the Platform as a Service (PaaS).

The PaaS model allows service consumers to easily develop and deploy value-added services by using workbench tools and/or libraries provided by a vendor; this helps the costumer to spend few resources for management and maintenance of large infrastructures of hardware and software. PaaS-based solutions have commonalties and specific characteristics that differentiate each other. Common features are, among other, the provisioning of a deployment environment, monitoring dashboard, as well as communication mechanisms to exchange information between the platform and the client. Computing resources transferred to client accounts usually depends on the scalability and Quality of Service required by the applications using the platforms.

The conjunction of the open platform with the PaaS paradigm has gained popularity among researchers who are focused on the integration of the IoT into the Web. PaaS is strongly related to the Web of Things (WoT). Generally speaking, the WoT tries to bridge the gap between embedded devices and the Web in order to integrate very different and isolated IoT networks with users environments through Web technologies, as well as with each other. In order to reach the major objective of the WoT, it is needed to adapt traditional Web technologies to create open interfaces, representation formats, discovery and communication protocols, or information retrieving mechanisms that facilitate integrating real-world entities including on the IoT. For instance, a common aspect of these approaches is the use of the Representational State Transfer (REST) architectural style [4] and the JavaScript Object Notation (JSON) for representation formats. In REST-based approaches, real world entities are identified by Unique Resource Identifiers (URIs) and their associated information can be accessed by means of invoking HTTP methods. These features (among others) make easier the development of IoT-based mashups aiming at gathering, processing or aggregating massive amounts of data from sensors and other virtual data sources. The design of approaches under these precepts should motivate the emerging of open platforms focused on abstracting every concept related to IoT in order to build the pillars for the future WoT.

1.1 Relevant Contributions of this Research in the Fields of the IoT and the WoT

Obviously, the emerging of novel approaches in the WoT needs to tackle with new and efficient implementations that move theoretical proposals into practical solutions in a short term of time. This Chapter compiles our recent research work [5–7], focused on providing a holistic solution that facilitates the design, development and deployment

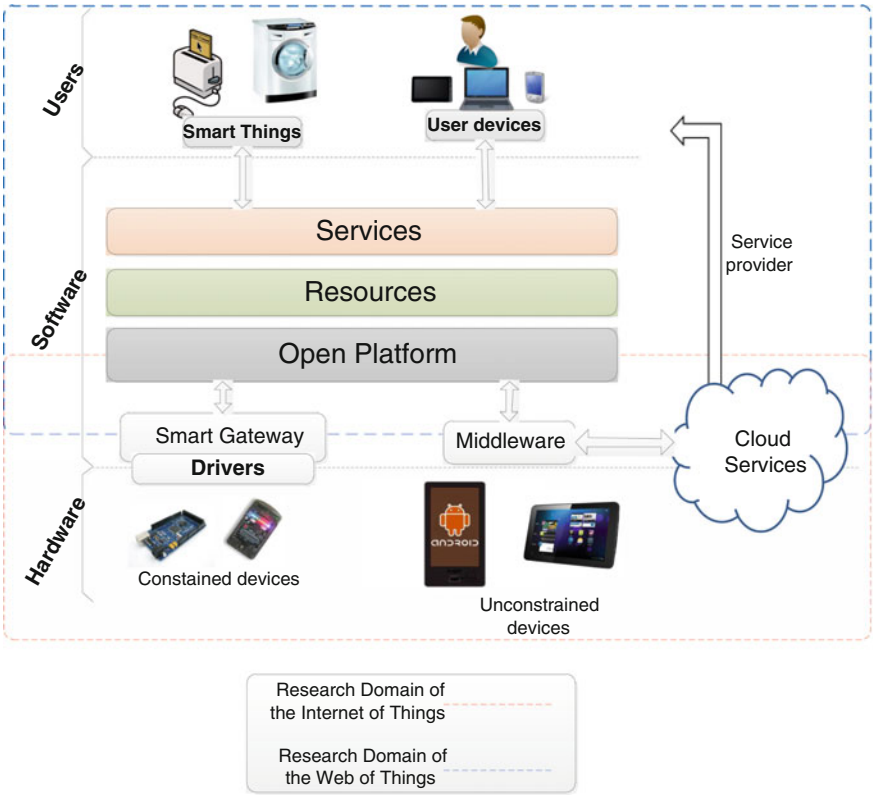


Fig. 1 The proposed reference architecture for the IoT and WoT research domains

of enriched smart environments according to the IoT and WoT paradigms, specially on the latter.

In order to efficiently address this issue, we defined a reference architecture which establishes a principle guide for our design decisions. This reference architecture gives a general perspective of the major areas to be developed in the research domains of the IoT and the WoT, i.e. a set of unified concepts and their relationships. It is important to highlight that a reference model has to be independent of issues as standards, technologies or implementations [8]. The proposed reference architecture for the IoT and the WoT research domains is shown in Fig. 1.

The architecture in Fig. 1 is defined according to three major layers. The lowest layer is composed of the hardware ecosystem. This layer is classified into two groups of devices: (i) *constrained devices*, i.e. those devices that have limited features and, thus, they have to outsource some processes to be run by external devices (usually Smart Gateways) which expose their functionalities to clients; (ii) *unconstrained devices*, i.e. those devices that have enough resources to run the necessary processes or middleware components to provide directly functionalities to clients through a

platform or third part cloud services. Concepts managed in this layer exclusively belongs to the Internet of Things domain. The next layer is the software layer which has to be supported by an open platform. The role of this platform is important since it has to provide a mechanism to set up the functionalities of the underlying hardware as resources (e.g. sensor, actuator, processing capabilities, etc.) and finally, to compose and orchestrate them in order to build simple or complex services. The software layer aims at implementing the research domain of the Web of Things, overlapping with the Internet of Things domain in some low-level issues as protocols, drivers or communication standards. Finally, the user layer is composed of clients which are going to consume the services exposed by the open platform. Those clients can be human using user devices (e.g. smart phones, tablets, laptops, etc.) or machine users as smart objects that might request services to perform high or low level tasks (e.g. to turn on an ambient light or analyze the capabilities of peer entities to compose services for making complex smart environments).

As commented previously, the proposal described here is focused on achieving a holistic solution for the IoT and the WoT. The research works that led us to our approach can be summarized into the following points:

- *The Web of Things Open Platform (WoTOP)*: The first research contribution included in this Chapter describes a novel open platform, called the Web of Things Open Platform. WoTOP allows interconnecting smart objects, business process and users. This platform is designed on the well-known Web technologies which enable vertical interaction. From a general perspective, our proposal addresses the major challenges to achieve a consistent business method for deploying services over an underlying IoT ecosystem consisting of networks of embedded sensors and actuators devices.
- *Methodology for Rapid Deployment of Smart Spaces*: Currently, people who deal with the WoT approaches need to work with specific software and hardware platforms as well as web technologies. The Do-it-Yourself movement is motivating the design of *creation environments* to enable no-technicians in the development and deployment of WoT-based smart spaces composed of embedded sensor and actuator devices and distributed business logic. We have designed a development methodology in order to alleviate the development process of complex and large smart spaces; our objective has been twofold: (i) to facilitate the integration of sensing and actuating functionalities into everyday objects and (ii) to enable a heterogeneous ecosystem of devices to integrate into the Web. The result has been the Resource-Oriented and Ontology-Driven Development (ROOD) methodology which is based on the Model Driven Architecture (MDA). This methodology aims at enabling the development of smart spaces through a set of modeling tools and semantic technologies that support the definition of the smart space and the automatic code generation for a specific hardware platform.
- *Prototyping of Real Smart Spaces*: Prototyping is an essential stage that allows proving, validating and consolidating theoretical concepts through their application in real deployments. For that purpose, we designed and deployed a smart shop which involved a number of different entities to offer enriched services to

hypothetical users. This smart space was used to validate most of the concepts and tools proposed in this Chapter.

The rest of the Chapter is organized as follows. Section 2 discusses open research challenges related to the IoT and the WoT, highlighting proposals that address the problem from both industry and academical perspectives. Section 3 and 4 compiles the major research contributions of this Chapter, i.e. the Web of Things Open Platform and the Resource-Oriented and Ontology-Driven, respectively. Section 5 describes the case study deployed for validating our contributions that were theoretically explained in previous sections. Conclusions of the Chapter and a outlook to research challenges are included in Sect. 6.

2 Open Research Challenges and Background

2.1 Summary of Research Challenges

As was commented in the previous Section the convergence of IoT, WoT and PaaS present different challenges when integrating constrained networks into the Internet by using internetworking techniques and middleware architectures. These challenges can be classified as following:

A. *Integration of heterogeneous IoT-based networks*: Usually IoT-based networks need to link ecosystems of technologically heterogeneous devices. This diversity makes difficult the integration of things into the Web. Some IoT platforms as Cosm [9] or Paraimpu [10] have designed their own strategics to integrate data streams from open source devices as Arduino [11]. Other platforms, as Sensinode [12] or SmartThings [13], have created their own hardware products which are compatible with their respective IoT platforms. The most appropriate solutions would be those offering high-level mechanisms to integrate additional types of embedded devices, based on both proprietary and open source hardware. The objective is to create a larger scene attracting developers specialized in most popular sensor and actuator embedded technologies (e.g. Crossbow, Zephyr, Sun SPOT, Arduino, etc.). To this aim, it is necessary to simplify the mechanism to integrate things as much as possible by hiding to developers the technical aspects of the platform.
B. *Adaptive scalability*: Typically, WoT platforms are provided through cloud services with high capabilities to manage milliards, even thousands, of things per user. This kind of services can scale a lot and they are only limited to the available resources of the servers supporting them. Apart from this, WoT platforms for homes and small businesses are being more and more popular since they allow users to take full control over security issues [13]. Thus, this is an important property for services managing sensitive information, e.g. those based on eHealth or home security. The latter type of platforms is deployed to work at smaller scale using *hubs* (also called *Smart Gateways*) through which any necessary devices

can be connected to a smart space. However, these Smart Gateways are limited in hardware resources, so their scalability is also limited. Thus, it is needed to explore a trade-off for small and medium scale deployments (e.g. in factors as number of deployed sensors/actuators or rate of generated events) in order to optimize the performance of these Smart Gateways.

C. *Data filtering and alert notification*: When managing a plethora of things generating big amounts of data it is necessary to implement mechanisms to filter raw data in order to optimize the processes that will store and dispatch information to the clients. Simple filtering mechanisms are typically defined by means of first-order rule engines or data source combination (e.g. aggregation or fusion of data sources) by applying functions to aggregate them. Data filtering techniques are very often related to the detection of events. Other approaches take advantage of Semantic Web techniques in order to disseminate data by performing complex inferences [14].

   Ideally, the design of open platforms should include an event-driven subsystem based on the *publisher/subscriber* communication paradigm. For example, Cosm provides an unsophisticated filtering mechanism based on simple rules in order to detect events and send notifications to a Twitter account. This functionality should be implemented through generic event-driven mechanisms in order to dispatch notifications to any subscribed application or service.

D. *Sharing of knowledge resources*: Sharing knowledge resources is the cornerstone of collaborative networks with different objectives, e.g. scientific, technological or social. Consequently, this aspect has to be taken into account when designing WoT open platforms. These platforms should provide mechanisms to share resources in collaborative networks or just to distribute such resources among users that may be interested in them.

   A very accepted proposal to offer resources sharing is the called *API key* [15]. These API keys allow accessing specific resources (e.g. sensor streams, actuator control or monitoring tools) only for trusted users. Additionally, those resources can be available both permanently and for a period of time. A recent proposal [16] tries to take advantage of social networks (e.g. Twitter, Facebook, LinkedIn, etc.) and their open Web APIs in order to share smart things among trusted users. This proposal is based on an authentication proxy, called Social Access Controller (SAC), which allows publishing and sharing smart things among user groups registered in some social network with specific credentials.

E. *Development and deployment of services*: The own nature of an open platform is to provide a public Application Programming Interface (API) that allows quick prototyping of complex applications by creating mashups consisting of a variety of both sensor data sources and actuator control points. Ideally, open platforms have to provide natively common services for many applications (e.g. data visualization or localization services). However, an enriched API should offer a set of development tools to application developers to create their own additional applications for processing and visualizing data. Blackstock et al. [17] propose WoTKit. This is an holistic solution, that offers a WoT architecture and a graphical toolkit allowing rapid development of IoT mashups by means of well known web technologies.

New trends are focusing on mapping APIs for IoT services into RESTful services in order to take advantage of Web techniques. Simon et al. [18] propose a toolkit that facilitates the integration of IoT-based smart things into the Web by defining RESTful services, which are mapped over embedded sensor functionalities. Zhenyu Wu et al. [19] work on the concept of Gateway as a Service (GaaS) in order to design a framework which seamlessly integrates third party embedded devices into the Internet by modeling RESTful services and mapping Web Services over them. Ideally, a development methodology can be offered in order to facilitate to developers, even those with no technical skills, to develop advance application and services from mashups of models of IoT-based environments. For example, a development methodology based on the Model Driven Engineering has already proposed by authors of this work [5]. The methodology offers development tools for rapid prototyping of WoT-based complex systems.

After the previous review of open challenges for IoT and WoT platforms, next Section provides an overview of the initiatives and projects that are coordinating their efforts to gather and conduct research results in different fields to reach standards and recommendations.

2.2 *Background: Standardization Initiatives and Other Industrial and Academical Projects*

From five years to date, many approaches have emerged with the aim at implementing technology agnostic solutions related to IoT or M2M research fields. This fact has raised the need of managing hardware and software interoperability. Many companies behind this IoT or M2M services belong to IPSO Alliance that was launched in 2008 as a non-profit organization to coordinate an initiative to establish the IP as standard network protocol for connecting smart things by means of carrying out common marketing efforts. Currently, the IPSO Alliance is composed of around 50 companies, including Google, Cisco, Ericsson or Alcatel. Additionally, there are successful commercial solutions that have taken advantage of the standards and recommendations specified by the partners of IPSO Alliance, for instance ThingWorx [20], AirVantange [21], Axeda [22], or SmartThing [13].

Aside from commercial approaches, academia is actively contributing to IoT-related research. In fact, the concept of Internet of Things was conceived in the MIT Auto-ID center. Auto-ID labs, which are spread around the world in seven countries (US, UK, Switzerland, China, Korea, Australia and Japan), work on architecting the IoT together with EPCglobal [23]. In Europe, part of the academic research work focused on IoT has been developed within projects funded by the EU Comission (under the Framework Programmes, ITEA or Artemis initiatives). Many of these projects are coordinated by the European Research Cluster on the Internet of Things (IERC) [24]. The IERC was founded within the FP7 in order to manage the wide range of results and applications from the European projects on IoT, and to coordinate

the on-going activities facilitating knowledge sharing, not only at European level but also at a global level. More than 30 EU-funded projects (some are still in execution) have been involved on the IERC initiative, among them AMI-4-SME, SENSEI, IoT-i, IoT-a or DiYSE [25].

Recently, some initiatives based on web-centric open platforms have come up contributing with interesting mechanisms intended to integrate isolated IoT-based networks into the Internet through cloud services. For instance, Cosm [9], EVRYTHNG [26], Paraimpu [10] or ThingSpeak [27] provide services to integrate any sensor and actuator device, or tagged object (e.g. with a RFID tag or QR code) into each other and into the Internet. These platforms usually provide RESTful interfaces that enable access to the information gathered from the mentioned sources through URIs that identifies each data stream, usually called *feeds* or *channels*. The most usual set of services provided by these platforms comprise processing, sharing, mashuping and visualization of sensors and actuators. Each one of these platforms are characterized by different features that contributes to establish the pillars of the current and the future Web of Things. However, any of the mentioned projects have proposed an integral solution which tackle the management of the Internet of Things, providing adaptive communication and information models for different business cases, as well as tools and methodologies to develop and deploy rapidly services involving smart things.

3 The Web of Thing Open Platform

As it was commented in Sect. 1, our contribution merges several research concepts (IoT, WoT, PaaS and development methodologies) which are certainly interrelated among them. We have made the most of this situation in order to reach a holistic solution to build modern smart spaces, enabling the integration of different interaction methods and based on the paradigm of Pervasive Computing (e.g. lightweight protocols, context-aware systems or localization-aware services). The latter is changing the way in which we currently understand Internet. The Future Internet will provide a wide range of *real-world* services supported by networked smart things setting up complex smart spaces on a global scale.

Our first approach aims at providing an open platform to provide developers with rapid and easy prototyping tools to plug smart things into the Web. This proposal is called the *Web of Things Open Platform* (WoTOP). The architectural model of the WoTOP is based on five design principles taking into account the research challenges mentioned in Sect. 2.1:

(i) Integration of isolated IoT-based networks composed of embedded devices implementing specific stack protocols that are not compatible with IP/TCP (*Challenge A*). The integration process is supported by a set of synchronized Smart Gateways which implements both IP/TCP stack and specific protocols depending on the embedded devices plugged to it.

(ii) Discovery, configuration and management of heterogeneous ecosystems of smart things, i.e. things equipped with sets of sensors and actuators as well as processing capabilities that are capable of performing inferences from its own contextual information (*Challenge A and B*). All those features are exposed to other entities in the smart space (peers, external or even humans).
(iii) Usage of an information model and standardized protocol based on the Representational State Transfer (REST) in order to offer a public API accessible to as many clients as possible (*Challenges D and E*). This API provides access in a uniform way without concerning neither communication issues nor data representation formats.
(iv) Support for different communication modes to deliver information to clients according to their information consumption requirements (*Challenge C*). The most typical communication modes are supported: *on-demand* and *event-driven*. Both communication modes will be accessible through API mentioned before.
(v) Management of massive amounts of data generated by large ecosystems of Smart Objects (*Challenge D*). This design principle is supported by an information model that enable persistence techniques that aims at optimizing the massive storage of data generated in smart spaces.

The following Section describes the WoTOP architecture supporting the above mentioned design principles.

3.1 Platform Architecture Overview

The WoTOP is based on a layered architecture composed of three layers interconnected among them through well-defined interfaces (see Fig. 2). In turn, every layer of the architecture includes subsystems and components that fulfil specific objectives. These subsystems and components are explained in the following.

A. *Internet of Things Ecosystem Layer*: The Internet of Things Ecosystem Layer enables the WoTOP to connect with *things* coexisting in the *real-world*. These things are usually *smart*, and they are characterized according to different natures and objectives within the *smart space* they belong to. Physically, a smart thing consists of one or more embedded devices equipped with sensors and/or actuators that enable it to play a role in the smart space, coexisting with other smart things to fulfil a given objective.
Smart spaces are usually composed by a wide variety of smart things. In order to deal with this technological heterogeneity issue, the Internet of Things Ecosystem Layer implements a mechanism based on the *Plug&Play* paradigm. Through this mechanism, devices of very different technologies can be plugged to the WoTOP and discovered *on the fly*, in order to expose smart thing capabilities to the higher layers of the architecture. The components implementing drivers for any technology are called *adapters* , and they are stored in a repository.

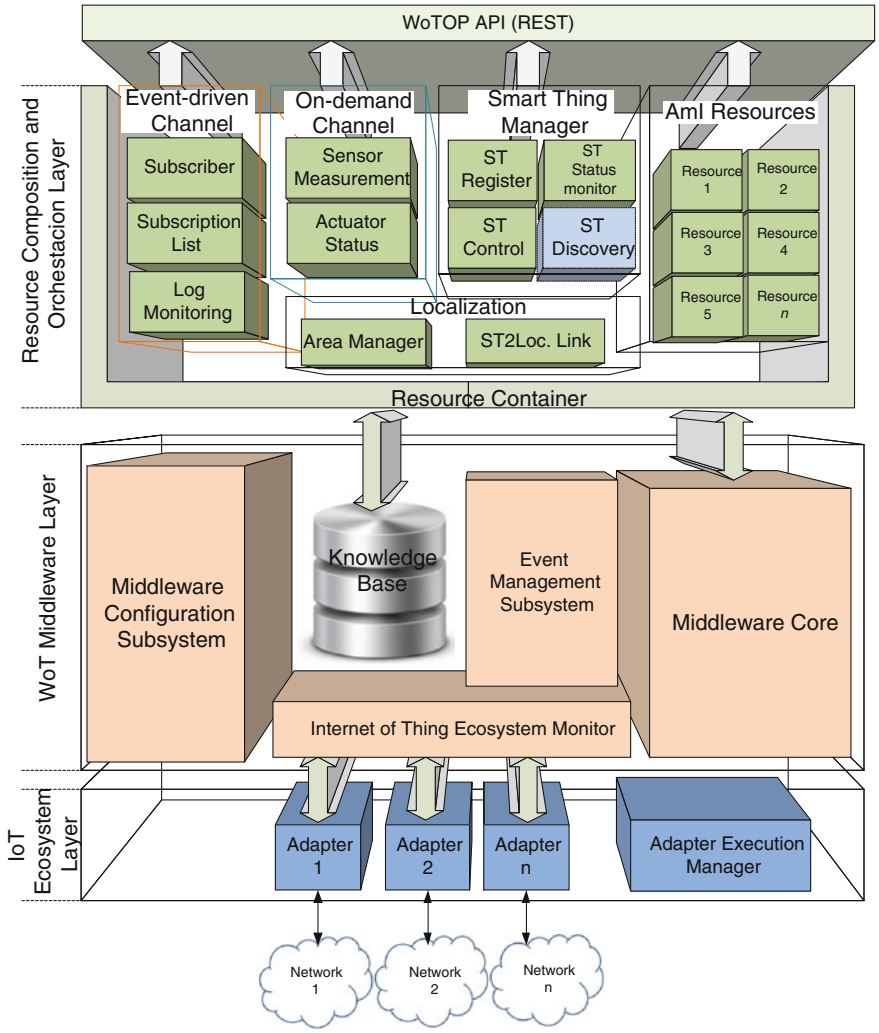


Fig. 2 The layered architecture of the Web of Things Open Platform

This mechanism is supported by an OSGi framework that allows scheduling the life-cycle of adapters, i.e. load, unload, run and stop. To integrate the adapters with OSGi, they are implemented as *bundles* that are stored in a OSGi Bundle Repository (OBR). When a new device is plugged to a WoTOP Gateway, the needed bundle is retrieved and loaded automatically.

B. *The Web of Things Middleware Layer*: The functionalities of this layer allow designers and developers of WoT services to model, implement and deploy complex smart spaces and expose them according to the WoT paradigm. The Web of Things Middleware Layer is composed of the following subsystems:

(1) *Internet of Things Ecosystem Monitor*: This component aims at listening data from the adapters deployed on the Internet of Things Ecosystem Layer. Basically, it registers every device connected to the WoTOP as well as its own context information and events generated by them. From the information collected from all those devices, this component creates and updates a registry called *Alive Smart Thing Register* (ASTR). The ASTR consist of a table of devices' identifiers associated to smart things, including relevant information about its status (i.e. battery level, current localization or the owner) and a cache with historical information related to the environment context in which they are deployed (i.e. sensor measurement, high level context information or actuator status). The information temporally stored in that cache is periodically dumped into a *Knowledge Base* which manages its persistency. The process commented before is shown in the Fig. 3.
The information stored in the cache of the ASTR is also processed by the Event Management Subsystem in order to detect events, as it is explained below.

(2) *Event Management Subsystem*: This subsystem was designed to enable WoTOP to manage asynchronous communications for those clients that need to consume context information according to specific requirements. This subsystem includes different mechanisms to send notifications or events to the clients. Two *event-driven* mechanisms are supported: (i) *condition-based*: events are dispatched to the clients when context information gathered from smart things matches some condition set up by the clients, e.g. temperature in room A $>$ 22 °C AND humidity in room A $<$ 35 RH; (ii) *contract-based*: If the context information available in ASTR's cache is relevant to the client and said information is updated, then events are dispatched to clients periodically, e.g. the location of a smart thing is notified to a client every 2 s. Both types of mechanisms can be initialized by clients through subscriptions that are defined and sent to the WoTOP accordingly to their interests.
Once the client has sent a subscription expressing an interest, a *Webhook* is instantiated in the client application . The *Webhook* allows opening an entry to receive and handle messages from WoTOP. A *Webhook* consists of defining a HTTP callback to deliver messages asynchronously to the client; it is identified by an URI that will be used by the *Event Management Subsystem* in order to POST events to the client interested in them. The Event Management Subsystem keeps two subscription tables, one per type of communication mode supported by it: (i) *Condition-based subscription table* and, (ii) *Contract-based subscription table*. The former is intended to store subscriptions to events that have to be sent if a condition occurs. The latter is intended to store subscriptions to events that have to be sent according to contract method. Processes and components involved in event dispatching are different for each type of event. Theses processes are shown in the Fig. 4.
As Fig. 4 shows, the data to be processed by the *Event Management Subsystem* always have the same origin, the *Internet of Things Monitoring Sybsystem* which gather them in a orderly manner. Then, the data are temporary buffered before being processed by the corresponding module of the *Event Management*

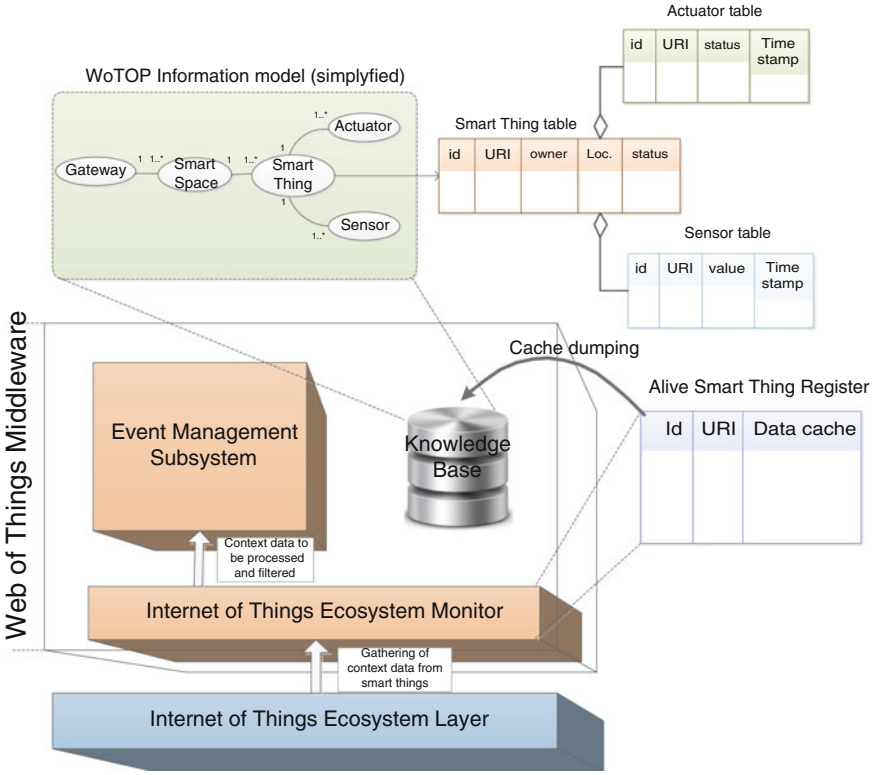


Fig. 3 Processing of the context data gathered through *the Internet of Things Ecosystem Monitor*

Subsystem. On the one hand, the buffered data are filtered by the *Rule Analyzer* (see Fig. 4a) which checks subscriptions that were configured according to the conditional mechanism. If one or more subscriptions match some Context data, then an event is detected and delivered to the client that is interested in it. On the other hand, the buffered data are periodically polled by the *Scheduler of Context Data Notifications* (see Fig. 4b) according to the subscriptions based on the contract mechanism. If more recent data are found and there is any subscriber to them, then they are dispatched to the subscribers.

(3) *Middleware Configuration Subsystem*: This subsystem aims at preparing and configuring the execution environment of the Web of Things Middleware. An optimized configuration depends on the features of the device that is going to run the middleware. The standard parameters that can be configured through this subsystem are related to the location and credentials to access the knowledge base, maximum number of clients that can access to WoTOP services simultaneously or maximum number of subscriptions that can be established in the subscription table.

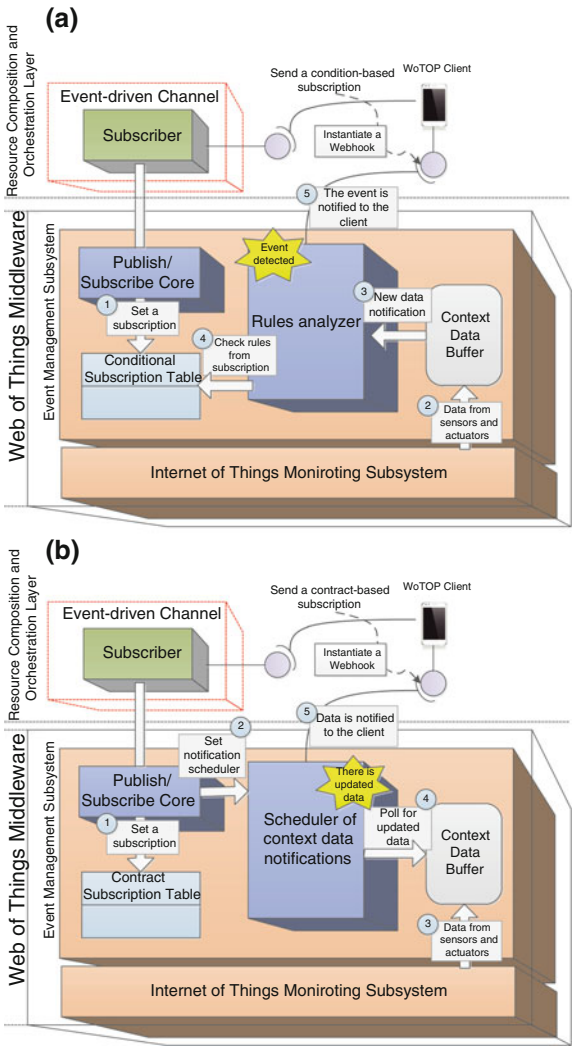


Fig. 4 **a** Event-driven communication based on the condition technique; **b** Event-driven communication based on the contract technique

(4) *Middleware Core*: This is an essential module of the Web of Things Middleware, whose major objective is to expose the interfaces of the components building the WoTOP's architecture, in order to facilitate the necessary communication among them. This kind of communication is based on OSGI services. Additionally, the Middleware Core can dynamically create REST end-points according to the components deployed on the *Resource Composition and Orchestration Layer*. The latter functionality is a cornerstone within the WoTOP architecture since it facilitates exposing RESTful services to the clients.

Table 1 REST interface of the Event-driven channel of the WoTOP

Access method and input message	Description

• **Create a subscription**
`POST /{gw-id}/eventmanager /subscriber`
`Content-type: application/json`
• **Update a subscription**
`PUT /{gw-id}/eventmanager /subscriber/{subs-id}`
`Content-type: application/json`
- **Conditional**
`{`
`"consumer": {"URI": <String>, "Id": <String>},`
`"producer": {"URI": <String>, "Id": <String>},`
`"type":<String>, //Contitional (1)`
`"end":<long>, //End time of the subscription`
`"event": [{ "value": <String>,`
`"key": <String>,`
`"unit": <String>,`
`"operator": <String>}, ...]`
`}`
- **Contract**
`{`
`"consumer": {"URI": <String>, "Id": <String>},`
`"producer": {"URI": <String>, "Id": <String>},`
`"type":<String>, //Contract (2)`
`"start":<long>, //Start time of the subscription`
`"end":<long>, //End time of the subscription`
`"samplingPeriod":<long>,`
`"event": { "key": <String>,`
`"unit": <String>}`
`}`
	These requests create (or update) a subscription for a specific event. The dispatching mechanism of to the client will depend on the type of subscription done previously (by condition or by contract).
	
	The reply to this request will content a unique subscription code which is needed for update or unsubscribe any subscription.
	
	`consumer` and `producer` parameters indicate the root URI (`URI`) and their id (`Id`) identifying uniquely the event consumer and producer, respectively. Event consumers must enable a RESTful handler in order to manage event callbacks which are based on POST messages (Webhooks).
	The format of an event is the following:
	`{`
	`"producer": {"URI": (String), "Id": (String)},`
	`"type": (String), //Conditional (1) or Contract(2)`
	`"timestamp": (long), //When the event was generated`
	`"payload": [ {"key": (String), "unit":(String),`
	`"value": (String) }, …]`
	`}`
	
	The event payload contains useful information of the event (structured in triples):
	-Key: Concept of the event.
	-Unit: Unit or type of the measured value.
	-Value: Value of the measurement that trigger the event.

• **List subscriptions of a consumer**
`GET /{gw-id}/eventmanager/subscriber?consumer= <Consumer URI>`
	This request returns a list of subscriptions related to an event consumer.
	The format of the returned messages is an array of JSON documents as follows:
	`[{`
	`"subscriptionCode": <String>,`
	`"subscription": <subscription format>`
	`}, …]`

• **Delete a subscription (unsubscription)**
`POST /{gw-id}/eventmanager/unsubcriber`
`Content-type: application/json`
`{`
`"consumer": {"URI": <String>, "Id": <String>},`
`"eventType":<string>`
`"subscriptionCode":<string>`
`}`
	A POST request has to be sent to the specific URI identifying a subscription which has to be deleted form the subscription table. This method attaches a JSON document that indicates the URI and Id of the event consumer, the type of the event and the code of the specific subscription to be deleted.

C. *Resource Composition and Orchestration Layer*: As mentioned in the previous point, the Resource Composition and Orchestration Layer is a strategic subsystem of the WoTOP architecture since this layer enables WoTOP to provide enriched RESTful services to manage smart spaces. Resources provided by this layer are RESTful services which are implemented by components. Those components are deployed on the *Resource Container* which facilitates the management of their lifecycle. The lifecycle of the components deployed in the *Resource Container* is planned in four stages: initializing, running, stopping and destroying. The first (initializing) and the fourth (destroying) stages allocate and free memory for the operation of the component, respectively. The second (running) and the third (stopping) stages are mainly focused on opening and closing the REST end-points that facilitates the access to the resources that are offered by the components. Those resources can be accessed through one or more HTTP methods according to the REST paradigm, i.e. GET, PUT, POST or DELETE. As an example, the Table 1 shows the REST interface of the Event-driven channel.

The components provide atomic services that can be offered to the clients without

needing support from other components. Additionally, the resource container provides mechanisms to perform composition and orchestration of atomic resources in order to provide complex resources involving two or more atomic resources. As shown in Fig. 2, WoTOP deploys some resources by default; these are classified according to the following groups: *Event-driven channel, On-demand channel, Smart Thing Manager* and *Localization*. These resources are deployed with the aim of exposing essential services of the platform that allow accessing middleware functionalities. The Event-driven and On-demand channels provide resources to the clients to allow them to consume information and to handle events which are generated by smart things and stored according to the information model shown in Fig. 3. The *Smart Thing Manager* resource provides functionalities for integral management of smart things composing a smart space. For instance, functionalities are offered to register a smart thing in the ASTR or to monitor the status of smart things registered in the ASTR through RESTful interfaces.

Additionally, WoTOP provides the *Localization* resource, which is deployed to provide positioning information that may be necessary to manage a given smart space. The major functionalities offered by this resource allow dividing the smart space in areas and linking smart things to those areas by means of semantic annotations. This resource could be potentially used by clients or external services using WoTOP.

Finally, the *Resource Composition and Orchestration Layer* allocates part of the *Resource Container* to deploy additional resources, so called *Ambient Intelligent Resources*. These resources are designed to extend the capabilities of the WoTOP according to the open nature of its architecture, by developing and deploying resource components that may take advantage of the available functionalities in the platform.

3.2 Development and Deployment of Smart Spaces using WoTOP

As shown in Fig. 5, WoTOP is hosted by one or more Smart Gateways that can be synchronized with each other in order to create domains of Smart Gateways. A domain of Smart Gateways aims at sharing knowledge resources (e.g. information collected from smart things or other AmI services) to offer them to application clients through a Platform as a Service (PaaS) paradigm. This paradigm allows providing services uniformly and seamlessly regardless of the heterogeneity and extension of the underlying smart space in terms of hardware and functionalities. According to the openness of WoTOP, some procedures were designed to facilitate the development and deployment of smart spaces.

In this subsection, we provide a brief introduction of the steps to follow to create a varied ecosystem composed of sensor and actuator devices, logic entities and human users. Specifically, three aspects are tackled in this subsection: (i) integration of additional sensor and actuator devices into WoTOP, (ii) development of new resources to offer new functionalities to client applications, and (iii) developmentof

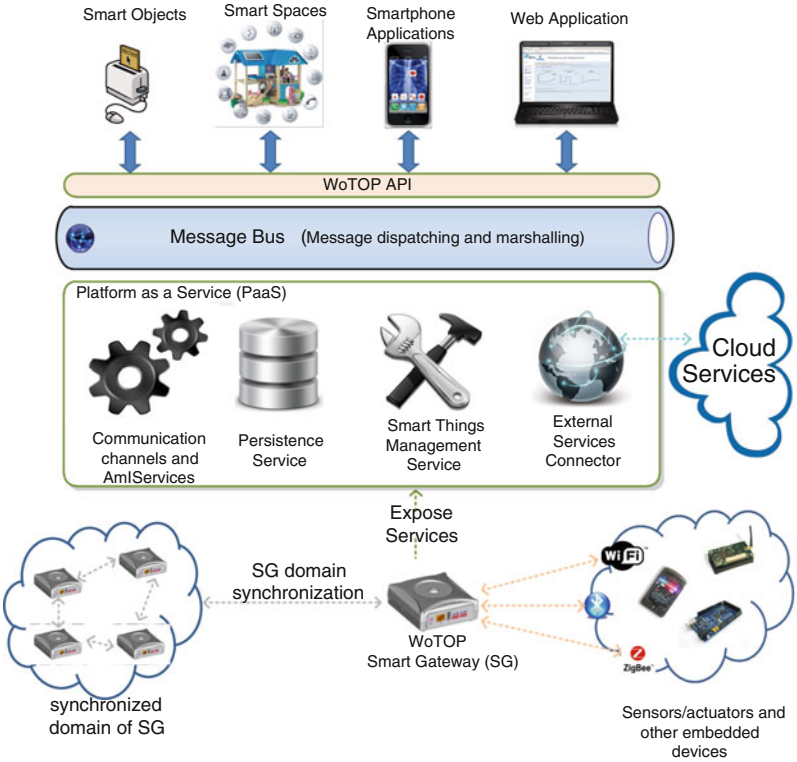


Fig. 5 Infrastructure, entities and services involved in WoTOP

client applications with capabilities to access WoTOP's resources. In order to explain those aspects, let us take as a reference the model shown in Fig. 5.

(i) *Integration of additional sensor and actuator devices*

There are two general methods to integrate additional sensors and actuators. The first one allows connecting those devices directly to some Smart Gateways hosting an instantiation of WoTOP. Firstly, this requires that the Smart Gateway is compatible with the devices to be connected at hardware level and that the specification of its communication protocol is open for the developer community. Let us assume that such preconditions are fulfilled thus, those devices can be physically connected to the Smart Gateways. Secondly, it will be necessary to deploy a specific adapter for those devices into the WoTOP. As commented in Sect. 3, every adapter has to be wrapped by OSGi bundles which are stored in an OBR. These bundles have to implement a specific protocol in order to enable bidirectional communication among WoTOP and devices. Additionally, every adapter must use the interface of the *Internet of Things Ecosystem Monitor*, which is the subsystem in charge of registering every device in WoTOP, as well as collecting and handling data received or sent from/to

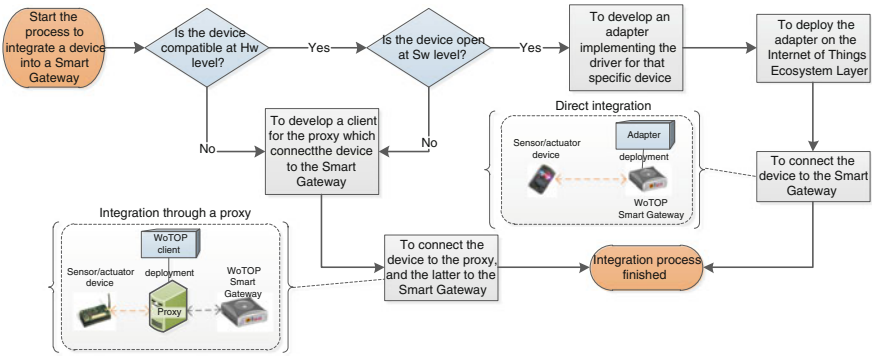


Fig. 6 The stages of the process for integrating devices to WoTOP

these devices. Whenever a device of that type is plugged into a smart gateway, the corresponding bundle will be automatically loaded to deal with it.

Only if the device to be integrated is not compatible at hardware level or its communication protocol is proprietary, its integration into WoTOP will depend on the framework provided by the manufacturer (composed of hardware and/or software elements). That framework will be used to create a kind of *proxy* among the device and WoTOP; this proxy will have to implement a client using the WoTOP API to get access to every resource deployed on the *Resource Composition and Orchestration Layer*. Specifically, that client would have to use the REST methods provided by the *Smart Thing Manager* resource in order to register or unregister the device to the WoTOP by means of the *ASTR* managed by the *Internet of Things Ecosystem Monitor*.

Figure 6 depicts a flow diagram showing the steps to be followed to integrate devices into WoTOP according to the two alternative methods explained before.

(ii) *Development and deployment of new resource components*

As commented previously in this Section, the *Resource Composition and Orchestration Layer* is the cornerstone to expose RESTful services into the Web. This layer not only manages several components by default to provide essential platform functionalities, but it can also deploy additional resource components that aggregate new platform services with the aim of improving and extending the usability of WoTOP for a wide range of scenarios. These components are called *Ambient Intelligence Resources*. Every developer is responsible of developing and deploying their own components inside the *Resource Composition and Orchestration Layer*; this layer is based on the RESTlet framework thus resources components must accomplish some requirements related to the internal work of that framework.[1] Those components can

[1] It implies that every resource component extends, at least, the super class *ResourceServer* to specify the accepted REST methods as well as the format of the message body. Then, it will be necessary to attach every component to a URI in order to forward every request to the right component. The latter is carried out during initialization sequence.

access every service provided by other resource components or by the *Web of Thing Middleware Layer* in order to reuse functionalities already implemented.

The idea behind building resource components which can perform atomic processes or integrate between each other to build complex services, is to create communities of developers that can share components to customize their own smart gateways.

(iii) *Development of WoTOP's application clients*

First of all, it is important to highlight that the purpose of *application clients* is completely different to that for *clients* used for implementing proxies that connect devices to WoTOP. Specifically, application clients are intended to create applications using the services offered by the WoTOP. They get access to those services through a message bus which hides the complexity of service. Such message bus provides a uniform interface that allows interacting with a smart space, i.e. to consume information generated by that smart space or to send messages to it in order to cause a specific behaviour. The most common way of consuming information consist of accessing the two information channels provided by default: On-demand and Event-driven channels, both provided by resource components at the *Resource Composition and Orchestration Layer*.

4 The Resource-Oriented and Ontology-Driven Methodology

4.1 Introduction

As explained in the previous Section, the WoTOP provides functionalities to integrate the lower layer of the reference architecture (see Fig. 1) into the Web by delivering mechanisms that facilitate the development and deployment of heterogeneous ecosystems for smart spaces. This WoTOP's feature fulfills a relevant requirement of the WoT research domain specified in the reference architecture, i.e. the ubiquitous and seamless interaction among client applications and smart spaces providing *real-world* services. However, there are other open challenges. Those challenges are going to arise in a near future, when smart spaces composed of hundreds, even thousands, of Web-enabled smart objects, will become daily omnipresent entities providing real-world services at global scale. There will be a need for sound methodologies that improve the links in the development chain of IoT and WoT, mostly the mapping of physical things into RESTful services in order to optimize the cost of deploying smart spaces.

Recent trends [6, 28–30] have adopted the concept of the Resource-Oriented Architecture (ROA) proposed by Fielding at the beginnings of 2000s [31]. The purpose of those proposals is to facilitate rapid developments and deployments in the context of the IoT and the WoT taking advantage of simplicity and versatility of ROA. Despite this promising characteristics, existing platforms do not still fully decouple lower layers (focused on hardware and protocols) from the higher layers

(mostly dedicated to the information management and the provision of services to application clients).

So far, any developer who faces the development and deployment of RESTful services over any platform or framework, still needs to have strong knowledge on general and specific technology aspects as communication protocols, specific programming languages or platforms of wireless sensor and actuator networks, among others. For this reason, some research works are addressing domain-specific development frameworks based on patterns and models since it is a solution to reduce costs with deployment time in large deployments of smart spaces. A recent trend in this field bets on the Model Driven Engineering (MDE) principles [32–34]. MDE-based methodologies simplify the process of design and development by using models and design patterns. Additionally, those approaches can increase the communication among participants working on the system development via standardization of languages and terminology, e.g. by means of a domain ad-hoc language specified by a work group in order to manage the life-cycle of a software product. Furthermore, MDE-based approaches can be very well tailored to the IoT and WoT, as this paradigm:

(i) Abstracts every part of the system through high-level models independently of the underlying software and hardware technologies.
(ii) Decouples consumers and providers of context resources (mostly sensor, actuators and logic processes), enabling a feasible reuse of model artifacts and software components.
(iii) Provides a model-based development framework to facilitate rapid and agile prototyping of complex deployments even for non-expert developers and users.

4.2 Architecture Principles of the ROOD Methodology

According to the points commented before, the research contribution included in this Section is focused on a MDE-based methodology which specifies a common abstraction model, semantically expressive, which provides the tools to define all the important elements of a smart space. A complete review of this contribution can be found in a previous work [5]. The initial motivation of this research was to provide a versatile solution to facilitate the development of different smart spaces composed of heterogeneous sensors, actuators and logic processors interacting among them through a variety of mechanisms. To address such a challenge, we propose the Resource-Oriented and Ontology-Driven Development (ROOD) methodology, which takes advantage from traditional MDE-based tools and improves them by providing semantic expressiveness. The ROOD was designed to be platform-agnostic, i.e. it can be used on almost every software and hardware platform without concerning the underlying software or hardware technology.

ROOD methodology is based on a OMG's (Object Management Group) MDE architecture: the Model-Driven Architecture (MDA). The MDA is a layered

architecture composed of three interrelated kinds of models: (i) Computational Independent Model (CIM), (ii) Platform Independent Model (PIM) and (iii) Platform Specific Model (PSM). These models have to be machine-readable so that they are successively transformed into other models, code stubs, schemas, test harnesses, and deployment scripts for diverse platforms [35]. OMG provides standardized tools to perform the MDA development methodology, particularly the Unified Modelling Language (UML) [36]. Domain Specific Modelling Languages (DSML) can be designed by means of a profile mechanism, provided by UML 2, with enough expressiveness and precision for almost any technological domain. In order to design a canonical MDA approach, we specified a novel DSL to model different aspects of smart spaces. On the one hand, this DSL allows defining the behaviour and contextual activities of smart objects from a high level point of view. On the other hand, they allow using modelling tools to define functional aspects of business processes from a low level point of view, associated to the previously modelled behaviours.

4.3 The Smart Space Modeling Language

ROOD methodology addresses the development of smart spaces from two different perspectives: (a) *the context activities*, which specify the behavior of resources (sensor, actuator, and interfaces for human interactions) used within a smart space and the relationships among them, and (b) *the smart object*, which provides a deployment perspective of the system involving information and processing models characterizing sensor and actuator entities within the smart space and its association with RESTful services. The ROOD methodology includes models related to both levels that encompass the mentioned features: (a) the *Environment Context Model* (ECM), and (b) the *Smart Object Model* (SOM). These models are instances of a DSL, the Smart Space Modeling Language (SsML) that was designed as an UML profile. Additionally, the modeling processes concerning those models are enriched through semantic technologies; concepts represented both in ECM and SOM are aligned to semantic contents that are stored in Knowledge Bases (KB) and defined according to an ontology called Smart Space Ontology (SSO). In that way, the ROOD methodology takes advantage from ontological resources to verify the completeness and consistence of ECM and SOM models according to the semantic description of the domain system; consequently the verification mechanism optimizes the model-to-model transformation processes from ECM to SOM.

The definition of SsML depends on the MDA architecture that is stratified in four abstraction levels (M0 through M3). The objectives of these levels are the following: (i) M0 contains instances of data for a specific platform; (ii) M1 is where the systems models are defined; (iii) M2 specifies the DSLs that take part in the definition of models at M1; (iv) Finally, M3 defines the Meta-Object Facility (MOF), that establishes the basis for different modeling languages.

Figure 7 shows the logical position of the SsML in the MDA architecture. As it can be seen, the SsML is in M2 layer and extends the UML metamodel; SsML uses

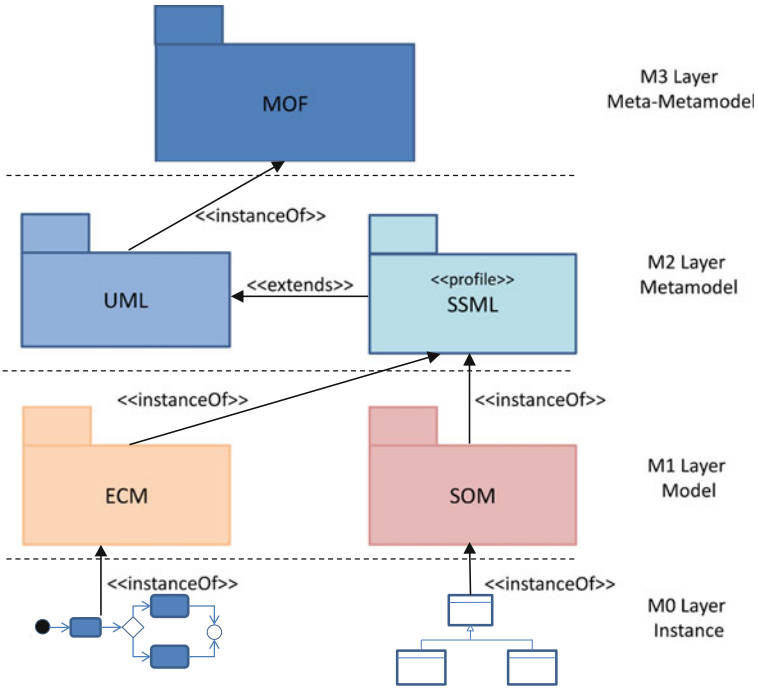


Fig. 7 The SsML according to the MDA architecture

the extension mechanisms defined in the UML 2 specification in order to create an own profile that defines every necessary element (entities, relations and interfaces) to model the smart space.

4.4 Stages of the ROOD Methodology

This Section presents the different stages of the ROOD methodology involving the elements of the architecture. This guideline focuses on the traceability between the concepts presented in ECM and SOM metamodels, as well as the mechanisms to verify models delivered in each stage.

As discussed in the previous section, any development methodology based on MDA consists of three main phases: (i) Computation Independent Model (CIM); (ii) Platform Independent Model (PIM); (iii) Platform Specific Model (PSM). Along these phases, MDA manages to separate the conceptual design (focusing on functional requirements of the system) from the platform features (defining no functional and technological aspects of the underlying architecture).

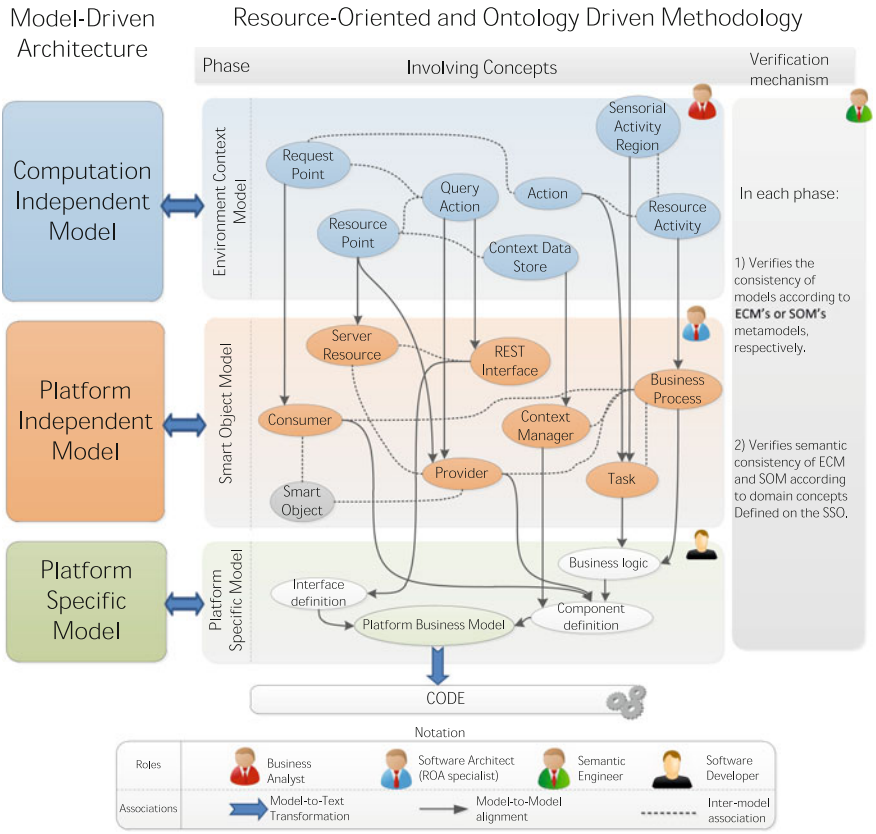


Fig. 8 The ROOD methodology according to the concepts managed in every phase, as well as model transformations and verifications between phases

Firstly, we identified a set of traceability relations between concepts in the different models. A full map of the mentioned traceability can be consulted in [5]. In Fig. 6, it is represented the major concepts that have to be modelled in each RODOs stage as well as the traceability between ECM and SOM elements. Additionally, the processes for model verification are included.

The concepts shown in Fig. 8 are defined in the ECM and SOM models. Such models set restrictions of use for different entities in each stage of the ROOD methodology, as well as their relationships. The entities specified in SSO are projected on concepts of ECM and SOM models which facilitates the verification of integrity and completeness of models in relation to the domain information stored in Knowledge Bases.

It is important to remark that the ROOD methodology is flexible and its stages are loosely coupled. By means of this feature, a wide range of professionals are enabled to work together, collaborating and working jointly in the development

chain, e.g.: (i) Business analyst; (ii) Software architect, specialized in ROA; (iii) Semantic engineer; and (iv) Software developer.

Before starting the first stage of the methodology, each element of the smart space, involving the specification and development of smart objects, will have to be analyzed. Meanwhile, every smart object in such smart space is supported by one or more platforms with a set of sensor and actuator devices. The behavior of the smart objects will be conditioned to the business processes and underlying tasks (e.g. those managing sensor measurements) that are performed by the platforms associated to them.

In ROOD, there are two different agents managing the business logic: consumers or providers. Finally, services exposed by the smart objects, to be consumed by other entities belonging to the smart space, are encapsulated into resources, which are defined accordingly to the REST architectural style. The ROOD methodology follows the most accepted REST implementation in which, for each resource, it is assigned a Unique Resource Identifier (URI), one or more HTTP method (GET, PUT, POST or DELETE) and specific formats for input and output messages (usually XML, RDF or JSON).

Once the Knowledge Base of the smart space is instantiated, the modeling phase of the ROOD methodology can start. Firstly, business analysts have to model the smart space using the elements defined in the ECMs metamodel. In this stage, only behavioral information of the smart space is represented without concerning the underlying platform, e.g. activity threads, actions, transitions between actions, and smart objects involved in that activity context.

The information contained in ECM models is used to partially generate the models in the next stage, SOM. This stage should be managed by software architects specialized in resource-oriented architectures, who will take advantage of the information from the ECM to model agents (providers or consumers of services), business processes and tasks. Moreover, it will be needed to set up the information system to manage context information that will have an influence in the current and future behavior of the smart objects. Finally, the involved software architects will design the necessary architectural elements that will offer the smart object resources as RESTful services. This step will provide the key piece to integrate the smart space into a WoT paradigm.

The models created in previous steps will be subjected to a verification process to check the consistency and integrity of the entities, relationships and other semantic information represented in them, according to the domain information stored in a scenario Knowledge Base.

The final step consists of generating program code from SOM models. For this aim to be achieved, the elements represented in SOM models are filtered through a model-to-text transformation mechanism. The percentage of generated code can vary depending on the underlaying platform but in any case, it will be totally generated. Therefore, a software developer is required to complete existing gaps in the code (e.g. configuration parameters for hardware peripherals or specific information to integrate devices into a communication infrastructure).

It is important to highlight that the transformation specified in the ROOD methodology (ECM-to-SOM and SOM-to-PSM/code) is conducted by mapping rules that in some cases are almost automatically generated and only partially automated in the remainder cases.

5 Case-Study Through Real Prototyping: Smart Shop

Smart spaces are designed to adapt to their inhabitants, at the same time that they can configure their operation in an efficient way. Combining WoTOP and the ROOD methodology, it is feasible to easily deploy and configure a smart space and its related services. In this Section, we describe how to do it to in a particular scenario.

5.1 Motivation Scenario

Let us consider a public space such a smart shop, in which visitors can wander about and explore its commercial offer. In this smart shop:

- Clients are able to augment the objects on sale, retrieve personalized offers or browse customized information about external services through their smart devices.
- Technical assistants are able to control the environmental conditions of the space. They can receive alerts and notifications in case that humidity, temperature or occupancy thresholds are exceeded. They may also establish the shop configuration that optimizes energy consumption while maximizing clients comfort.
- Shop managers may configure specific atmospheres, e.g. through virtual contents and lighting.

To enable these services, it is necessary to configure an infrastructure of sensors and actuators, which connected to WoTOP will be able to provide the needed information to the consumer services. In particular, to deploy the services above, the following infrastructure is required:

- An accurate positioning system, providing centimeter error.
- A network of humidity, temperature and light wireless sensors.
- A network of wireless actuators (e.g. capable of managing the air conditioning, the lights and the blinds).

And consumer services include:

- An augmented reality service with a backend, which allows managers to configure the virtual offering of contents. These contents will be delivered taking into account the users preferences and context.
- An environmental monitor with a configuration board for alerts.

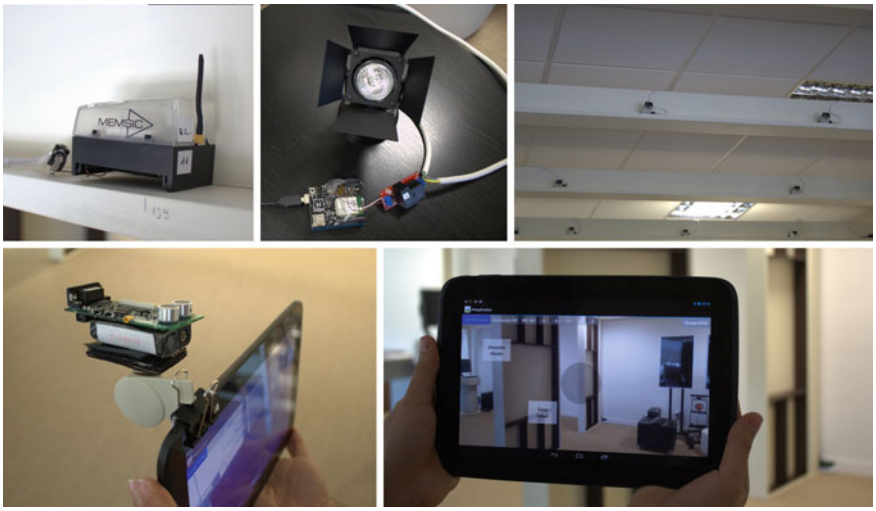


Fig. 9 Infrastructure deployed to simulate the Smart Shop. From *top to bottom* and *right to left*: **a** MicaZ node with a sensor board integrated, fixed to the ceiling; **b** Arduino Uno connected to a LED lamp; **c** Cricket-based network deployed strategically on a ceiling structure; **d** Cricket node attached to a tablet PC; **e** A tablet PC running a AR application

- An environmental optimizer, to control the actuators while maximizing comfort and minimizing energy consumption.

We have prototyped this service scenario in our Experience Lab, a testing environment located in the Montegancedo Campus of the Universidad Politecnica de Madrid. In this space, it is feasible to customize technical infrastructure, furniture and décor to simulate different environments. A sample of the variety of elements deployed for our study case is shown in Fig. 9. In the next subsections, we detail how the infrastructure sensors and actuators have been deployed and connected to WoTOP.

5.2 Deployment Infrastructure

The necessary deployment infrastructure for the smart shop is shown in the Fig. 10. The core of the infrastructure is a Smart Gateway running a WoTOP's instance. The role of this Smart Gateway is twofold. On the one hand, it is responsible of gathering data from smart things and dispatch them to the application clients according to their interests in consuming them. On the other hand, the Smart Gateway can forward messages from client applications to actuators whenever they consider.

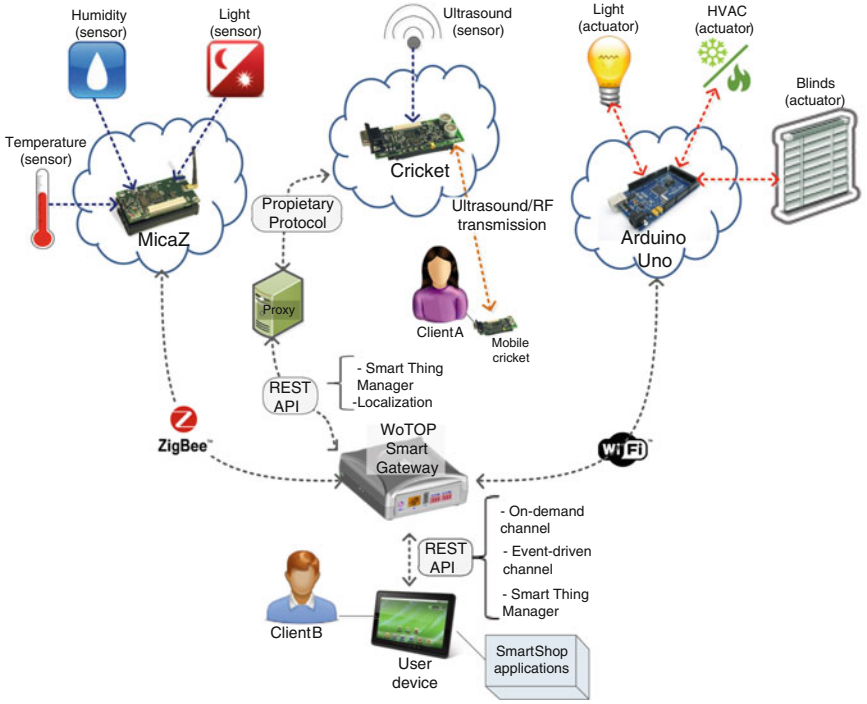


Fig. 10 The deployment infrastructure necessary for the proposed smart shop and the users involved on it

Basically, the sensing and actuating infrastructure is based on wireless embedded nodes of different technologies: Crossbow MicaZ,[2] Crossbow Cricket[3] and Arduino Uno.[4] Each one has different roles within the scenario that are collected in the Table 2.

Most of this infrastructure is deployed on specific facilities of the smart shop, i.e. they are fixed nodes associated to those places. Exceptionally, there will be one or more mobile nodes based on the Cricket platform. The mobile node is used to transmit two concurrent message based on the combination of RF and ultrasound technologies. These messages are listened by several beacon nodes to calculate the localization information attached to the mobile node.[5] Note that the Cricket network is connected to the Smart Gateway through a proxy according to the method explained in Sect. 3. The rest of embedded nodes are directly integrated into the Smart Gateway.

Beyond the networks of embedded sensor and actuator nodes, the infrastructure is also composed of user devices that are provided to the clients of the smart shop. These devices have a significant role within the smart shop since they make possible

[2] http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf

[3] http://www.willow.co.uk/Cricket_Datasheet.pdf

[4] http://arduino.cc/en/Main/arduinoBoardUno

[5] For more information about the localization algorithm, visit: http://cricket.csail.mit.edu/.

Table 2 Classification of the deployed smart things according to their sensors and actuators, data inputs and outputs, as well as services provided by them

Smart thing	Sensor/ Actuator	Data input/ Output	Main functionality and associated services
Crossbow MicaZ	**Sensors**: • Temperature • Humidity • Light	**Output**: temparature, humidity and light measurements both in event-driven and on demand modes	**Funct.**: to dispatch temperature humidity and light measurements **Services**: saving energy and ambiance configuration
Crossbow Cricket	**Sensors**: • Ultrasound	**Input**: RF signal + ultrasound signal **Output**: difference between reception timestamp of RF and ultrasound signals in the listeners	**Funct.**: to calculate accurately the localization attached to mobile users within specific areas **Services**: AR, saving energy and recommendation of products and services
Arduino Uno	**Actuators**: • Blind • Led lamps	**Input**: Command message: turn on/off for lamps; up/down for blinds	**Funct**: To activate actuators according to commands received from context-aware agents running on client applications **Services**: AR, saving energy and ambiance configuration

the interaction between human users and real world services which are provided through the Smart Gateway. For a reasonable performance of client applications and other agents running on those devices, it is necessary an appropriate configuration and usage of the services provided by WoTOP. This requires to identify the producers and consumers of information, as well as the *method* used to dispatch information between producers and consumers. The next section includes a brief overview of the configuration of the WoTOP's services for our smart shop, including the most important parameters.

5.3 Configuration of WoTOP's Services for the Smart Shop

As mentioned above, the first stage for reaching a suitable configuration of the WoTOP's services consist of identifying producers and consumers of information. The information producers of the smart shop were gathered in Table 2. In general they correspond with embedded sensor nodes, but also the Localization resource is identified as an information producer. The latter can provide preprocessed information from specific coordinates generated by Cricket-based network, e.g. areas of the building in which users are located at some point.

The information consumers are mostly client application running on user devices, for instance tablets and smartphones, that will process such information to offer products or services to the shop users. In other occasions, consumers will enable autonomous responses from the actuators in the smart space. Previously, a method to dispatch information between producers and consumers of information has to be set up. As said in Sect. 3, WoTOP provides two different modes to transmit information to consumers: event-driven and on-demand.

Let us illustrate the configuration of the smart shop through the saving energy service. This service is associated to the *sustainable consumption agent* which consumes concrete events to minimize energy consumption. Specifically, these agents will perform *conditional-based* subscriptions for events that are generated from temperature, humidity and light sensor nodes to manage situations in a untended and smart way according to the received sensor values, e.g. to configure automatically the HVAC system, pull up/down blinds and set up light intensity of the LED lamps of a room/area of the shop, depending on the energy efficiency rating needed for the building.

Additionally, the *sustainable consumption agent* can make *contract-based* subscriptions in order to collect periodical localization information from clients, in order to improve even more the energetic efficiency (e.g. to perform fine-grained configuration of the HVAC and LEDs lamps depending on the accurate localization of the clients and staff in an area of the shop).

The *ambience configuration agent*, that is associated to the smart ambiance configuration service, also needs to make *contract-based* subscriptions for localization of users to offer interaction functionalities to control some parameters of the environment, e.g. to control actuators on-demand mode by pointing the camera of the user device (e.g. tablet or smart phone) to the specific actuator.

Table 3 shows a list of producers and consumers of information associated to the services deployed in the smart shop. Every producer and consumer is related to a URI that, in the case of consumers, identify the agent which has to receive a callback message, i.e. a POST message with an event in the body. The format of the messages encapsulated into the event payload (JSON documents) is also shown in Table 3.

From the information collected in Table 3, subscriptions of different types have to be done. The number and type of those subscriptions will depend on the number of consumers being interested in consuming specific types of events. Table 4 shows examples of two type of subscriptions that could be sent to the Smart Gateway from different agents.

It is important to highlight that interactions between agents running on user devices and actuators deployed on the smart shop (e.g. between *ambiance configuration agent* and HVAC, blinds or LEDs lamps) are carried out by means of typical request/response methods, i.e. using on-demand communication mode of WoTOP. Those requests are sent to the Smart Gateway by means of a PUT method to the specific resource and, from there, they are forwarded to a specific actuator. The same communication mode is used to obtain sensor values instantly but, in this case, it is sent a GET request to a resource associated to a sensor.

Table 3 Event configuration for services provided by client applications in the smart shop

Smart Shop Service	Information management		Event type*	Payload format
	Producer	Consumer		
• Saving Energy • Smart Ambience Configuration	**Temperature sensor** URI: `http://[gw-id]/→ →/…/sensor/temp`	• **Sustainable consumption agent** URI: `http://[IP]:[port]/→ →/agent/energysav` • **Ambience config. agent** URI: `http://[IP]:[port]/→ →/agent/ambience`	1	`{ "key": "temp", "unit":"celsius", "value": (String) }`
	Humidity sensor URI: `http://[gw-id]/→ →/…/sensor/hum`		1	`{ "key": "hum", "unit":"RH", "value": (String) }`
	Light sensor URI: `http://[gw-id]/→ →/…/sensor/ligth`		1	`{ "key": "temp", "unit":"lux", "value": (String) }`
	Localization (coordinate) URI: `http://[gw-id]/→ →/…/loc/coordinate`		2	`[ {"key": "x", "unit":"double", "value": (String)}, … , {"key": "z", "unit":"double", "value": (String)} ]`
Augmented Reality: • Recommendation of products and services • Smart Ambience Configuration	**Localization (coordinate)** URI: `http://[gw-id]/→ →/…/loc/coordinate`	• **Location-aware recommendator agent** URI: `http://[IP]:[port]/→ →/agent/recommedator`	2	`[ {"key": "x", "unit":"double", "value": (String)}, … , {"key": "z", "unit":"double", "value": (String)} ]`
	Localization (area) URI: `http://[gw-id]/→ →/…/loc/area`	• **Location-aware recommendator agent** URI: `http://[IP]:[port]/→ →/agent/ambience`	1	`{ "key": "area", "unit":"integer", "value": (String) }`

*) Type of event according to the two supported communication modes : *contract* (1) and *conditional* (2).

Table 4 Subscription examples in JSON format: on the left, conditional-based subscription from a saving energy agent to temperature values in a MicaZ node; on the right, a contract-based subscription from an ambience agent to localization information of a mobile Cricket node

```
{
    "consumer": {"URI": "http://[IP]:[port]/",
                 "Id": "/agent/energsav"},
    "producer": {"URI": "http://[gw-id]",
                 "Id": "/micaz/1/sensor/temp"},
    "type": "1", //Contitional (1)
    "end": "86400000", //24 hours
    "event": [{ "key": "temperature",
                "unit": "Celsius",
               "value": "25",
                "operator": "GREATER"}, ...]
}
```

```
{
    "consumer": {"URI": "http://[IP]:[port]/",
                 "Id": "/agent/ambience"},
    "producer": {"URI": "http://[gw-id]",
                 "Id": "/cricket/1/loc/coordinate"},
    "type":"2", //Contract (2)
    "start":"20000",//Start time (ms) of the subscription
    "end":"300000", //End time (ms) of the subscription
    "samplingPeriod":"1500", //Sampling period (ms)
    "event": [{ "key": "x",
                "unit": "centimeter"},
              { "key": "y",
                "unit": "centimeter"}
               ]
}
```

To conclude, although it was not mentioned in this Section, the configuration of WoTOP's services, focused on managing the information exchange between the entities of the smart shop, can be carried out by means of the ROOD methodology. Currently, we are working on a completely functional prototype of a modeling tool based on ROOD that will be able to be used to model every entity involved on a smart space taking into account their roles (consumer and/or producers), as well

as the major characteristics of their interactions with other entities belonging to the same smart space (communication modes, message format, subscription parameters, etc.) among other characteristics mentioned in Sect. 4.

6 Conclusions

In this Chapter, some important aspects of the Web of Thing were analysed, together with our approaches to deal with them. In particular, we firstly presented the Web of Things Open Platform (WoTOP) that was designed keeping in mind an essential principle: to bring the gap among the physical world and human users by developing smart spaces which interconnect seamlessly smart things, business process and users. WoTOP takes advantage of well-known Web technologies as well as of one of its most popular approaches in the pervasive computing field, the WoT paradigm, in order to hide the underlying technological heterogeneity. Such heterogeneity comes from a IoT-based ecosystem, composed of a number of different sensor and actuator devices which can be connected to WoTOP's smart gateways by means of two different methods: (i) the first one enables to connect devices directly to those Smart Gateway and, (ii) the second one, to connect devices indirectly through a proxy, that is ad-hocly developed according to its technological features (e.g. protocols and hardware).

The second contribution, briefly introduced in this Chapter, was the Resource-Oriented and Ontology-Driven (ROOD) methodology which is based on the Model-Driven Architecture (MDA) of the OMG. This methodology aims at facilitating the rapid and friendly deployment of WoT-based smart spaces, built on heterogeneous ecosystems and involving different producers and consumers of resources. For this purpose, visual modelling languages have been specified by extending the UML 2 profile so that no-technicians could develop and deploy smart things belonging to a smart space. Moreover, every stage of ROOD methodology is enhanced with validation mechanisms that autonomously analyses the models to validate them both syntactically and semantically.

Finally, the feasibility of the major research contributions presented in this Chapter were demonstrated through a case study. The motivation scenario used for our case study was a smart shop. That smart space was partially deployed in a simulated environment, using a variety of technologies based on wireless networks of embedded sensor and actuator devices, and some multimedia devices running specific applications to interact with the smart shop. The core of the smart space is a Smart Gateway hosting an instance of WoTOP through which every entity (physical or logical) of the smart space can be connected among each other rapidly and seamlessly, accordingly to the requirements of the smart space.

6.1 Outlook to Future Research Challenges

During 2000's decade technical and scientific disciplines related to Pervasive Computing progressed exponentially, opening novel and exciting research challenges which have provided new basis for integration of computers in the everyday life. Particularly, the Internet of Things and the Web of Things paradigms provide a wide amount of service possibilities.

The major idea behind IoT is to achieve a network layer for constrained devices highly compatible with Internet environment through IP-based protocols. In this research field, the most important initiative is 6LoWPAN which is an Internet Engineering Task Group (IETF) Working Group founded with the purpose of designing of protocols and mechanisms to integrate seamlessly networks of wireless and resource-constrained devices into Internet keeping in mind the major features of both types of networks (e.g. routing, discovery, security or memory footprint). Although interesting results have been reached in form of RFCs, there is still a lot of work to be done before reaching an standard that is agreed by industry and academia.

Beyond network layer, interesting research challenges have risen in the edge between the IoT and WoT research domains. Those initiatives have been focused on an application perspective consisting of tackling a direct integration of resource-constrained devices into the Web [37]. Among proposals in this field, the Constrained Application Protocol (CoAP) is being widely accepted in the IoT community. CoAP is a RESTful protocol which minimizes the complexity of mapping with HTTP enabling resource-constrained devices to deploy tiny Web servers that are characterized by low footprints and workload. The IETF Constrained RESTful environments (CoRE) Working Group is carrying out the major standardization work for CoAP. Despite CoAP is in final phases of standardization, current Web browsers lacks capabilities to use CoAP-based devices seamlessly without requiring cross-proxies. To the best of our knowledge, only one proposal [38] has addressed the issue of integrating CoAP connectivity capabilities into browsers to facilitate the communication between human users with resource-constraint devices like if they were conventional Web servers.

Additionally, the management of bi-directional data streams is still an issue under discussion. HTTP is a stateless protocol which works adequately for *request-response* operations initiated by clients (e.g. read/write data from/to a embedded sensor or actuator) but lacks of mechanisms to manage data streams in event-driven scenarios based on publisher/subscriber paradigm. In this kind of scenarios, information is asynchronously generated by embedded devices, thus it must be dispatched to specific clients (only the interested ones in that information) as soon as possible. In this Chapter, we have detailed an event-driven subsystem, which was implemented for WoTOP. This subsystem is based on Webhooks to send information from smart things to clients. We could prove that Webhooks works correctly under controlled local environments (e.g. LANs or VPNs). However, a proper working of Webhooks is not guaranteed when clients are located at external networks since the call-back POST messages, that encapsulates events, would have to be transmitted through firewalls that could

discard that type of traffic. Alternatives to Webhooks have been proposed as the use of Extensible Messaging and Presence Protocol (XMPP) or Comet-based mechanisms. The former is an open protocol based on real-time messages in XML format, widely used in instant messaging applications (e.g. Google Hangouts). The latter is a mechanism that enables Web servers to push data back to clients based on AJAX programming. Despite both proposals would accomplish requirements for dispatching information to the clients asynchronously, they are not suited for embedded devices since they use heavyweight and verbose formats as XML. Moreover, they usually are optimized for Web browsers and plugins to be run on them, restricting their capabilities to implement customized application clients.

Other alternatives propose extending Atom and RSS protocols to create publication/subscription brokers which can manage feeds of information. Clients can subscribe to that feeds through a message broker in order to receive data via callbacks when they are published. For example, the PubSubHubbub[6] initiative provides the mentioned mechanisms using the PuSH protocol. Nonetheless, this type of solutions could be unsuitable for scenarios in which sensitive information is managed (e.g. logistic, eHealth or finances) since every event have to be dispatched by means of a message broker depending on a third part which could not provide necessary Quality of Service level (in terms of timeliness or reliability), or sufficient security parameters.

Other research topics that are being considered are those related to the integration of smart things into the Semantic Web. Recently, some promising approaches have arisen with the purpose of semantically annotating smart things for different applications, for instance to collaborate among devices to reach a common goal by means of reasoners [39]. It will be interesting to observe how these approaches will evolve to support complex and large smart spaces in the future, when semantics will be necessary to search information and create mashups of services related to the physical world.

To conclude, it is certainly important to mention that one of the pillars of the Future Web of Things will lie on the existence of methodologies to prototype, deploy and maintain large smart spaces, rapidly and easily, in order to reduce costs as much as possible. As was proposed in ROOD methodology, a major characteristic of this type of methodologies consist of providing friendly tools (essentially based on visual elements) to facilitate no-technicians to deploy smart space abstracting from low level issues. The Model-Driven Architecture (MDA) is a step forward in this direction.

Acknowledgments This work has being supported by the Government of Madrid under grant S2009/TIC-1485 (CONTEXTS). The authors also acknowledge related discussions within the THOFU initiative, funded by the Spanish Center for the Development of Technology.

[6] https://code.google.com/p/pubsubhubbub/

References

1. Weiser, M.: The computer for the 21st century. IEEE Pervasive Comput. **99**(1), 19–25 (2002)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**, 2805 (2010)
3. Besbes, M.A., Hamam, H.: An intelligent RFID checkout for stores. In: 2011 International Conference on Microelectronics (ICM), pp. 1–12 (2011)
4. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. In: Proceedings of the 2000 International Conference on Software Engineering, pp. 407–416 (2000)
5. Corredor, I., Bernardos, A., Iglesias, J., Casar, J.R.: Model-driven methodology for rapid deployment of smart spaces based on resource-oriented architectures. Sensors **12**(7), 9286–9335 (2012). URL http://www.mdpi.com/1424-8220/12/7/9286
6. Corredor, I., Martínez, J., Familiar, M.: Bringing pervasive embedded networks to the service cloud: a lightweight middleware approach. J. Syst. Architect. **57**(10), 916–933 (2011)
7. Corredor, I., Martínez, J., Familiar, M., López, L.: Knowledge-aware and service-oriented middleware for deploying pervasive services. J. Netw. Comput. Appl. **35**(2), 562–576 (2012)
8. Brown, P., Estefan, J., Laskey, K., McCabe, F., Thomton, D.: Reference Architecture Foundation for Service Oriented Architecture Version 1.0. Technical Report, OASIS (2012). URL http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf
9. Cosm–Internet of Things Platform Connecting Devices and Applications for Real–Time Control and Data Storage (2013). https://cosm.com/
10. Paraimpu–The Web of Things is more than Things in the Web. http://paraimpu.crs4.it/
11. Arduino Website (2013). arduino.cc/.
12. Sensinode Ltd. (2013). https://www.sensinode.com/
13. SmartThings–Make your world smarter (2013). http://smartthings.com/
14. Gomez-Goiri, A., de Ipina, D.L.: Assessing data dissemination strategies within triple spaces on the web of things. In: 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 763–769 (2012)
15. Farrell, S.: API keys to the kingdom. IEEE Internet Comput **13**(5), 91–93 (2009)
16. Guinard, D., Fischer, M., Trifa, V.: Sharing using social networks in a composable Web of Things. In: 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 702–707 (2010). ID: 1
17. Blackstock, M., Lea, R.: IoT mashups with the WoTKit. In: 2012 3rd International Conference on the Internet of Things (IoT), pp. 159–166 (2012)
18. Mayer, S., Guinard, D., Trifa, V.: Facilitating the integration and interaction for real-world services for the web of things. In: Urban Internet of Things: Towards Programmable Real-Time Cities (UrbanIOT 2010); Workshop at the Internet of Things 2010 Conference (IoT 2010) (2010)
19. Wu, Z., Itala, T., Tang, T., Zhang, C., Ji, Y., Hamalainen, M., Liu, Y.: Gateway as a service: a cloud computing framework for web of things. In: 2012 19th International Conference on Telecommunications (ICT), pp. 1–6 (2012)
20. ThingWorx–M2M and Internet of Things Application Development Platform (2013). https://www.thingworx.com/
21. AirVantage–M2M Platform (2013). http://www.sierrawireless.com/airvantage
22. Axeda–M2M cloud service (2013). http://www.axeda.com/
23. Thiesse, F., Floerkemeier, C., Harrison, M., Michahelles, F., Roduner, C.: Technology, standards, and real-world deployments of the EPC network. IEEE Internet Comput. **13**(2), 36–43 (2009)
24. IERC–European Research Cluster on the Internet of Things (2013). http://www.internet-of-things-research.eu/
25. Corredor, I., Martínez, J.F., Familiar, M.S.: Research experiences about internetworking mechanisms to integrate embedded wireless networks into traditional networks. In: Interconnecting Smart Object with the Internet Workshop (European Commission -IETF) (2011)

26. Evrything–Make products smart (2013). http://www.evrythng.com/
27. ThingSpeak (2013). https://www.thingspeak.com/
28. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the Web of Things. In: 2010 Internet of Things (IOT), pp. 1–8 (2010)
29. Villalonga, C., Bauer, M., López, F., Huang, V., Strohbach, M.: A Resource Model for the Real World Internet, Smart Sensing and Context, vol. 6446, pp. 163–176. Springer, Heidelberg (2010)
30. Zhang, W., Jiang, L., Cai, H.: An ontology-based resource–oriented information supported framework towards RESTful service generation and invocation. In: 2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), pp. 107–112 (2010)
31. Fielding, R.: Architectural styles and the design of network-based software architectures. Ph.D. Thesis, University of California, Irvine (2000)
32. Katasonov, A., Palviainen, M.: Towards ontology-driven development of applications for smart environments. In: 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 696–701 (2010)
33. Soylu, A., Causmaecker, P.D.: Merging model driven and ontology driven system development approaches pervasive computing perspective. In: 2009 24th International Symposium on Computer and Information Sciences, ISCIS 2009, pp. 730–735 (2009)
34. Tetlow, P., Pan, J., Oberle, D., Wallace, E., Uschold, M., Kendall, E.: Ontology driven architectures and potential uses of the semantic web in systems and software engineering (2006). URL http://www.w3.org/2001/sw/BestPractices/SE/ODA/
35. Management Group (OMG), O.: UML Infraestructure Specification (2011)
36. Gherbi, T., Meslati, D., Borne, I.: MDE between promises and challenges. In: 11th International Conference on Computer Modelling and Simulation, UKSIM '09, pp. 152–155 (2009)
37. Kovatsch, M., Mayer, S., Ostermaier, B.: Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things. In: Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012, pp. 751–756 (2012)
38. Kovatsch, M.: CoAP for the web of things: from tiny resource-constrained devices to the web browser. In: The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '13, p. 1495 (2013)
39. Mayer, S., Basler, G.: Semantic metadata to support device interaction in smart environments. In: The 2013 ACM International Joint Conference on Pervasive and Ubiquitous, Computing, p. 1505 (2013)

Embedded Web Technologies for the Internet of Things

Walter Colitti, Nguyen Thanh Long, Niccolò De Caro and Kris Steenhaut

Abstract The adoption of open standard communication protocols based on embedded IP and Web technologies is a key building block for the realization of the Internet of Things (IoT) vision, where trillions of smart objects will be connected and will open up exciting possibilities for the creation of new and innovative services. Embedded Web protocols are interoperable, reduce the overall IoT architectural complexity, and deliver a long term value as it has already been demonstrated by the success of the Internet and of the Web. This chapter discusses on the adoption of IP and in particular of Web technologies in smart objects for the realization of the Web of Things (WoT). The work reviews the main protocols and discusses on how and why they improve interoperability, reduce the overall IoT architectural complexity and facilitate the integration between the IoT and Web applications.

1 Introduction

Recent market studies in telecommunications have estimated that in the near future billions or trillions of smart objects will connect physical environments to the Internet. A smart object can be defined as any physical entity enhanced with sensing/actuating, computation and communication capabilities. The concept of having physical entities connected to the Internet is commonly known as the Internet of Things (IoT). IoT will unleash exciting possibilities and challenges for a variety of application domains, such as smart metering, e-health, logistics, building and home automation [1, 2].

The choice of the communication protocols has been a major debate among the IoT community. The IoT has been largely dominated by proprietary and domain specific protocol stacks. This has been motivated by the need of performance optimization of severely constrained devices and by the vendor lock-in based market, which forces

W. Colitti (✉) · N. T. Long · N. De Caro · K. Steenhaut
Department ETRO-INDI, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
e-mail: wcolitti@etro.vub.ac.be

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_3,

customers to be locked into a long term relationship. The use of highly specialized proprietary protocols implies the need of sophisticated application gateways between the network of smart objects and the Internet. Application gateways complicate the overall architectural complexity, hamper the scalability and complicate the application development. In addition, proprietary and specialized siloed protocols are also an obstacle for the creation of new innovative services [3].

The fragmentation of communication technologies has been considered as one of the main obstacles hampering the realization of the IoT vision. The ubiquity of sensors and actuators will unlock huge opportunities for the Internet of Services (IoS). But a necessary requirement is that the IoT will be based on open standard communication technologies which enable interoperability and plug-and-play capabilities. Open standard communication protocols based on Internet and Web technologies are a way to deliver long term value [4].

The introduction of open standards based on IP and Web technology into smart objects has been considered as a way to overcome the IoT protocol fragmentation problem and consequently to improve interoperability, flexibility, scalability and facilitate the integration of the IoT with the Internet and the Web without complicating the architectural complexity. This is a common vision shared by several communities, such as academia, industrial players and standardization bodies. The IP for Smart Objects (IPSO) Alliance, a cluster of major IT/telecom players and wireless silicon vendors, promote the use of embedded IP into constrained devices. The Internet Engineering Task Force (IETF) has standardized IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), which enables the transmission of IPv6 packets in networks of resource constrained devices and IPv6 Routing Protocol for Low-power and Lossy networks (RPL), which is an IPv6 based routing protocol for the IoT [5–7].

A low power embedded IPv6 networking layer opens up the possibility to embed Web services into smart objects based on the Representational State Transfer (REST) and on the widely known Hyper Text Transfer Protocol (HTTP) up to the sensor/actuator level. This is commonly known as the Web of Things (WoT) paradigm [8]. The use of HTTP on top of 6LowPAN based smart objects eliminates the need of application gateways which can be replaced by less complex 6LoWPAN enabled edge routers. This scenario preserves the end-to-end design principle when connecting the WoT to the Internet/Web [9].

HTTP based Web applications are based on design principles different than IoT applications. In addition, it has been demonstrated that HTTP is too cumbersome for constrained devices. The main reason is that it is based on the Transmission Control Protocol (TCP) which has a large overhead. Motivated by this problem, the IETF Constrained RESTful environments (CoRE) working group, has defined a web transfer protocol called Constrained Application Protocol (CoAP). CoAP is a REST based web transfer protocol which includes several HTTP functionalities re-designed for constrained devices. CoAP is based on the User Datagram Protocol (UDP) instead of TCP [6, 10, 11].

The aim of this chapter is to give an overview on the introduction of standard IP/Web technologies in networks of smart objects. We discuss on the benefits of

embedded IP and on how this technology improves IoT interoperability and architectural complexity. We describe embedded Web technologies, giving special attention to CoAP which is the protocol we have adopted in our IoT system development, and we review the main state-of-the-art work on Web standards. We also discuss on the main architectural issues for the integration of networks of web based smart objects with web applications. We do so by describing a concrete example of a REST/CoAP based Wireless Sensor Network (WSN) we have developed for greenhouse monitoring.

The rest of the chapter is organized as follows. Sections 2 and 3 discuss on embedded IP and on the embedded Web, respectively. Section 4 focuses on CoAP and on the main differences with HTTP. Section 5 describes the main building blocks which enable the integration of a REST/CoAP based WSN with a web application. Section 6 discusses on the main advantages of the adoption of open standard Web based technologies. Section 7 concludes the chapter.

2 Embedded IP Networking Technologies

The realization of a WoT requires the use of embedded IP technologies. This section gives an overview of 6LowPAN and discusses on how it brings advantages in terms of interoperability, architectural complexity and scalability.

The use of IP technology in smart objects is not straightforward. IP was designed for relatively stable networks and for non-constrained nodes. Sensor/actuator motes are severely constrained in terms of power, memory, communication bandwidth and energy. As a consequence, networks of smart objects like WSNs can be considered an instance of Low Power and Lossy Networks (LLN) [12, 13], in which the communication protocols need low power consumption and in which the packet loss may be deteriorated by the physical environments (fading, interferences, bit errors, etc.). This problem has motivated IETF's standardization activity in defining 6LoWPAN so that to enable the use of IPv6 in networks of resource constrained devices, such as the ones based on IEEE 802.15.4 radio links.

6LoWPAN defines an adaptation layer which provides encapsulation and header compression mechanisms allowing IPv6 packets to be transferred between resource constrained devices. IP packets in general have a significantly large packet header resulting in a large overhead which increases the motes' energy consumption and the packet loss probability. The header compression in the 6LowPAN adaptation layer eliminates redundant fields of the packet header. The adaptation layer also provides fragmentation and reassembly mechanisms needed to transfer IPv6 packets in multiple 127 bytes long IEEE 802.15.4 frames [5].

A major advantage of the use of embedded IP based technologies is the interoperability, which is one of the main reasons for the success of IP. IP has been designed to work with different link layers (wired, wireless, etc.). This is of critical importance when considering the heterogeneous nature of the IoT. IoT consists of tiny constrained devices which communicate wirelessly but also of more powerful

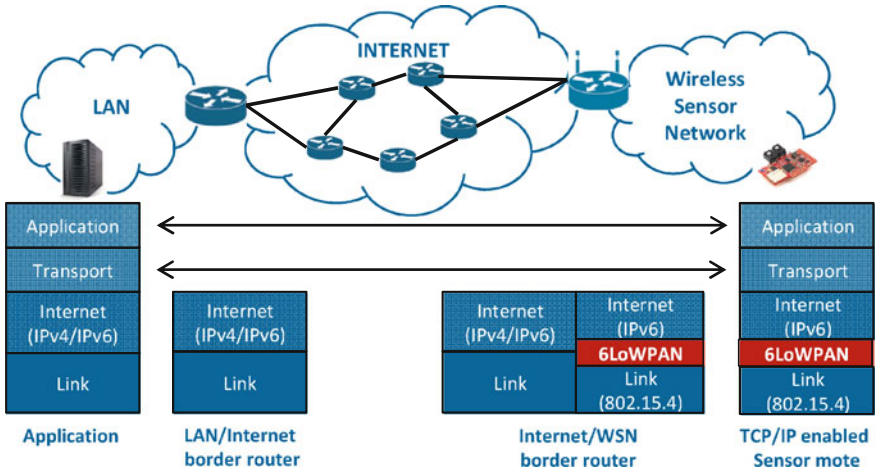


Fig. 1 Embedded IP technologies decrease the architectural complexity by replacing application gateways with border routers

devices connected to the network with wired communication. The layered architectural principle of IP facilitates the communication between these heterogeneous devices. In addition, IP has been designed based on the end-to-end principle, so that to decouple the networking technology from the upper application protocols which could evolve independently. The network is transparent to the application logic which is understood only at the end points. Without the end-to-end principle, application functionalities should have been pushed into the network with consequent increase of the architectural complexity. This would have hampered the huge application potential of the Internet and would have prevented IP to become the preferred networking protocol for a variety of applications and data, such as text, voice, video, etc.

The advantages of the end-to-end design principle will be reflected in the IoT if IP technology will be adopted as much as possible. Application developers can design applications independently of the networking mechanisms and of the internal functionality of embedded devices. In addition, IP technology enables the development of applications and systems which are based on widely used protocols such as HTTP, Simple Network Management Protocol (SNMP) and Dynamic Host Configuration Protocol (DHCP) [9].

Figure 1 gives an idea of the simplification of the network architectural complexity when embedded IP based technology is deployed in the IoT. The figure shows a simple case in which an application running in a certain Local Area Network (LAN) needs to access a 6LowPAN enabled WSN via the Internet. The WSN is connected to the Internet via a WSN border router. On the WSN side, the border router is equipped with a low power wireless data link interface, such as IEEE 802.15.4, which is typical in WSNs. On top of the data link layer, the 6LowPAN adaptation layer compresses/decompresses IPv6 packets going from/to the Internet. With IP being the networking protocol in the WSN, the edge device connecting the Internet

to the IoT is nothing else than a router with a 6LowPAN adaptation layer. The figure also shows that the application server and the sensor being accessed have the same TCP/IP stack and consequently the end-to-end principle can be preserved across the three network segments (LAN, Internet and IoT). If the sensor had a different non TCP/IP based stack, the edge device connecting the IoT to the Internet would not be a border router but an application gateway. It is straightforward to understand that application gateways know the application logic and consequently the end-to-end principle would not be preserved between the Internet and the IoT. In addition application gateways decrease the network flexibility and hamper the scalability. Whenever there is a protocol upgrade or a new application is developed, the change has to be reflected in the application gateways [9].

3 Embedded Web Technologies

The previous section has discussed on the benefits of introducing standard IP networking into the IoT. This section gives an overview of how also the Web concept and technologies have been considered to be ported to the IoT for the realization of the WoT.

Web Services have been considered as a solution to facilitate the integration of the IoT with the Web. Web services allow application developers to rely on platform-independent and reusable software, which drastically simplifies the application development process. In addition, IoT monitoring and actuation functionalities can become part of the mashup process which has been one of the key success factors of the Web 2.0. Developers can seamlessly mash up functionalities provided by traditional web services with the ones offered by embedded devices. This opens up the possibility to develop innovative and creative applications [14, 15].

There are two major Web service paradigms: *WS-** and *Representational State Transfer (REST)*.

In the WS-* architecture, the service functionality and interfaces are declared in a Web Service Description Language (WSDL) file. It uses Simple Object Access Protocol (SOAP) messages with an Extensible Markup Language (XML) payload and an HTTP based transport protocol to provide remote Procedure Calls (RPC) between clients and servers. There has recently been an effort to adapt the paradigm to constrained devices [16]. A lightweight version of WS-*, called Device Profile for Web Services (DPWS) has also been proposed [17].

REST is an architectural style based on the concept that everything is modeled as a resource. Resources are abstractions controlled by the server and identified by a Universal Resource Identifier (URI). Resources are accessed by clients in a synchronous request/response fashion using an application protocol. REST is not tied to a particular application protocol. However, the vast majority of REST architectures nowadays use HTTP. HTTP manipulates resources by means of its methods GET, POST, PUT, etc. One of the main advantages of REST is that the resources are decoupled from the services and therefore can be arbitrarily represented by means

of various formats. These means that more compact formats with a lower overhead compared to XML can also be used, such as Java Script Object Notation (JSON). JSON has a compact data representation but does not have the flexibility of XML. A better compromise between the compact representation and the flexibility of XML is a binary representation of XML called Extensible XML Interchange (EXI). In [6] other binary representations of XML are described, such as Fast Infoset, a standard defined by International Telecommunication Union Telecommunication Standardization Sector (ITU-T), and Binary Extensible Markup Language (BXML), defined by the Open Geospatial Consortium (OGC). The author in [6] shows that EXI provides a more efficient encoding up to 97 % smaller than the equivalent XML. The efficiency of EXI has also been demonstrated in [18].

REST has widely been accepted as the preferred architecture for the WoT [8, 19, 14]. Firstly, REST is based on loosely-coupled stateless interactions and consequently Web services based on RESTful Application Programming Interfaces (APIs) can be reused and combined in a straightforward manner. Secondly, REST has a more lightweight structure than WS-* which makes it more appropriate for networks of constrained devices. The better performance of REST compared to WS-* when applied to constrained devices has been demonstrated in [20] in terms of overhead, memory footprint and completion time. Thirdly, the World Wide Web is largely dominated by REST based Web services and therefore porting the REST principle to the IoT would simplify its integration with the Web.

Various research papers have already demonstrated the feasibility of applying REST/HTTP to the IoT for the realization of the WoT.

In [19], the authors propose to reuse the open, interoperable and standard principles of the modern Web architecture to integrate physical objects to the Web. The field of application considered in the work is the emerging field of Smart Home. Since in REST/HTTP the discovery is done by following links and consequently there is not mechanism for device discovery, the authors develop an ad hoc device discovery procedure which locates and register information related to embedded devices. In order to achieve interoperability with heterogeneous embedded devices and services, the authors propose to use the Web Application Description Language (WADL), an XML based file format that provides a machine readable description of HTTP based Web applications. WADL is language and platform independent and it models the resources provided by an embedded device and the relationships between them. The authors use the Tmote Sky, a sensor/actuator programmable platform, on which they implemented five different resources: three resources to monitor temperature, humidity and illumination, one resource to activate a LED lamp and one resource to measure the energy consumption of some electrical appliances. The system developed in this work also includes an application framework which supports the creation of mashups and advanced rules in a simple way and in any programming language that supports HTTP.

In [8], the authors apply the REST/HTTP principle to embedded devices. They use the Sun SPOT platform, a WSN platform which includes a Web server which exposes sensors, actuators and internal functionalities as REST resources represented in JSON and accessed using HTTP requests. The actuators are controlled using synchronous

HTTP calls (client pull) while monitoring functionalities are accessed through a syndication mechanism via an external Atom(Pub) server. The authors implement a *physical mashup* mechanism which apply typical Web 2.0 patterns to embedded devices. They show the benefits of the system by developing two applications: an energy aware Web dashboard, which allows people to control and experiment with the energy consumption of their appliances, and the physical mashup editor, which enables users to create physical mashups by means of visual components and without requiring any programming skills.

In [21], the authors apply embedded Web technologies to the building automation domain. They present an approach to integrate tiny wireless sensor/actuator nodes into an IP/Web based network. Sensor and actuator nodes run a small Web server on top of a TCP/IP stack. The Web server exposes the sensing and actuating functionalities in a RESTful way and therefore can be accessed by means of HTTP methods. The data is represented in the JSON format. Service discovery based on multicast DNS messages enables the system to integrate new devices without additional configuration effort. The authors measure the average response time of the embedded Web based devices when responding to periodic HTTP GET requests and show that the system offers an acceptable performance given the limited computing power and memory constraints of the hardware platform.

Although the feasibility of adopting REST technologies in the WoT has been proven and its advantages have been discussed as shown in the aforementioned research work, there are still performance problems in the realization of a REST based WoT, especially for the verbosity of HTTP which makes it not appropriate to constrained devices. The authors in [8] evaluate the performance of the proposed REST based WoT system and argue that the verbosity of HTTP does not prevent highly efficient applications to be implemented. However, when battery life time and latency is critical, more optimized protocols which minimize the latency have a better performance. When a sub-second latency can be tolerated and battery life time is not critical, the advantages of HTTP outweigh the loss of performance. In addition to the battery life time and latency, when applying the web service principle to network of constrained devices the difference between IoT applications and traditional Web applications need to be taken into account. IoT applications are short-lived and web services reside in battery operated devices which most of the time sleep and wake up only when there is data traffic to be exchanged. In addition, such applications require a multicast and asynchronous communication compared to the unicast and synchronous approach used in standard Web applications [6]. These characteristics of IoT applications suggest that HTTP is not appropriate for the WoT and that a more optimized REST based application protocol needs to be adopted.

4 Constrained Application Protocol

The previous section has described how the REST/HTTP principle has been applied to smart objects to realize the WoT and how HTTP might not be optimal for constrained devices as a consequence of its verbosity and of its design principles which

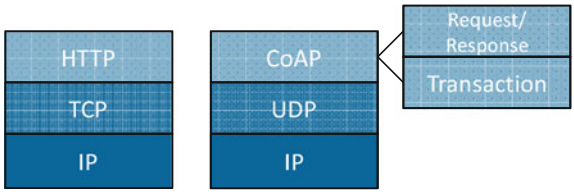


Fig. 2 Difference between the HTTP and CoAP stacks

are not suited for the IoT. The unsuitability of HTTP for the IoT has motivated the effort of the IETF CoRE Working Group in defining a new protocol called CoAP. This section is dedicated to a description of CoAP with main accent on the major difference with HTTP and on the interoperability effort being done during the standardization process. We will also review the main CoAP related literature work.

CoAP is a web transfer protocol optimized for resource constrained networks. It consists of a subset of HTTP functionalities which have been re-designed taking into account the low processing power and energy consumption constraints of small embedded devices such as sensor motes. In addition, various mechanisms have been modified and some new functionalities have been added in order to make the protocol suitable to the IoT. The difference between HTTP and CoAP protocol stacks is illustrated in Fig. 2.

The first significant difference between HTTP and CoAP is the transport layer. TCP flow control mechanism is not appropriate for IoT applications and its overhead is considered too high for short-lived transactions. In addition, TCP does not have multicast support. CoAP is built on top of UDP which has lower overhead and provides multicast support.

CoAP is organized in two layers. The *Transaction* layer handles the single message exchange between end points. The *Request/Response* layer is responsible for the requests/responses transmission and for the resource manipulation. The dual layer approach allows CoAP to provide reliability mechanisms even without the use of TCP as transport protocol. In fact, a message can be labeled as *Confirmable* which implies that it is retransmitted using a default timeout and exponential back-off between retransmissions, until the recipient sends the *Acknowledgement* message. In addition, the two layer approach enables asynchronous communication which is a key requirement for the WoT in which embedded web servers are not always able handle requests immediately.

Differently from HTTP, CoAP includes a built in observation mechanism which allows a client to subscribe to a resource hosted on a CoAP server and to receive notifications upon a change of the resource state [22]. The protocol also includes a technique for discovering and advertising resource descriptions. The mechanism is based on two approaches recently defined by IETF for HTTP: the well-known resource path */.well-known/scheme* (RFC 5785), which defines a "well-known location" where to store resources to be easily located, and the *Web Linking* (RFC 5988), which defines a framework for typed links [10]. Since the resource descriptions need to be machine interpreted, the IETF CoRE group has also worked on the

standardization of the *CoRE Link Format* which standardizes the resource description format itself [23]. In order to achieve a uniform and interoperable resource discovery, a smart object running a CoAP server can provide a resource description available via the well-known URI */.well-known/core*, as done in the HTTP the well-known resource path. The */.well-known/core* resource accessed by a CoAP server returns a list of links to the resources hosted by the server and other link relations.

One of the major design goals of CoAP has been to keep the message overhead as small as possible and limit the use of fragmentation. HTTP has a significantly large message overhead. This implies packet fragmentation and associated performance degradation in networks of constrained devices. CoAP uses a short fixed-length compact binary header of 4 bytes followed by compact binary options. A typical request has a total header of about 10–20 bytes.

In [24], we have compared a REST/HTTP based WSN with a REST/CoAP WSN. We have demonstrated that when using CoAP as application protocol there is a significant performance improvement in terms of energy consumption and response time. The improvement is consequence of the lower number of bytes transferred in a CoAP client-server transaction compared to an HTTP client-server transaction. Thanks to the header compression, after being encapsulated in the UDP, 6LoWPAN and MAC layer headers, the CoAP packet can be transferred into a single MAC frame which has a size of 127 bytes.

Although CoAP is work in progress, it has already been addressed in some research work, such as in [25] and in [26].

In [25], Maenpaa et al. propose an architecture for peer-to-peer (P2P) federation of geographically distributed WSN islands. The WSN islands are low power and lossy networks which use CoAP as application protocol. The border routers of the WSN islands are equipped with cellular radio interface and are organized in a wide area P2P overlay network. The wide area network provides various services such as a common namespace and rendezvous. It uses Resource Location And Discovery (RELOAD), a P2P signaling protocol being designed and standardized by IETF. The P2P system is proposed as a decentralized alternative to the centralized approaches used in CoAP, such as DNS-SD and Resource Directory. In order to integrate CoAP and RELOAD, the authors introduce a CoAP application usage for RELOAD. The application uses basic functionalities such as registration of CoAP resources in the RELOAD overlay, *rendezvous* using dedicated connections for CoAP and *rendezvous* using tunneling and the use of RELOAD overlay as a cache for sensor data. The use case chosen to simulate and study the performance of the approach is road traffic and road condition monitoring in the Finnish highway network.

In [26], the authors propose an approach which moves the application logic form the embedded devices to the cloud. The approach enables the reuse of deployed devices for different applications without changing the firmware. On the device side, the approach is based on the implementation of thin servers, embedded web servers which do not host any application logic and which only provide interfaces to elementary functionalities such as sensor/actuator access and configuration of device parameters to interact with the physical world. The thin server's interfaces are RESTful APIs mainly implemented with CoAP, or with HTTP for more powerful devices.

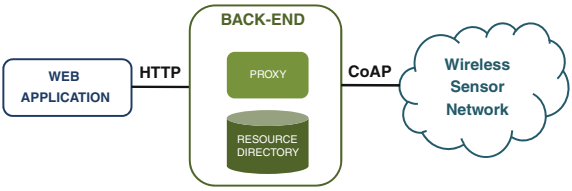


Fig. 3 Building blocks of an end-to-end system integrating a REST/HTTP Web application with a REST/CoAP WSN

The authors argue that in order to simplify the application development, APIs must be self-descriptive and concise, like the ones based on the CoRE Link Format and Microformats. On the back-end side, application logic is based on scripting languages (e.g. JavaScript) which are widely known by application developers. The authors use Californium, a CoAP framework implemented in Java, integrated with JavaScript support for the development of the application logic. In particular, a coapRequest object API provides the same functionality as the well known xmlHttpRequest API.

5 Architecture of an End-to-End System for Integration of a WoT with the Web

The previous sections discussed on the enabling technologies and protocols for porting the REST principle to the IoT and therefore for the realization of the WoT. This section discusses on the integration of the WoT with the Web in order to show how standard embedded web technologies facilitate the integration and simplify the end-to-end architecture. As baseline for the discussion, we use the prototype design and development of a system which integrates a REST/CoAP based WSN with a web application and which facilitates the seamless exploration, access and visualization of WSN data. The platform has been developed for real-world deployment in a greenhouse. However, The core building blocks of the platform can be reused for several application domains which need to seamlessly explore, access and visualize WSN data. An overview of the overall system is given in Fig. 3.

As illustrated in Fig. 3 the end-to-end system consists of three main building blocks: the REST based WSN, the back-end and the application. Sections 5.1 and 5.2 describe the WSN and the back-end, respectively. This section only aims to describe the architecture of an end-to-and REST based system which integrates a WSN with a Web application. Therefore, the description of the monitoring application and its functionalities is out of the scope of this work.

5.1 The REST Based WSN

The WSN is a set of Zolertia Z1 motes. Zolertia is a programmable wireless sensor platform with a radio chip supporting the 2.4 GHz IEEE 802.15.4 standard. The sensor motes run Contiki OS, a widely known operating system for constrained devices such as wireless sensors and actuators. Each mote runs ContikiMAC, which is the Contiki default MAC protocol, the Contiki 6LowPAN stack and RPL routing protocol, and a Contiki low-power CoAP server called Erbium [27]. The motes are equipped with an embedded temperature sensor and have been expanded with additional humidity and light sensors to meet the requirements of greenhouse monitoring. Temperature, humidity and light measurements are exposed as CoAP resources.

For interoperability purpose, the resources are exposed based on the IPSO Application framework, which defines a template used to expose REST resources hosted by CoAP or HTTP servers [28]. The framework groups resource types into *Function Sets*, each one having a recommended root path under which its sub-resources are organized. Each Function Set is assigned a Resource Type parameter which allows the resource to be discovered. The resource semantics defined are compatible with the HTTP Web Linking and with the CoRE Link Format. The data encoding format used is JSON.

5.2 The Back-End

The back-end is the core component of the end-to-end system. It is located on the edge beetween the WSN and the Internet and aims to integrate the WSN with the Web application in a seamless and transparent way. Its main modules are implemented in Java and run on a standard Linux Ubuntu machine physically located at the edge between the WSN and the Internet. It is therefore equipped with two data link layer technologies, being Ethernet and IEEE 802.15.4. The former one is a standard Ethernet card integrated in the machine, while the latter one is realized with a Zolertia Z1 mote running Contiki, connected to the USB port and configured to be a border router.

The CoAP functionalities in the back-end box are enabled by Californium, the Java based CoAP framework introduced in [26]. Although being relatively new, there are already several CoAP implementations for several programming language [29]. Our choice for Californium is driven by its modular and flexible architecture and by the fact that it provides HTTP-CoAP proxy functionality (see Sect. 5.2).

The backend integrates the WSN with the Web application via two major functionalities: the transparent WSN resource access and the Web based WSN resource search and discovery. The two building blocks enabling these two functionalities are described in the two following sub-paragraphs.

5.2.1 Cross-Protocol Resource Access

Resources exposed by CoAP based sensor motes can be accessed by requests sent by a CoAP client. However, when integrating the WSN with a Web application, the data need to be accessed via HTTP requests sent by an HTTP client, unless the HTTP client is equipped with CoAP functionalities. The solution of having CoAP functionalities at the Web client side has been proposed in [30]. The author proposes a CoAP plugin for the Mozilla Firefox Web browser. The plugin registers a protocol handler for the scheme *coap* and therefore allows a user to discover and access the WSN resources by directly entering CoAP URIs in the address bar or by followed links on Web pages. Although this is an interesting solution, it cannot be largely adopted while integrating WSNs with the Web. Firstly, it requires users to have the CoAP browser installed which is not a typical scenario. A typical web scenario consists of a user accessing, via an HTTP client, a web application running on a web server, similar to the case shown in Fig. 3. In addition, having a CoAP client running at the user side implies that the communication will go over UDP not only in the WSN network segment but also in the Internet segment. Although CoAP over UDP includes reliability mechanisms based on retransmissions, TCP should remain the preferred transport protocol in the Internet segment for better performance and higher reliability [31].

The problem of accessing CoAP servers from HTTP clients has been addressed by the IETF CoRE group who has defined the guidelines for mapping HTTP requests into CoAP ones and vice versa [31]. The mapping operation is executed in a reverse proxy usually placed at the edge of the constrained network, as illustrated in Fig. 3. The HTTP-CoAP proxy enables cross-protocol resource access by transparently mapping an HTTP request coming from a Web application into a CoAP request to be forwarded to a CoAP server and vice versa.

It is worth underlining that although the proxy translates two application protocols, it does not have the same level of complexity as application gateways. As mentioned in Sect. 2, an application gateway is aware of the application logic, which has major impact on the architectural complexity. The HTTP-CoAP proxy is transparent to the application logic because the two protocols are both based on REST and therefore they use the same set of verbs to manipulate the resources. The proxy only maps the Request/Response model of CoAP into HTTP (the REST requests/responses). The underlying messages exchanged on the Transaction layer have no effect on the proxy functions [31].

In our system shown in Fig. 3, the application server runs in an IPv4 network. When the application wants to query a CoAP server to retrieve its data, the Web application server generates a GET request with the URI http://sensor.domain.com/data, where *sensor* is the name of the the CoAP server, *domain.com* is the network domain of the WSN and of the proxy, *resource* is the requested resource, which in our case can be temperature (*temp*), humidity (*hum*) or light (*light*). Once received the GET request, the proxy generates the URI coap://sensor_ipv6_address.domain.com/resource where sensor_ipv6_address is the IPv6 address of the CoAP server. Once the CoAP server sensor receives the URI, it generates a response with the data embedded into

a JSON file. The proxy maps the CoAP response into an HTTP response and sends it back to the Web application server.

In order to decrease the dependency between the Web application and the proxy and between the proxy and the CoAP servers, the system relies on the Domain Name System (DNS) which guarantees that the client does not need to know the IPv4 address of the proxy and the proxy does not need to know the IPv6 addresses of the wireless sensor motes. This implies that the DNS keeps an A record resolving the IPv4 address of the proxy and an AAAA record resolving the IPv6 address of the sensor mote.

The description of the main mapping operation shows that the proxy is protocol agnostic, meaning hat the Web application accesses WSN resources as if it were using HTTP and without having knowledge that the sensor data are accessed via CoAP. The transparency is guaranteed by a homogeneous URI mapping. This implies that during the mapping operation, the authority and the path of the resource URI do not change. The only translation undergone by the URI is in the scheme (*http:* translated into *coap:*).

5.2.2 Resource Search and Discovery

As described in Sect. 4, CoAP includes a technique for discovering and advertising resource descriptions. A CoAP server can provide a description of the available resources via the well-known URI */.well-known/core* which returns a list of links to the resources hosted by the server. There are cases in which this approach does not guarantee good performance in terms of search and discovery capability. As an example, we have calculated the time needed to discover all the resources in the WSN with an increasing number of sensor motes. We define the *discovery time* as the time needed by the CoAP client to send a GET request to the */.well-known/core* interface of every sensor mote and to receive the description of all the resources exposed by the motes. For a number of motes equals to 9, 13 and 25 the discovery time is around 8, 13 and 42 s, respectively. This indicates that the resource discovery mechanism might degrade the performance in case of large networks and when trying to explore resources exposed by several sensor motes. Another case in which the CoAP discovery mechanism might hamper the search and discovery functionality is when there are sleepy motes which sleep for long periods of time [23].

The IETF CoRE group has addressed this problem by introducing the Resource Directory (RD), a registry which stores the descriptions of the resources exposed by the CoAP servers. When exploring the WSN resources, the web application can send a query to the RD instead of accessing the */.well-known/core* interface of every sensor mote.

CoAP server motes register their resource descriptions in the RD by sending a POST message which also indicates the period of time during which the registration is valid. Once this period has elapsed, the RD erases the corresponding resource description, which indicates that the CoAP server is not reachable. The RD keeps the registration alive upon receiving refresh messages from the CoAP server.

Resources registered on the RD can be queried by means of CoAP GET requests which can also include filtering options which limit the scope of the lookup to certain resource types or to a limited set of CoAP servers. The resource discovery process is made transparent to the Web application by means of the HTTP-CoAP proxy module described in Sect. 5.2.

6 Discussion

Up to now we have seen how embedded web services allow smart objects to become web servers which expose resources in a REST fashion as any traditional Web server. Although they facilitate the integration of the IoT with the Web, embedded web services might have a cost for constrained devices and therefore might cause performance degradation. Firstly, the CoAP/UDP/6LoWPAN/ stack significantly increases the memory footprint which might not be tolerated by certain severely constrained devices. In addition, the overhead introduced by the IP and web oriented communication protocols might have a significant impact on the power consumption and consequently on the battery lifetime. Battery duration remains a serious bottleneck for the realization of the IoT vision and therefore needs to be taken into consideration when engineering networks of smart objects.

In order to have a general idea of the degradation introduced by embedded web mechanisms in a WSN, we show a simple experiment which measures the average mote's power consumption in two different data collection simulation scenarios. In the first scenario the data is collected via CoAP request/responses, while in the second via a simple application which uses the same request/response approach as CoAP, but does not include the reliability and other more sophisticated features provided by CoAP. The reference application has been developed on top of the same UDP/6LowPAN stack as the one used in the CoAP scenario. In both cases the requests are initiated by the client running in the back-end. The experiments have been executed with Cooja, an emulator of Contiki based sensor motes. Cooja provides a tool called Energest able to estimate the energy spent by a mote while transmitting, receiving, processing and while being in idle state. The aim of this experiment is not to provide an exhaustive and quantitative analysis of the power consumption but to give a qualitative indication of the cost that needs to be paid in terms of power consumption when using CoAP. The experiment has been executed with three mesh networks containing 9, 13 and 25. The results have been calculated as average values of several independent simulation runs. The results are shown in Fig. 4.

Figure 4 clearly shows that when using CoAP the average power consumption of a server mote is significantly higher than the simple application built on top of UDP. This is mainly consequence of CoAP's reliabilty mechanism based on retransmissions. The diffeence between the two protocols is about 20 % for 13 motes, 25 % for 15 motes and 30 % for 25 motes. In fact larger networks are likely more congested

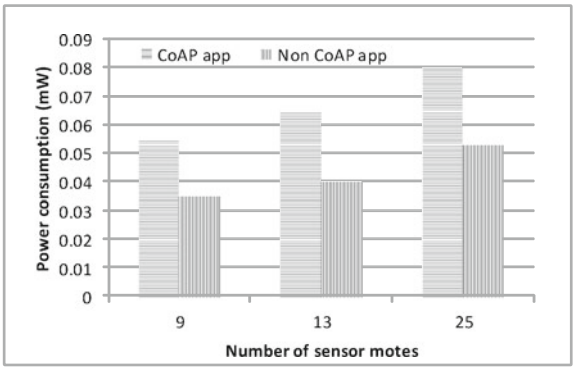


Fig. 4 Power consumption of a CoAP and non-CoAP based application

and consequently CoAP executes more retransmissions. The higher the number of retransmissions, the higher the power consumption.

Although having an impact on the mote's power consumption, the more sophisticated mechanisms of CoAP are needed to increase the network reliability. Figure 5 shows the success rate measured in the two scenarios described in the previous experiment. The success rate is defined as the ratio between the number of responses successfully received by the client and the number of requests sent by the client. CoAP achieves a very high success rate in all the three simulated networks while the application without CoAP has a success rate of about 95 % for 9 and 13 motes and drastically reduced (86 %) for 25 motes.

These two simple experiments show that a key aspect of IoT communication protocols is to find the right balance between power consumption and reliability. This is also true and rather straightforward for protocols in other wired and wireless communication networks as well. But the constrained nature of smart objects makes the performance optimization of IoT protocols significantly more complex. This has been one of the main reasons why the IoT has largely been dominated by several service specific communication protocols. Communication engineers have mainly tried to design protocols highly optimized for specific tasks and application domains. Highly specialized proprietary protocols were also used by monitoring and control system vendors to force customers to be locked into a long term relationship.

Proprietary and domain specific protocols can no longer be the adopted solution in the present and future IoT. Sensor and actuator networks are no longer a niche technology used in a few specific sectors. The trillion smart objects vision suggests that we will be largely surrounded by sensors and actuators. This is also consequence of the recent emerging open hardware based technologies which allow people to build their own connected objects even without engineering skills. The ubiquity of sensors and actuators will unlock huge opportunities for the creation of new innovative services. But a necessary requirement is that the IoT will be based on

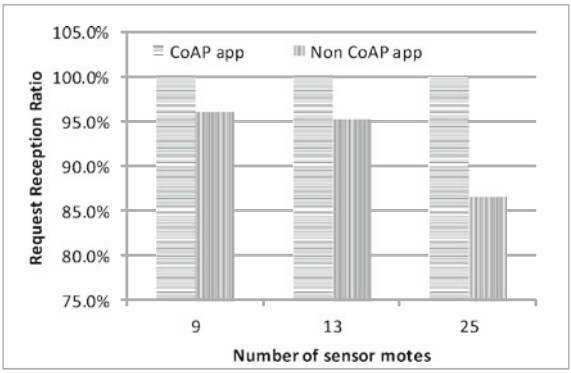


Fig. 5 Success rate of a CoAP and a non-CoAP based application

open standard communication technologies which enable interoperability and plug-and-play capabilities [3].

Interoperability has been the main reason for the success and large scale adoption and deployment of IP and Web technologies. It has been the result of a large open standardization effort which has guaranteed that IP equipment made by different manufacturers are interoperable in a plug-and-play fashion and that Web client and servers can communicate in a platform and programming language agnostic way. The IETF has reproduced the same interoperability driven standardization process for the IoT communication protocols. This is proven by the CoAP plugtest events organized by the IETF CoRE group in collaboration with the European Telecommunications Standards Institute (ETSI). These events aim to test the interoperability of CoAP in a multi-vendor and multi-service environment. The first CoAP plugtest event has seen the participation of several research organizations and industrial players. A total of 3406 tests were made, to test several CoAP functionalities between CoAP client/server implementation of different institutions. More than 90 % of the tests succeeded and demonstrated the robust interoperability nature of CoAP [29].

Although the complexity of communication protocols increases when designed for heterogeneous devices with different functionalities and constraints, open standard communication protocols based on Internet and Web technologies are a way to deliver long term value. They eliminate the vendor lock-in problem, decrease the cost of network maintenance and create opportunities for new and innovative business models. This has been widely accepted by the industrial world and demonstrated by the fact that some industry driven communication protocols which were born without Internet and Web functionalities are being transformed into Internet/Web oriented protocols. An example is ZigBee. ZigBee is a proprietary specification for wireless communication between sensors and actuators maintained and published by the ZigBee Alliance which include several big industrial players. ZigBee is based on the IEEE 802.15.4 radio link layer. The network and transport layer are proprietary and not based on the TCP/IP protocol stack. The application layer in ZigBee is organized in profiles, each profile providing standard interfaces and device definitions for product interoperability in a vertical application domain, such as home automation,

building automations and telecom services [9]. In particular, the application profile related to energy services, called Smart Energy Profile (SEP), has gained momentum especially in the US market. In its 1.0 version, the Smart Energy profile does not support IP/Web communications as a consequence of the fact that the ZigBee stack is not IP based. However, when the smart grid market has identified the need for interoperability, the ZigBee Alliance has started a significant re-design of the ZigBee stack and of the SEP. The network/routing layer of the new ZigBee IP stack is now based on the IETF 6LowPAN and RPL, while the transport layer on TCP and UDP. AS for the SEP, its 2.0 version is based on the REST architecture and requires the use of HTTP for interoperability. The use of CoAP as application protocol and UDP as transport protocol is also considered in the case of more constrained devices. The encoding format adopted is EXI.

The migration of a mature and industry driven protocol like ZigBee towards IP/Web technologies proves the fact that the IoT community has accepted that open standards will significantly boost the IoT. There will surely be cases in which more specialized and non IP based protocols will continue to exist. But this will likely happen only in certain vertical domains which need specific functionalities. The trillion smart objects vision will necessarily require a migration from the siloed approach, with service specific sensor and actuator networks, to the open and interoperable IoT and WoT which will pave the way for the realization of an exciting Internet of Services.

7 Conclusion

This chapter gave an overview on the introduction of standard IP/Web technologies in the IoT. We discussed on the benefits of 6LowPAN and showed how it improves interoperability and reduces the overall architectural complexity. We described how embedded Web technologies based on REST/HTTP have been adopted in the IoT for the realization of the WoT. We introduced CoAP, an HTTP like protocol which is more appropriate for constraint devices. We presented the prototype design and development of a system which integrates a REST/CoAP based WSN with a web application. We described the architecture of the back-end system which provides Web based WSN resource search and discovery functionality, enabled by the CoAP based Resource Directory, and the transparent WSN resource access, enabled by the HTTP-CoAP proxy. We showed that the proxy is protocol agnostic and transparent to the application logic. We showed that IP/Web technologies for heterogeneous devices makes it more complicated to find a balance between power consumption and reliability. However, open standard communication protocols based on Internet/Web technologies are a way to deliver long term value and therefore need to be a key ingredient of the future IoT.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Net. **54**,(15) 2787–2805 (2010)
2. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. Comput. Net. **10**(7), 1497–1516 (2012)
3. Ko, J., Eriksson, J., Tsiftes, N., Dawson-Haggerty, S., Vasseur, J. P., Durvy, M., Terzis, A., Dunkels, A., Culler, D.: Beyond interoperability pushing the performance of sensor network IP stacks. In: 9th ACM Conference on Embedded Networked Sensor Systems, November 2011
4. Zorzi, M., Gluhak, A., Lange, S., Bassi, A.: From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. IEEE Wireless Commun. **17**(6), 44–51 (2010)
5. Dunkels, A., Eriksson, J., Finne, N., Österlind, F.: Low-Power IPv6 for the internet of things. In: 9th International Conference on Networked Sensing Systems, 1–6, June 2012
6. Shelby, Z.: Embedded web services. IEEE Wireless Commun. **17**(6), 52–57 (2010)
7. Hui, J.W., Culler, D.E.: IPv6 in low-power wireless networks. Proc. IEEE. **98**(11), 1865–1878 (2010)
8. Guinard, D., Trifa, V., Wilde, E.: A Resource oriented architecture for the web of things. In: Internet of Things 2010 International Conference (IoT 2010), November 2010
9. Vasseur, J.P., Dunkels, A.: Interconnecting Smart Objects with IP: The Next Internet. 1st ed. Morgan Kaufmann. Burlington, Massachusetts (2010)
10. Bormann, C., Castellani, A.P., Shelby, Z.: CoAP: an application protocol for billions of tiny internet nodes. IEEE Int. Comput. **16**(2), 62–67 (2012)
11. Shelby, Z., Hartke, K., Frank, B.: Constrained Application Protocol (CoAP). Internet-Draft, draft-ietf-core-coap-13, 2012 (Work in progress).
12. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., Alexander, R.: In: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550, 2012
13. Gaddour, O., Koubâa, A.: RPL in a nutshell: a survey. Comput. Net. **56**(14), 3163–3178 (2012)
14. Zeng, D., Guo, S., Cheng, Z.: The web of things: a survey. J. Commun. **6**(6), 424–438 (2011)
15. Guinard, D., Trifa, V., Pham, T., Liechti, O.: Towards physical mashups in the web of things. In: 6th IEEE International Conference on Networked Sensing Systems, June 2009
16. Priyantha, N.B., Kansal, A., Goraczko, M., Zhao, F.: Tiny web ser- vices: design and implementation of interoperable and evolvable sensor networks. In: The 6th ACM conference on Embedded Network Sensor Systems (SenSys '08) 253–266, 2008
17. Jammes, F., Smit, H.: Service-oriented paradigms in industrial automation. IEEE Trans. Industr. Inf. **1**(1), 62–70 (2005)
18. Castellani, A. P., Gheda, M., Bui, N., Rossi, M., Zorzi, M.: Web services for the internet of things through CoAP and EXI. In: IEEE International Conference on Communications (ICC), 2011
19. Kamilaris, A., Trifa, V., Pitsillides, A.: The smart home meets the web of things. Int. J. Ad Hoc Ubiquitous Comput. **7**(3), 145–154 (2011)
20. Yazar, D., Dunkels, A.: Efficient application integration in IP-based sensor networks. 1st ACM Workshop on Embedded Sensing Systems for Energy Effciency in, Buildings, 43–48 November 2009
21. Schor, L., Sommer, P., Wattenhofer, R.: Towards a zero-configuration wireless sensor network architecture for smart buildings. In; Proceedings of 1st ACM Workshop On Embedded Systems For Energy-Efficient Buildings (BuildSys: pp. 31–36. USA, Berkeley) 2009
22. Hartke, K.: Observing Resources in CoAP, Internet-Draft, draft-ietf-core-observe-07, 2012 (Work in progress)
23. Shelby, Z.: Constrained RESTful Environments (CoRE) Link Format. RFC 6690, 2013 (Work in progress)

24. Colitti, W., Steenhaut, K., De Caro, N., Buta, B., Dobrota, V.: Evaluation of constrained Application Protocol for Wireless Sensor Networks. In: 18th IEEE International Workshop of Local and Metropolitan Area Networks (LanMan), 1–6 November 2011
25. Maenpaa, J., JIMENEZ Bolonio, J., Loreto, S.: Using RELOAD and CoAP for wide area sensor and actuator networking. EURASIP J. Wireless Commun. Net. **2012**, 121. Springer (2012)
26. Kovatsch, M., Mayer, S., Ostermaier, B.: Moving application logic from the firmware to the cloud: towards the thin server architecture for the internet of things. In: 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), July 2012
27. Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power CoAP for contiki. In: 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, October 2011
28. Shelby Z., Chauvenet, C.: The IPSO Application Framework, draft-ipso-app-framework-04, 2012, (Work in progress)
29. Lerche, C., Hartke, K., Kovatsch, M.: Industry adoption of the internet of things: a constrained application protocol survey. In: 7th International Workshop on Service Oriented Architectures in Converging Networked Environments, September 2012
30. Kovatsch, M.: Demo abstract: humanCoAP interaction with copper. In: 7th IEEE International Conference on Distributed Computing in Sensor Systems, June 2011
31. Castellani, A., Loreto, S., Rahman, A., Fossati, T., Dijk, E.: Best Practices for HTTP-CoAP Mapping Implementation, Internet-Draft, draft-ietf-core-http-mapping-02, 2013 (Work in progress)

High-Level Internet of Things Applications Development Using Wireless Sensor Networks

Zhenyu Song, Mihai T. Lazarescu, Riccardo Tomasi, Luciano Lavagno and Maurizio A. Spirito

Abstract Wireless Sensor Networks (WSN)-based data gathering solutions were envisioned from the beginning of the Internet of Things (IoT) paradigm because of their important characteristics such as low-cost, long-term versatile sensing and actuation capabilities, and distributed resilient bidirectional communications. However, many technologies converge into WSN platforms from several disciplines. Their complexity and rapid pace of change may prevent most WSN and IoT players to keep up, since they need to focus on low cost and short development time targets. Also important is the quality assurance to prevent the degradation of the overall reliability perception needed for wide WSN adoption. Consequently, a versatile and reusable WSN platform may benefit both industry and academic research by speeding up and facilitating the distributed WSN application programming. The framework presented provides the application developers with a set of development tools and software component libraries that allow high-level application architecture description, graphical definition and simulation of the component behaviour, and automatic generation of both network simulation models and code for target node programming.

Z. Song · R. Tomasi · M. A. Spirito
Pervasive Technologies Area, Istituto Superiore Mario Boella, Turin, Italy
e-mail: song@ismb.it

R. Tomasi
e-mail: riccardo.tomasi@polito.it; tomasi@ismb.it

M. A. Spirito
e-mail: spirito@ismb.it

M. T. Lazarescu (✉) · R. Tomasi · L. Lavagno
Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, Turin, Italy
e-mail: mihai.lazarescu@polito.it

L. Lavagno
e-mail: luciano.lavagno@polito.it

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_4,

1 Internet of Things and Wireless Sensor Networks

More than a decade ago, the Internet of Things (IoT) paradigm was coined [1], in which the computers would be able to access data about objects and the environment without human interaction. Complementing human-entered data was important in order to increase the accuracy and pervasiveness of physical world information in ICT systems at a sustainable cost.

Initially adopted and popularized by the Auto-ID Center at MIT and market analysts [2, 3], the IoT concept received growing attention until recently it became one of the most hyped terms. The vision encompass the ability to uniquely identify, represent and access objects at any time and anywhere, in an Internet-like virtual structure.

Traditionally, radio-frequency identification (RFID) was one technology considered key enabler for the IoT paradigm. Alongside, other low-cost identification and communication technologies are also used: device-centric technologies, such as near field communication (NFC), Bluetooth, barcodes, Quick Response (QR) codes, ZigBee, digital watermarking, and networking technologies, like 3GPP Long Term Evolution/Long Term Evolution-advance (3GPP LTE/LTE-A), Vehicular Ad-hoc Networks, Wireless Sensor Networks, and Wireless Local Area Networks. These blend the IoT concept into the Internet of Everything (IoE) vision, where nearly *everything* is connected to the Internet using machine-originated or machine-to-machine (M2M) communications. Almost all smartphones, tablets and PCs are always connected, and so are becoming other objects (like home appliances, cars, clothes, door locks, light bulbs, toys, thermostats, vending machines, industrial instruments, data collecting sensors) to provide a variety of services without human intervention.

The sheer versatility of the IoT paradigm makes it suitable for an ever increasing variety of applications. In turn, these applications make use and rely on the many diverse technologies and devices that concur to its realization. And all this widening diversity makes increasingly difficult to define "typical" requirements for the devices and their programming [4].

Along the RFID, Wireless Sensor Networks (WSNs) were considered a key technology for IoT paradigm capillary pervasiveness able to bring richer, versatile low-cost sensing and actuation to applications. Intrinsically versatile, WSNs are suitable for a wide variety of applications and systems, with very diverse requirements and characteristics. However, this makes difficult to define specific application requirements and research directions, WSNs remaining a challenging multidisciplinary area. An efficient implementation very often requires a good collaboration between users, application domain experts, hardware designers, and software and firmware developers.

As a long-term value, the IoT technologies allow going beyond application-specific "vertical" pervasive systems. A single "horizontal" ecosystem of interoperable objects and services can seamlessly co-operate to provide new services and optimization opportunities. In this vision, a key role is attributed to the tools that support their modeling, design and development, mainly because these can unify

computing paradigms and development workflows among vendors, architects, integrators and application developers.

2 Developing IoT Systems: Context, Scenarios and Challenges

We will analyze several representative scenarios where the IoT technology is used attempting to extract a subset of abstract use-cases and their common key features. Based on this analysis, we will then extract the key challenges that impact on the IoT development tools and workflows.

2.1 A Reference Workflow for IoT Developments

Figure 1 shows a possible reference development workflow for IoT applications that allows to introduce the IoT development context.

Traditionally, requirement definition starts by considering scenarios of independent vertical domains or markets, also named "silos," e.g., Intelligent Transportation System, Smart Grids, Logistics.

Multiple use cases can be defined within any of these scenarios. This phase normally entails a thorough analysis of domain-specific aspects, including the analysis of the expected uses and needs of key stakeholders and users. Even though a significant part of the IoT vision value lies in its "horizontality," a main usage scenario is usually needed to secure the initial deployment funding.

The initial analysis can produce requirements of different styles and granularities. However, for adequate specificity, they must be mapped or split among a pre-defined set of architecture components. Standard Architecture Reference Models (ARMs) are defined to this purpose, such as that of the IoT-A project[1]. For instance, the mapping defines the information view of the system including all data structures, i.e., the models and protocols employed to share information across different components.

High-level architectures are often split into more controllable components within detailed component design phases. These are then implemented by teams with specific expertise (e.g., communication, interoperable services, low-power embedded systems) and skills for specific hardware/software platforms and tools. At this stage, new components are often integrated with COTS (Commercial Off-The-Shelf) devices at various levels. Where available, these provide specific functions within the system under development, including proper interplay with pre-existing legacy systems where needed.

After the integration and the optional validation phase, the system is ready for commissioning. This consists in its deployment and configuration in the final use

[1] The IoT-A consortium. Internet of things architecture. http://www.iot-a.eu/

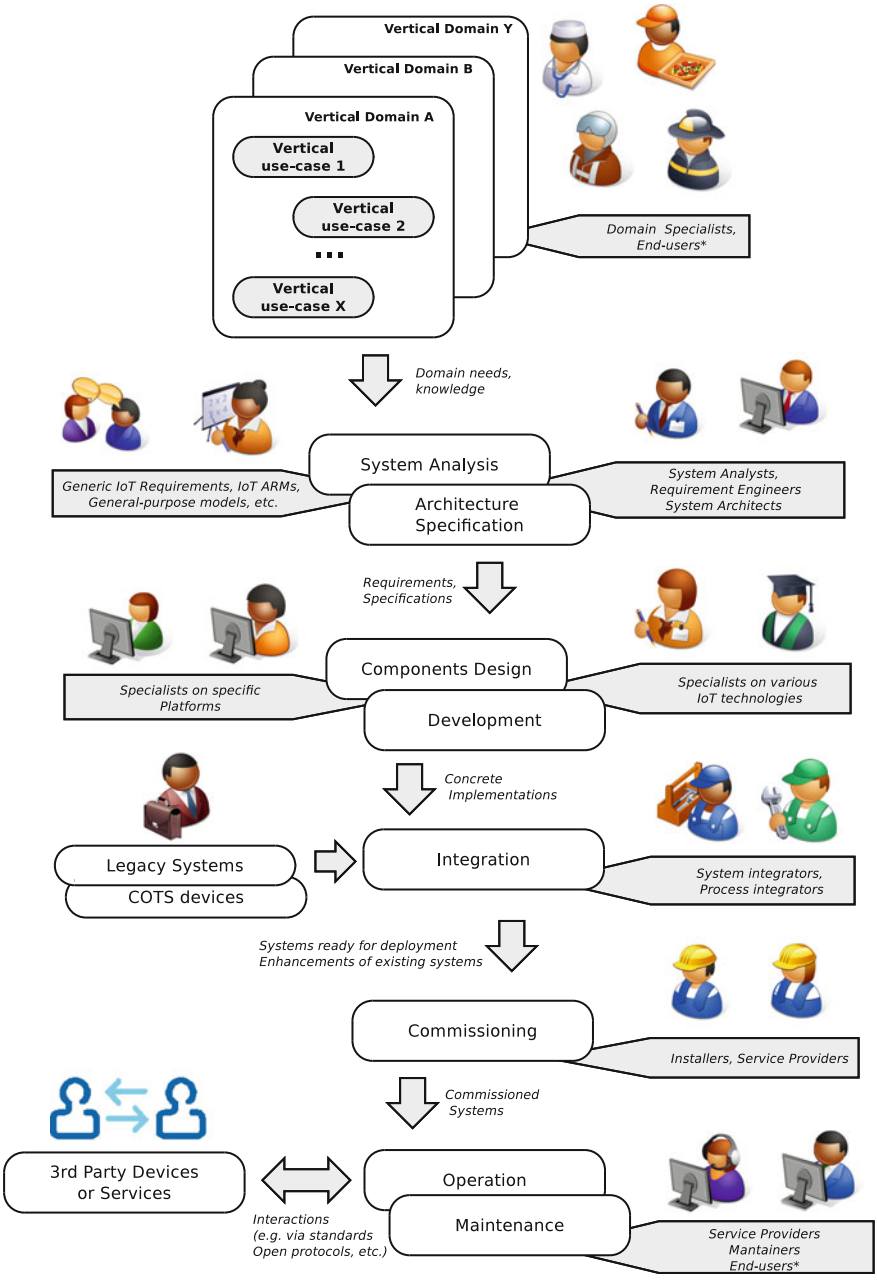


Fig. 1 IoT system design and development work-flow

scenario, and is often associated with the physical delivery to the end-users and their training. Finally, the operational system and receives local or remote maintenance.

The tools can support the development in different phases of the workflow. At high-level architecture definition they help describing and modeling all key aspects of the domain in a consistent information view. They should allow different levels of typing, specification, and semantic annotation for the project elements, such as protocols, key variables and parameters. Such models and parameters, possibly standardized, should be associated to all components and are converted in data structures for project algorithms.

The development-support tools cover a prominent role in detail design and development of specific components. At these levels, they typically link the design high-level abstractions (objects, variables) defined to their specific implementations. They help especially the joint developments by several different experts and to unit test the components.

In this phase, tool interoperability is key. Experts in specific IoT fields should use their specific tools, typically focusing on a well-defined set of technologies. At the same time, these tools should allow sharing models, interfaces and parts of the development process.

During the integration, commissioning and operation stages (including maintenance), the tools should support the development of IoT components able to interoperate with a broad set of COTS systems, legacy devices, and installation and maintenance tools. This may entail supporting the broadest set of available open protocols and standards, where applicable.

IoT Application Scenarios

A wide collection of studies on IoT markets, applications and use cases is available in the scientific, technical and business literature. However, due to high diversity and heterogeneity of the IoT definitions and deployments, a widely accepted taxonomy of IoT applications is still elusive.

Referencing a common IoT application taxonomy is nevertheless important for the IoT community. It simplifies the information exchange and improves the targeting of investments in IoT technical developments. Moreover, it is also a primary tool helping to keep into account the various requirements. Common requirements across different domains shall influence general-purpose IoT features, while domain-specific requirements shall be considered for optional features to be supported by IoT development in specific markets.

This section provides an overview of the main IoT application domains currently identified by the research community. For each domain is suggested a set of key challenges that may potentially impact the IoT development-support tools. A restricted set of possible use cases is analyzed in each domain to identify specific challenges.

The analysis follows the categorization suggested in the IERC (European Research Cluster on the Internet of Things) Clusterbook [5], modified based on other international road-maps, white papers and review articles [4, 6–17].

Smart Cities

Smart Cities vision involves a broad set of applications aiming to make cities more sustainable, safe and enjoyable thanks to new ICT-supported models of cooperation among citizens, institutions and companies (e.g., utilities, service providers). Most applications require a large number of networked devices and systems deployed on city-wide scale, making of Smart Cities a primal playground for IoT technologies [18].

The applications of interest range from systems supporting urban mobility and its safety (e.g., Smart Parking, Traffic Congestion, Intelligent Transportation Systems), to applications monitoring or optimizing assets and critical infrastructures in cities (Structural Health, Smart Lightning, Smart Roads), to systems monitoring and protecting the citizens' quality of life (Noise Urban Maps, Waste Management).

Structural health focuses on monitoring the status of buildings, bridges or other infrastructure, e.g., by sensing their vibrations [19]. The challenges include large scale deployment, unmanned operation, self-healing, processing and detecting events from raw data via (potentially complex) algorithms running on-board the nodes, embedding data transmission and management logic on tiny devices, low-power constraints, inter-node coordination (coherent data time-stamping).

Traffic congestion monitors car and pedestrian traffic leveraging fixed [20] or mobile [21] sensors. The challenges include large scale deployments, integration of data from heterogeneous devices and data sources, interoperability of different vendor sensors, interoperability with other systems (e.g., mobile terminals, legacy traffic monitoring platforms), input and output location-awareness, distribution of intelligence (e.g., due to time constraints), minimizing the commissioning and maintenance effort.

Participatory sensing collects, analyses and shares data through participation of local communities of volunteers [22]. The challenges include working with heterogeneous data and sensors, the need for data annotation and semantic interoperability, integration of crowd-sourced knowledge, interoperability with community-provided devices, trust and privacy.

Smart Environment

The Smart Environment domain include any system monitoring or preventing critical events in wide, unpopulated areas where significant environmental risks have been identified. Such areas require constant monitoring and alerting about critical events, such as fire in forests, landslides and avalanches in mountain areas, earthquakes in seismic areas, exceptional air or water pollution events in heavy industrial or safety-critical areas.

Forest fire detection in remote forest areas to alert public authorities [23]. The challenges include large-scale deployments without fixed communication infrastructure, strict low-power constraints, effective use of energy harvesting, unmanned operation, integration of multi-hop communication solutions, need for efficient data processing on board constrained nodes, real-time reporting of critical events, self-configuration, self-healing.

Remote seismography collects seismic data from remote areas [24], e.g., by means of microphones or seismic-acoustic sensors. The challenges include installations in harsh conditions, need for data-processing techniques to trigger event detection, synchronization and time-stamping of sensed data, reliable data retrieval with constrained bandwidth.

Pollution monitoring of potentially dangerous gases in both urban and remote areas using fixed or mobile sensors [25]. The challenges include data collection from fixed and mobile installations and geotagging, large sets heterogeneous sensor calibration or self-calibration, power supply management.

Smart Water

Smart Water applications include all systems for water monitoring in both civil and natural environments, such as water quality monitoring (e.g., presence of chemicals) in rivers or in water distribution infrastructure, water leakage detection in pipes or buffer tanks, monitoring of levels of rivers, dams and reservoir, e.g., for flood or drought early warning.

Water leakages detection monitors pipes or distribution networks, e.g., by means of pressure, flow or other types of sensors [26]. The challenges include installation and operation in harsh conditions (e.g., underground), time-correlation of spatial data, unmanned operation, combination and fusion of data from heterogeneous sensors possibly provided by different vendors, unmanned management of sensors in remote locations.

Level-monitoring and flood detection for basins, dams, rivers, lakes checks levels and detects floods [27]. The challenges include continuous monitoring and processing of sensed data, lack of communication infrastructure, strict real-time constraints when critical events occurs, resilient communication, distributed processing and monitoring.

Smart Metering

The Smart Metering domain is a wide area of application. It involves remote monitoring (and control) of a large population of networked meters that provide data for accounting and consumption billing of various commodities, such as electricity (both consumed and generated in the so-called "Smart Grid"), water, gas and oil in storage tanks, cisterns or transportation systems (e.g., water pipes, oil pipelines), level of silos stock. The most mature Smart Metering solutions are currently in the electric power distribution market.

Metering in the smart electric grid are applications for monitoring and accounting consumption via remote, automated meters [28]. The challenges include security, privacy, reliability and transparency, interoperable standardized and certified solutions, low-latency and robust communication for more advanced control applications.

Metering of heating systems for monitoring consumption and efficiency of heating systems, e.g., to optimize or to partition heating costs across different units [29]. The challenges include low cost, ease of installation and maintenance, interoperable standardized and certified solutions, interoperability with legacy systems.

Security and Emergency

The Security and Emergency domain, among the first explored in the WSN for IoT field [30], includes protection of areas sensitive to people intrusion (e.g., perimeter access control) or to environmental factors that increase the risks for people and goods (e.g., monitoring liquids in areas with dense electric outlets, detecting presence of gases and leakages in chemical factories, detecting radiations close to nuclear power stations).

Perimeter control and tracking detects and localizes trespassers (e.g., thieves, enemy troops) using distributed sensors and alarms [31]. The challenges include tight security requirements, resilience to interference and disruption, complex data management and processing for intruder detection.

Disaster recovery solutions assist rescuers after major disasters like radiation leaks or floods [32]. The challenges include resilience to interference and disruption (e.g., many sensors disabled or moved), real-time requirements during emergency, interoperability with mobile networks (e.g., aerial vehicles or land robots), geolocation, self-healing and reconfiguration.

Retail

The Retail domain includes systems to simplify stocking (Supply Chain Control), storage (Smart Product Management), marketing (Intelligent Shopping Applications) and selling (NFC Payment) in shops and malls.

Smart shopping employs marketing-oriented applications in retail environments, e.g., to track customers' behaviour or feed targeted advertisements in malls [33]. The challenges include context awareness (e.g., location-dependent behaviour, time awareness), interoperability with existing business systems, location and time-awareness, fusion of heterogeneous data sources to detect user behaviour.

Stock control for shelves and stocks real-time monitoring in shops to simplify the supply chain management, e.g., alerting for stocks that run out [34]. It includes M2M monitoring of automated vending machines. The challenges include real-time operation, ease of maintenance, integration with legacy enterprise systems, deployment-specific configurations.

E-Payments enable payments via short-range communication technologies like NFC [35]. The challenges include strong security requirements, interoperability with payment systems available on COTS devices.

Logistics

The Logistics domain includes systems to support scenarios involving storage and shipping of goods. It includes good monitoring during transportation (e.g., monitoring of vibrations, strokes, box opening, break of cold chain), detection of improper storage conditions (e.g., flammable materials close to heat sources), location of items (e.g., in storages, harbours) or vehicles (e.g., for navigation support or theft prevention).

Smart transport supports good monitoring during transport, e.g., in trucks or cargo ships [36]. The challenges include monitoring system reliability and trust (e.g.,

anti-tampering), automatic association of sensor readings to goods, interoperability across different logistics platforms.

Smart logistics support good detection and tracking in warehouses [37] and combined monitoring [38]. The challenges include standard identification and addressing schemes, tight integration with Enterprise systems, large scale operations, reliable readings, standard solutions for tagging.

Industrial Control

Industrial control is one of the first application domains for IoT technologies (e.g., for remote monitoring of manufacturing lines through SCADA systems). These systems are typically deployed in manufacturing or process industries in order to remotely check machineries (e.g., M2M applications), diagnose the status and position of moving vehicles or robots or to monitor the conditions of the manufacturing environment (e.g., air quality monitoring in food processing industries). These systems help the transition of industrial automations from custom, closed developments towards more flexible internet-oriented schemes, directly integrated with enterprise and management systems. They simplify data-intensive applications like real-time tracking or predictive maintenance.

Manufacturing applications leverage autonomous M2M interactions to monitor and optimize production lines [39]. The challenges include interoperability across heterogeneous system, context-awareness, fusion of information from various data sources, reliability in harsh industrial environments, large scale operations, strong safety constraints.

Mobile robotics applications in industry in which mobile robots interact with fixed IoT infrastructures, e.g., to support internal logistics in manufacturing plants. The challenges include support for mobile operations, context-aware behaviour, interoperability of robots with fixed sensors.

Smart Agriculture

Smart Agriculture focuses on monitoring the soil used for growing agricultural products, plants, greenhouses, or the environmental conditions (e.g., weather).

Crop monitoring for moisture, salinity, acidity, etc. using sets of chemical sensors, e.g., to ensure product quality [40]. The challenges include operation in harsh conditions, lack of central infrastructure, self-configuration, self-powered operations, cost constraints.

Smart green houses are controlled and optimized via wireless sensors and actuators [41]. The challenges include reliable operation (especially for control), integration of heterogeneous sensors, control logic support using distributed sensors and actuators, simple maintenance, low-cost.

Smart Animal Farming

Closely related to Smart Agriculture, Smart Animal Farming enhances the productivity of animals for meat and related products (e.g., milk, eggs) by monitoring the animal health conditions at different stages, animal tracking and identification (also used for product traceability), and living environment.

Animal monitoring looks for animal behavior, e.g., to detect states of sickness or stress [42]. The challenges include data integration from heterogeneous sensors, processing of large sensor datasets, data processing algorithms and techniques, operation in harsh conditions.

Meat traceability from farm to fork for animal-derived products, including monitoring of physical parameters during various operation phases [43]. The challenges include intermittent connectivity, standard identification and addressing schemes, tight integration with enterprise systems, large scale operations.

Domotics and Home Automation

The Domotics and Home Automation are centered on homes and commercial buildings. They include applications to improve home safety, ease of use and sustainability, e.g., through resource consumption monitoring (energy, heating, water), remote control of appliances, optimization of air and ventilation or prevention of theft and intrusion.

Building automation monitors and optimizes building subsystems, such as lighting or heating, ventilation and air conditioning (HVAC) [44]. The challenges include interoperability up to the semantic level, integration with legacy solutions, need for standard protocols and interfaces, secure and safe operation.

Appliances control, automatic and remote, for, e.g., energy usage optimization in Smart Grid Environments [45]. The challenges include interoperability up to the Semantic level, standard protocols and interfaces, ease of use, context-awareness features, real-time operations, secure and safe operations, privacy.

e-Health

e-Health systems focus on monitoring of patients or assets needed to assist the health of people in any specific condition, including, e.g., disabled or elderly or under special training or dietary situation. Specific applications include, e.g., detection of fall of elderly people living alone, or monitoring of stocks of medicines in hospitals.

Patient surveillance in Ambient-Assisted Living (AAL) applications [46] and other systems to monitor patients inside or outside hospitals [47], including monitoring of living parameters and position tracking. The challenges include ensuring the privacy of patients' data, data protection, management of sensed data, support for patient mobility, wearable systems – low-power operation.

Fall detection, prevention and reaction for elderly or disabled people [48]. The challenges include location-awareness, data processing complexity for fall detection, mobile and nomadic operations, reliability requirements.

IoT Application Challenges

The analysis of the main IoT application scenarios in Sect. 2.1 leads to a set of challenges that are currently faced by the field, summarized in Table 1.

Table 1 IoT application challenges by use case domains

Challenges	Domains												
	Smart Agriculture	Home Automation	e-Health	Logistics	Retail	Security and emergency	Smart animal farming	Smart cities	Smart environment	Smart grid	Smart Metering	Smart water	Industrial control
Cloud-based large data processing						○	○	○		○	○		
Collaborative (untrusted) management								○					
Deployment-specific configuration		○	○	○	○								○
Distributed data and logic processing		○						○					
Extraction of context and environment		○		○	○			○					○
Harsh environment	○					○	○		○			○	○
Heterogeneous devices	○	○	○		○		○	○		○	○	○	○
Interoperability up to semantic level		○						○					
Interoperability with processes and systems		○		○	○	○	○	○			○	○	○
Large-scale and/or decentralized				○			○	○	○	○	○	○	○
Limited bandwidth	○						○	○	○				
Location awareness			○		○	○		○	○				
Low cost	○			○									
Low power operation	○		○					○	○				
Low-latency and quality of service	○	○			○	○			○	○	○		○

(continued)

Table 1 (continued)

Challenges	Domains												
	Smart Agriculture	Home Automation	e-Health	Logistics	Retail	Security and emergency	Smart animal farming	Smart cities	Smart environment	Smart grid	Smart Metering	Smart water	Industrial control
Mobility			○				○	○	○				○
No communication or power infrastructure	○								○			○	
On-board processing			○			○		○	○			○	
Privacy		○	○		○			○		○	○		
Reliability	○		○	○		○				○	○		
Resilience to disruption and interference						○				○	○	○	
Safety		○	○			○				○	○		○
Security		○	○		○	○			○	○	○		○
Self-calibration, self-configuring, self-healing	○	○	○	○		○		○	○	○	○	○	○
Simple and low-cost maintenance	○	○	○	○	○	○	○	○	○	○	○	○	○
Standardization and certification		○		○	○		○	○		○	○		○
Time correlation	○	○			○			○	○			○	

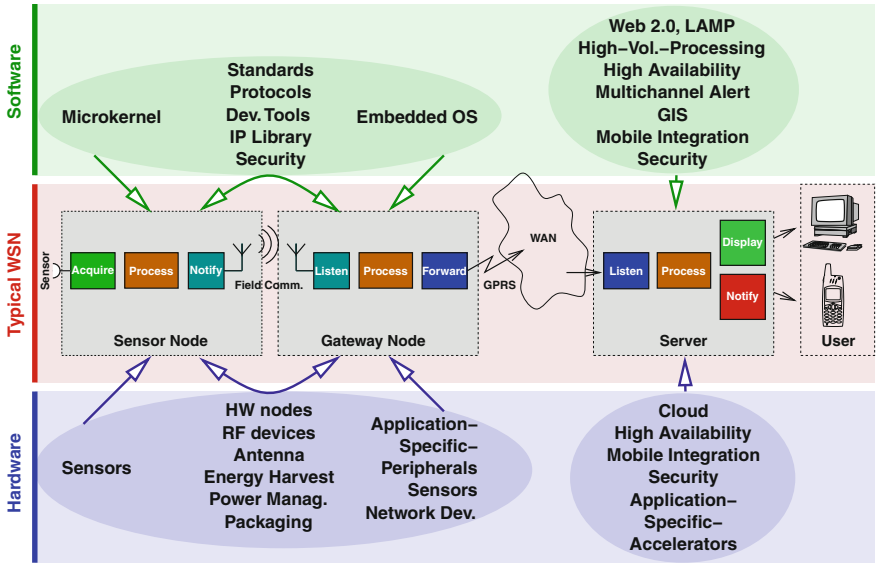


Fig. 2 A wide variety of technologies converge into a typical tiered WSN platform

Beyond the technical/scientific factors mentioned in the table, it is important to remember that the IoT domain is also impacted by non-technical challenges of different nature, including social, process reliability, training, awareness, collaboration and economic challenges, which should be addressed as the IoT domains and technologies evolve and mature.

3 WSN Application Development Overview

The IoT paradigm valued since the beginning the low-cost long-term WSN-based data gathering solutions able to provide versatile sensing and actuation through distributed resilient bidirectional communications.

Early WSN projects made use of large-scale ad hoc, multi-hop, unpartitioned networks of randomly deployed and mostly fixed and homogeneous tiny, resource-constrained devices. This traditional definition is still applicable to a significant class of applications (e.g., for the military and outdoor environmental monitoring domains).

For instance, Fig. 2 shows the main components of a typical tiered WSN platform for monitoring applications and some of the hardware and software technologies it is made of. The block diagram in the middle section shows one or more sensors (transducers) that are attached to the sensor nodes. These are typically small (in size, resources and cost) and are suitably deployed within the application field to

perform the measurements needed by the application. The sensors typically have limited data storage, processing, energy reserves, and short-range communication capabilities that can be used to process locally and forward the field measurements required. Often they are expected to operate reliably and unattended for very long periods (years).

The data sent by the sensor nodes ultimately reach a gateway (or sink) node, either directly or after a few forwarding hops through the network. The gateway nodes usually have larger memory, data processing, energy reserves, and long-range out-of-field communication capabilities. Among other functions, the gateway nodes can, e.g., buffer and further process the field data, and communicate with the application server to transfer field data and configuration instructions.

The server is typically located outside (and can be far away from) the application field and has processing, storage, and supply power to reliably store field data for long periods of time, and allow to retrieve them in formats suitable for visualization or further processing. Also, the server may be able to issue alerts on specific field conditions or events.

However, many emerging WSN applications cannot be adequately defined in this way and the structure of this generic platform and its components often need to be adapted to best suit the application requirements and environmental conditions [49]. For instance, the sensor nodes, the gateways, or the server can be omitted for specific applications. Also, the communication channels between the sensors (if any) and with the gateways can be uni- or bidirectional, and form peer-to-peer, star, tree or mesh topologies.

The top and bottom sections of Fig. 2 shows some of the many and multidisciplinary technologies, both hardware and software that concur to the realization of the WSN platform for most applications. Mastering all these technologies is difficult for most WSN and IoT players, and keeping up with their rapid pace of change adds to the challenges. Very often an efficient collaboration between users, experts of the application domain, hardware designers, and software and firmware developers is necessary to achieve efficient WSN implementations [50].

Considering all these, a definition of the WSN design space can be used as a framework for discussion and coordination of both research efforts and the development of flexible WSN platforms, which can be easily adapted to effectively support the many application needs. Both industry and academic research can benefit from a versatile and reusable WSN platform, easy to tailor to support a broad range of applications, and including development tools, support for several hardware devices, a library of software components for both the embedded and server parts of the application, and tools to support the field planning and deployment [51]. However, the development and maintenance of such a platform can be very costly and also involve professionals with very diverse expertise, from advanced web and UI design to low level firmware, and from IDE design to code generation techniques for high level application descriptions. Quality assurance is also very important, since overall unreliability perception is still a limiting factor to WSN wider adoption.

Hardware vendors usually provide WSN development platforms tailored around the devices they produce. These usually consider most typical uses of the vendor's

hardware, being often hardware-centric rather than application-centric. As such, they can require significant extension or adaptation to cover a broader range of applications and may significantly lag the state of the art in the WSN field, since they follow the progress of the producer.

Most of the open projects for embedded (TinyOS, freeRTOS, Contiki) or server (GSN) components of the WSN platform are largely vendor- and hardware-independent. But they also fail to address all development needs of a complete WSN application, often requiring significant adaptation and integration for an effective use by system integrators.

3.1 WSN Development Abstractions

Wireless sensor networks often operate under tight resource and budget constraints, their range of application continuously diversifies, and, traditionally, WSNs are developed and deployed by single organizations. These circumstances often facilitate the optimization across the typical layers, which are valuable for component interchangeability and reuse for better reliability and lower costs [52]. The early WSN advances and applications, some of which are shown in Table 2, clearly display the tendency to blur the boundaries between layers even after the efficiency benefits of the layers was generally recognized.

Along with the perceived lack of reliability, application development is still an issue that hinders WSN wider adoption. Real-world applications continue to rely mostly on node programming very close to the embedded operating system, which increase the development time, costs, skill requirements, and limit the component reuse and the overall application reliability.

Consequently, domain experts rarely develop efficient WSN applications [50]. The characteristics and requirements of the WSN devices on the one hand, and the approaches for application development on the other provide a rich set of functions which can meet many diverse applications requirements. However, a good understanding of these is needed in order to choose the best platform and methodology for a given application.

Various programming models were proposed to overcome WSN programming difficulties. Considering the level of abstraction from the operating system (OS) and the operation of the underlying hardware, these can fall under the categories of:

1. OS-level programming;
2. virtual machine (VM) or middleware programming;
3. network macroprogramming.

The first two address the node-level programming [53] while the latter hinges on network-level WSN programming (macroprogramming) abstractions [54].

Table 2 WSN layers are usually unevenly covered by development frameworks and applications. Often layer separation is blurred for better implementation efficiency

	A	T	N	L
Aloha [preamble sampling-based protocol]				●
B-MAC [asynchronous adaptive preamble sampling protocol]				●
BVR [Beacon Vector Routing, greedy point-to-point routing]			●	○
CODA [dynamic Congestion Detection and Avoidance]		○	●	○
CTP [Collection Tree Protocol with hop-based metric]			●	○
Deluge [protocol for network-scale node reprogramming]		●	○	
DIP [probabilistic spatial node Density Inference Protocol]		●	○	
Drain [sink-originated distributed best route discovery]		●	●	○
Drip [registration-based unnamed reliable message dissemination]		●	○	
Flush [adaptive reliable bulk multihop transport protocol]		●	●	○
FPS [Flexible Power Scheduling for distributed power management]		●	●	●
Fusion [of multiple distributed field data over unreliable channels]		○	●	○
GAF [Geographic Adaptive Fidelity for location-based routing]	●	●	●	●
Great Duck Island [habitat monitoring application]	●	●	●	●
Golden Gate Bridge [structural health monitoring application]	●	●	●	●
MintRoute [many-to-one cost-based routing for data collection]			●	●
MultihopLQI [routing protocol based on Link Quality Indicator]			●	○
North Sea [industrial sensing application]	●	●	●	●
PEDAMACS [topology-aware multihop TDMA protocol]				●
PSFQ [Pump Slowly, Fetch Quickly transport protocol]		●	●	●
Redwoods [environmental monitoring application]	●	●	●	●
RMST [Reliable Multi-Segment Transport for guaranteed delivery]		●	●	●
SCP-MAC [Scheduled Channel Polling protocol]				●
S-MAC [fixed-duty sleep-synchronized preamble-sampling protocol]				●
SPIN [Sensor Protocol for Information via Negotiation]		●	●	
T-MAC [synchronized contention-based adaptive-duty protocol]				●
Volcano [application to monitor an active volcano]	●	●	●	●
WiseMAC [synchronized preamble sampling infrastructure protocol]				●
X-MAC [short-preamble low power listening protocol]				●

A Application, *T* Transport, *N* Network, *L* Link

OS-Level Programming

OS-level programming models consist in developing the application logic using directly the resources made available by the embedded OS. Of the latter, one of the first and widely used is TinyOS [55], a component-based OS that allows modular programming using the C-based nesC language [56]. The modules have interfaces that are connected using configurations, the latter being used to describe also the application. The encapsulation provided by the modules hides the implementation details, but the low level of abstraction, the event-based style, and the blocking nature of the operations may complicate the implementation even of simple applications.

Asynchronous messages is one technique to reduce the programming complexity, e.g., TinyGALS [57], which allows a globally asynchronous (message passing

through asynchronous FIFO queue), locally synchronous (within sequential modules) programming model. SOS [58] implements message passing between modules using a priority queue, and CoMOS [59] uses preemptive inter-module message handling.

More OS layers or the extension of the event model can also reduce the programming complexity. SNACK [60] facilitates the design and reuse of parameterizable service libraries and applications can be defined by combining services. T2 [61] provides a *telescopic abstraction* hybrid made of a horizontal decomposition (to support different hardware devices at low level) and a vertical decomposition (to support platform-independent functionality at high level). OSM [62] addresses the limitations of the static association of states to actions and the implicit change of the program state in a pure event-driven programming model. OSM separates the state and transitions, the latter and the associated actions being a function of state and events, adds parallel composition for concurrency and hierarchical composition for program refinement.

Thread abstraction is also proposed to address the limitations and complexity of an event-driven programming model. While the event-driven model tends to optimize the microcontroller sleep time, the static per-thread stack allocation may be too expensive for the typically limited RAM of the sensor nodes. However, the blocking execution context of threads can significantly simplify the programs and the programming.

An early lightweight concurrency model is proposed by Fiber [63] on top of TinyOS, while MANTIS OS [64] provides fully preemptive, time-sliced multithreading. TinyThread [65] implements cooperating multithread as a library for TinyOS with explicit and implicit execution context yielding, and per-thread stack estimation. Protothreads [66] implement a similar multithreading approach for another event-driven OS, Contiki, but without a per-thread stack, to improve efficiency. Y-Threads [67] concept is similar to Fiber, but implements preemptive multithreading with a shared stack for non-blocking computation behaviours and separate stacks for blocking control behaviours. The latter typically need a small stack which leads to better overall hardware resource utilization.

Virtual Machine/Middleware Abstraction

Virtual Machines/Middleware (VM/MW) simplify the remote node reprogramming at the expense of some processing overhead and total code size. The latter can be minimized by limiting the generality of the VMs to subsets relevant to application domain(s). Also, VMs provide platform-independent execution models facilitating code reuse.

Interpreter-based VM as Maté [68] and ASVM [69] are implemented on top of TinyOS and provide increased safety and reduced application code size, while Melete [70] extends Maté for concurrent applications. VMStar [71] allows the remote update of both VM and application code. Remote programming is also included at OS- or MW-level, for complete application or at finer grains (module or thread),

as the necessity to update the application in the field rises. t-Kernel [72] addresses reliability by providing OS protection and virtual memory through application code modifications at load time.

Macroprogramming

Other WSN programming abstractions address the development of applications as distributed processing at the *network* level, providing models and semantics to define the communication and coordination among nodes. Unlike node-centric abstractions, where the network behavior emerges from the processing and interactions defined for single nodes, macroprogramming abstractions allow high-level behavioral descriptions at network-level.

The *programming paradigm* that defines how the program elements (functions, variables, computations) and their interactions can be expressed. It has a significant impact on the learning curve, especially for domain experts that may have limited programming experience.

Application description using *imperative* statements, which indicate explicitly the way to change program state, are by far the most used, be these event-driven (e.g., nesC) or sequential (e.g., Pleiades [73]). *Declarative* programming allows the description of the application goal without explicitly defining how to accomplish it. The programming can be functional (e.g., Regiment [74, 75]), rule-based (e.g., Snlog [76]), SQL-like (e.g., TinyDB [77]), and special-purpose (e.g., Logical Neighborhoods [78]). *Hybrids* mix declarative and imperative programming approaches, such as imperative for node computation and declarative for inter-node communications (e.g., ATaG [79]).

Pleiades extensions to C language allow to address the network nodes and their internal state. Its network-level instructions are executed by only one network node at a time with constructs designed to iterate through nodes, which make it particularly suited to guarantee the distributed execution of concurrent applications. *Regiment* constructs allow to apply functions and store the output on one or more nodes in a region and *Snlog* uses logical programming constructs as predicates, tuples, facts, and rules. The rules express the processing, the facts are the initial tuples (initial state), and the tuples (the data) is structured according to predicates, similar to relational database tables. *TinyDB*, as the earlier *TAG* [80], implements an energy-optimized WSN-wide SQL query system from the base station. A sensor table implements the data model as one line per sensor node and one column for each sensing capability. It can be filled with field data following SQL queries or proactively, simplifying the expression of data collection applications. *Logical Neighborhoods* allows the definition of node connections based on their logical affinity in the application, regardless of their physical location. It is mostly suited for heterogeneous and decentralized sense-and-react applications. *ATaG* is built around the abstract task and data item, with copies of tasks that can run on different nodes and abstract channels that connect data items to the tasks that consume them. It is mostly suited for sense-and-react applications that require complex operations to decide the actions.

4 Model-Based Design, Simulation and Debugging Framework for WSNs

WSN application developing and testing is often labor-intensive and error-prone due to the wide diversity of requirements and application domains. A rich set of tools to support WSN application development, testing and deployment is available addressing different aspects of the process: high-level modeling, architectural design, abstract and detailed code development, code generation, application- and network-simulation, co-simulation, deployment, validation, debugging, performance monitoring and evaluation [55–81].

The rich choice of tools, devices, and techniques can provide adequate functionality to meet most application requirements. However, a single comprehensive approach has not yet emerged (see Sect. 3.1). The widening technology and tool diversity is still difficult to master to obtain satisfactory implementations and the application-domain experts often need to collaborate with the users and developers to achieve efficient solutions.

In response to this need, the WSN development tools evolve to improve the support for functional composition and complex distributed scenarios, with service-driven models playing a key role.

In the following we present a complete and flexible framework that can speed up and facilitate the programming of distributed WSN applications. It is based on the graphical design tools Simulink®[2] and Stateflow®[3], both part of the widely used Matlab[4] environment. It provides the application developers a set of development tools and software component libraries that allow a high-level description of the application architecture, a graphical environment for the definition and simulation of component behaviour, and automatic generation of network simulation models and of code for target nodes.

The application architecture is described using the standard Web Service Description Language (WSDL). The functional design of the architecture components can be described as either white, gray or black boxes. The *white* boxes can be graphically constructed using high-level abstract concurrent models, as Stateflow® diagrams. The *black* boxes are either binary components or code written in the C language. The *gray* boxes are a suitable combination between diagrams and external code.

Figure 3 shows the main phases of the development flow. After the high level application design and simulation phase, the high-level abstract application model can be used to automatically generate several representations, using a Model-Based Design methodology. It can be used to generate network simulation models for the widely used OMNeT++ and MiXiM simulation environments, and for some of the most popular WSN operating systems, TinyOS and Contiki OS for deployment on

[2] Mathworks Simulink http://www.mathworks.com/help/simulink/index.html

[3] Mathworks Stateflow – Finite State Machine Concepts http://www.mathworks.com/help/toolbox/stateflow/index.html

[4] http://www.mathworks.com/products/matlab/

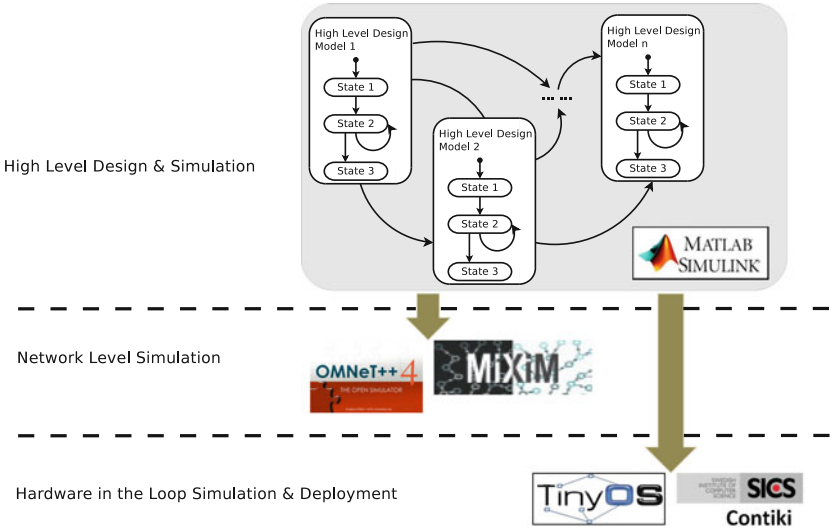


Fig. 3 Overview of the framework development flow

target nodes. Moreover, hardware-in-the-loop simulation can be used to increase the simulation accuracy with real-time physical data from a few node running in sync and communicating with the software simulation.

The integration of the WSN into IoT applications can be simplified when targeting the Contiki OS. In this case, the framework can generate support for web services which can make the node application accessible through typical web-oriented methods (e.g., a normal web browser).

In the following we will illustrate the operation of the platform and how the framework can facilitate WSN application design from high-level conceptual application description, platform-independent graphical design, rapid prototyping, and automatic target code generation for multiple targets.

4.1 Abstract Design Model

The proposed framework [82] makes use of a high-level abstract functional unit, called Abstract Design Model (ADM), to define the structure and behaviour of the target WSN application.

As shown in Fig. 4, each ADM is a programming abstraction, a self-contained unit. It is perceived as a black box by other units and is externally characterized by the tunable attributes and services it uses and provides. The internal details of the ADMs do not depend on external entities, and they can be connected together in a loosely-coupled fashion.

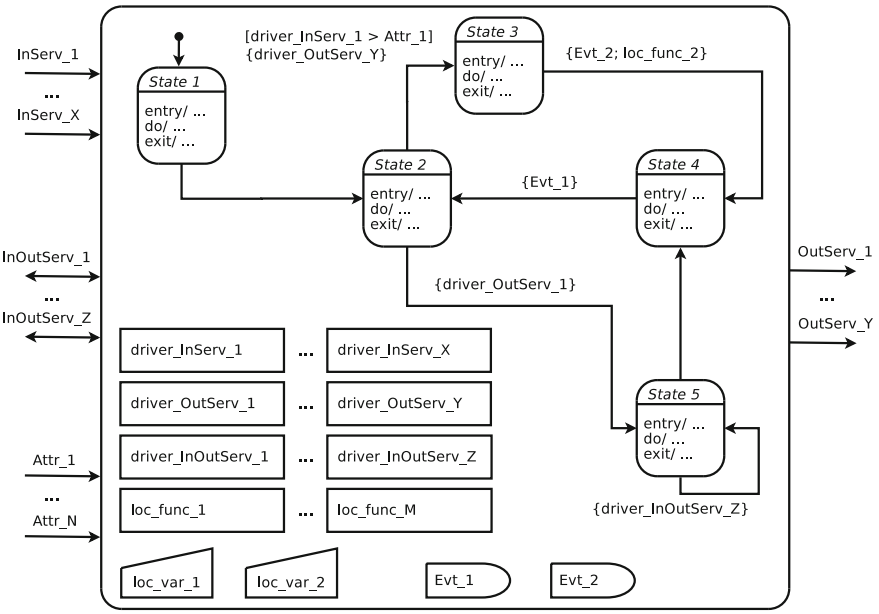


Fig. 4 Components of a typical Abstract Design Model of the framework

Therefore, the WSN applications running on the nodes can be modeled as a set of interconnected ADMs that exchange service messages through their service ports. An inbound service port (e.g., `InServ_1`, ..., `InServ_X` in Fig. 4) imports the incoming messages for an associated service used by the host ADM, while an outbound service port (e.g., `OutServ_1`, ..., `OutServ_Y` in Fig. 4) exports the outgoing messages for a service it provides. Inbound and outbound service ports can also be combined into a bidirectional port (e.g., `InOutServ_1`, ..., `InOutServ_Z` in Fig. 4), useful when a request-response communication pattern is needed. Moreover, the tunable attributes that are exposed by an ADM (e.g., `Attr_1`, ..., `Attr_N` in Fig. 4) allow the developer to modify how it performs without changing its internal logic.

An outbound service port of an ADM can be wired to an inbound service port as long as they share the same service type. These are similar to function calls and can be used to transmit service-specific messages between ADMs, without exposing their implementation. The service ports can also be left disconnected, meaning no incoming messages for the floating input ports and outgoing messages discarded on the floating output ports.

ADM behavior is represented using an event-driven hierarchical Finite State Machine (FSM) described using state charts, as shown in Fig. 4. The logic flow (i.e., the transition from the current state to the next) can be controlled either by its internal default transitions or by service messages imported from other ADMs. These messages are processed by the FSM according to the values of its tunable attributes.

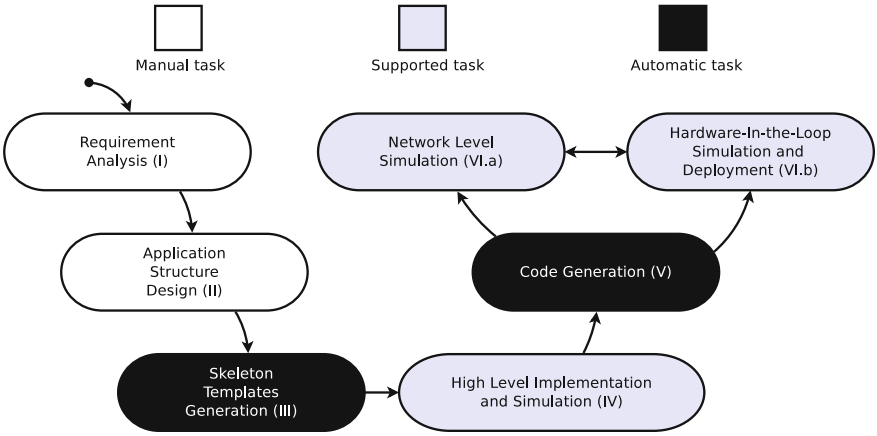


Fig. 5 Framework development flow

Computation results are attached to outgoing service messages sent through output service ports. User-defined operations can be implemented inside each state or between state transitions. They will be executed upon entry, permanence, or exit phases of each state.

Besides tunable attributes, each ADM can have local variables (e.g., `loc_var_1` and `loc_var_2` in Fig. 4) and local events (e.g. `Evt_1` and `Evt_2`).

FSMs can be nested using sub-charts. These can be integrated into higher level FSMs in sequential or parallel execution order, sharing the incoming and outgoing messages, attributes, local variables and local events.

ADMs modularity and loosely-coupled port binding allows to construct an application independent of ADM location within its architecture. ADM port binding can be both intra-node and inter-node. When all application ADMs are within a node, the ports are locally connected and the service messages are implemented by platform-dependent messages or data sharing mechanisms. When not all ADMs are located on the same node, neutral service messages are exchanged by the nodes (e.g., nodes in the radio transmission range) to implement the logic binding of ADM service ports. Moreover, when the ADM service ports are published on the Internet, the developer can raise the target application abstraction level from a single node or local group to the whole Internet.

4.2 Development Flow

Figure 5 shows the V-shape diagram of the development flow of the framework.

There are three types of development activities in the development flow: manual tasks, supported tasks and automatic tasks.

Manual tasks are development activities that are not directly supported by the framework and are performed using other development tools. The *supported tasks* imply some activities performed manually by the developer, but they are also supported by specific tools of the framework. *Automatic tasks* are fully performed by the framework.

The development flow allows the application developer to perform different tasks to design, build, implement or transform the application constituent ADMs. Each ADM can be instantiated in different forms in the framework, for different purposes. For instance, it can be represented by a Simulink®/Stateflow® block in the high-level implementation and simulation phase, for high-level description and debugging. Or it can be an OMNeT++/MiXiM module for large network simulation. It can also be a TinyOS component or Contiki OS process for node deployment, which will be further detailed in the following.

Task I: Requirement Analysis

The first step in the design flow is a *manual task*. It consists in the analysis of the requirements of the target WSN application by listing the desired functionality and supported attributes, e.g., measurands monitored by the application, operations expected from the nodes, state variables exposed as tunable attributes, and criteria employed to validate the application and evaluate its performance.

We will assume a use case where the nodes collect and perform a distributed processing of the data from a temperature sensor. Each WSN node wakes up periodically to carry out the following tasks:

1. sample the temperature values with the desired sampling frequency;
2. collaboratively average these values with those from its one-hop neighbors within a sliding time window;
3. broadcast its calculated average temperature value to its neighbors.

Task II: Application Structure Design

The analysis result can be used to drive the application structure design phase, where the developer conceptually decomposes the application into a set of interconnected ADMs, each carrying out a part of the entire functionality. Since the ADMs are characterized by services and attributes exposed on their boundary, their internal behaviour may not be detailed in this step. The developer just assigns the requirements listed earlier to the constituent ADMs by defining their boundaries. This can be done describing the ADM services and attributes either manually or by importing them from a library (e.g., created in previous designs).

The framework employs Representational State Transfer-based (REST) web service design approaches for their flexible WSN integration. REST is a software architecture for distributed systems [83], such as the World Wide Web that has smoothed

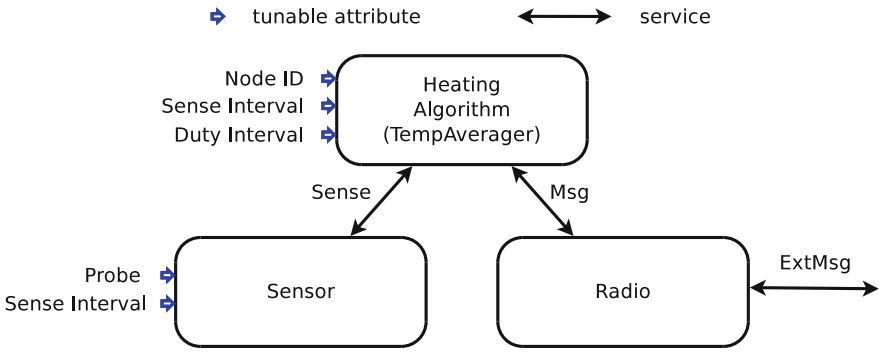


Fig. 6 The model structure of the node application

the way of developing, sharing and reusing IoT applications. It relies on stateless, client-server, cacheable communication protocols, virtually always HTTP.

These help the application developer to construct the ADMs, while WSDL 2.0 is adopted as the ADM description language. The definition of conceptual structures of the application models and related ADMs are *manual task* that can be accelerated using a WSDL graphical editor, such as [84] and [85].

In the proposed use case, the following parameters have been identified as potential tunable attributes for each node:

1. its own node identifier (NodeId);
2. sample refresh interval inside the averaging algorithm (SenseInterval);
3. the sampling period of the temperature sensor (SamplingInterval);
4. the size of the time window to compute averages, which is equal to the node duty interval (DutyInterval).

Based on the requirement analysis (first step in Fig. 5), the design is split on three interconnected ADMs: a `Sensor`, `Radio` model, and `Algorithm` (`TempAverager`) models, as shown in Fig. 6.

The Sensor model samples and pre-processes temperature data. The Radio model interfaces with the protocol stack for short range communication with neighbor nodes. The TempAverager model handles all on-board data processing. Both the Sensor and the Radio models are connected to the TempAverager model to exchange service messages (e.g., `Sense` and `Msg` in Fig. 6).

The developer manually defines a boundary description file for each ADM. These are imported in the framework for the next step, template generation.

Task III: Skeleton Template Generation

This step starts by supplying the ADM description files to the framework to be automatically mapped to skeleton templates defined as a Stateflow® blocks. All

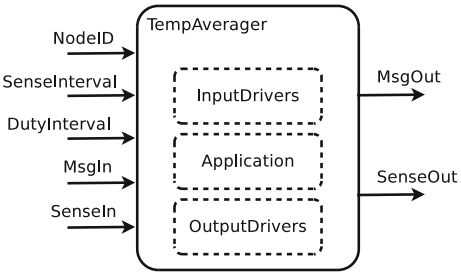


Fig. 7 Overview of the generated skeleton template

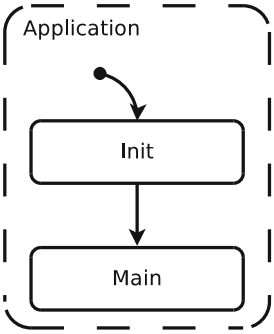


Fig. 8 Application subchart

ADM services and attributes defined in the ADM templates are interpreted as ports or pair of ports (for a request-response service).

For each port, a driver function is automatically created inside the skeleton template, which hides the low-level Simulink® handling of signals and service messages. These functions handle the service messages exchanges through ports. As the input services, each tunable attribute has an input port and a state port variable allowing to set the attribute value from outside the ADM.

Figure 7 shows the skeleton template generated automatically for Stateflow®. Each skeleton template is created with three FSMs that run in parallel: `InputDrivers`, `Application` and `OutputDrivers`.

The `Application` FSM (shown in Fig. 8) contains the ADM main logic.

The `InputDrivers` and `OuputDrivers` FSMs handle the generated input detectors and output actuators for the input and output ports (shown in Figs. 9, 10). These do not need to be changed by the developer.

The generated skeleton templates can be used as the starting point for the implementations in the next step.

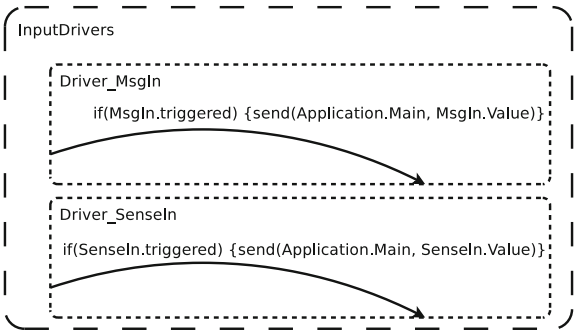


Fig. 9 InputDrivers subchart

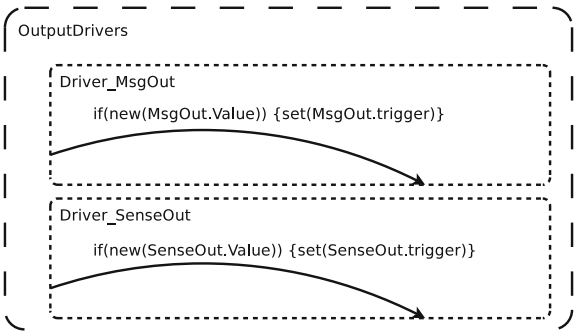


Fig. 10 OutputDrivers subchart

Task IV: High-Level Design and Simulation

In this phase, the developer performs a *supported task* involving the high-level application implementation, simulation and debugging. The implementation consists mainly in skeleton template completion combination:

Skeleton template completion consists in adding the application-specific functions to the skeleton template. These mainly process the imported service messages based on the values of the exposed tunable attributes, and generate the outgoing service messages.
The ADM internal logic is defined at high level, using state charts and flow graphs, independent on the specifications of the target platform. As shown in Fig. 4, the operations defined inside the states will be executed on state entry, permanence or exit phases. The operations defined between two connected states will be executed when the state changes through that state connection. The developer-defined operations can execute any local functions (e.g., `loc_func_1`, ..., `loc_func_2` in Fig. 4) to perform computational tasks or generate outbound service messages.

Skeleton template combination allows to compose the application structure (e.g., as in Fig. 6). This is done by wiring the ADMs outgoing ports to incoming ports and assigning suitable values to the their attribute ports.

Once the implementation of the skeleton templates is completed, the high-level application models can be simulated and debugged at node-level. Afterwards, they can be used to generate the implementations for different simulation or target platforms.

Task V: Code Generation

The skeleton template filled with functional interconnected ADMs, simulated and debugged, can be used for automatic code generation. A framework tool converts the high-level and platform-independent design into target code that runs in different network simulation environments or on target operating systems (OSs) and platforms. These can be:

Simulink®/Stateflow® platform that can be used for node-level and small-scale network simulation. No application translation is necessary;

OMNeT++/MiXiM that can be used for large-scale network simulation. Each ADM that composes the target WSN application is mapped to a *simple component*, the programming unit used by the simulators;

TinyOS which is a popular event-driven embedded OS for WSN nodes. This can be used for code deployment on target nodes. Each functional ADM of the application is automatically converted to a TinyOS *module* component containing the ADM internal logic;

Contiki OS is another popular event-driven embedded OS for WSN nodes. Each ADM is instantiated as a protothread process and the code generated can also be run in the COOJA simulator.
Moreover, Contiki OS allows to generate a RESTful web service support if needed. This allows publishing the application as a standard web service on the Internet without code modifications.

Task VI.a: Network-Level Simulation

Before network deployment, node behaviour and performance should be checked and the target application optimized at node- or network-level.

Model generation converts the high-level functional ADMs to *simple modules* in a node instance for the simulation environment. For the use case proposed, the `Sensor`, `Radio` and `TempAverager` functional blocks are converted to simulator simple modules, as shown in Fig. 11. An "adaptor" object is provided by the development framework as a layer that conveys the messages exchanged across the node instance borders.

Node instances are manually interconnected and configured using the simulation initialization file. It can specify configurations like field size, node receiver sensitivity, initial position and mobility. For example, it can assign values to node properties like `NodeId`, `AvgWinSize`, `SamplingPeriod` and define the properties of the

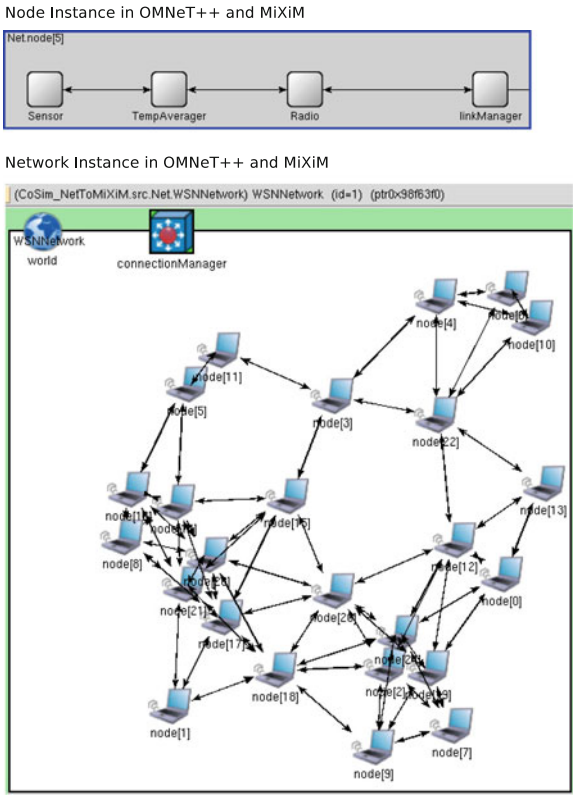


Fig. 11 Simulation in MiXiM and OMNeT++

communication channels between nodes. The network simulation environments also allow the definition of various scenarios, e.g., node mobility or variable communication channel properties.

Any algorithm for the application logic provided by the simulation environments can be easily connected to node instances allowing its performance evaluation in a large-scale simulation.

Task VI.b: Hardware-in-the-Loop Simulation and Deployment

The framework supports automatic code generation for hardware-in-the-loop (HiL) simulations for both target OSs:

TinyOS generation maps each ADM to a *Module* component. These components react and process the internal and external events, service messages, and post internal tasks. They can be manually wired together and assigned values to their attributes using the generated OS *Configuration* file.
HiL simulation can be set up for TinyOS as shown in Fig. 12. The code for the

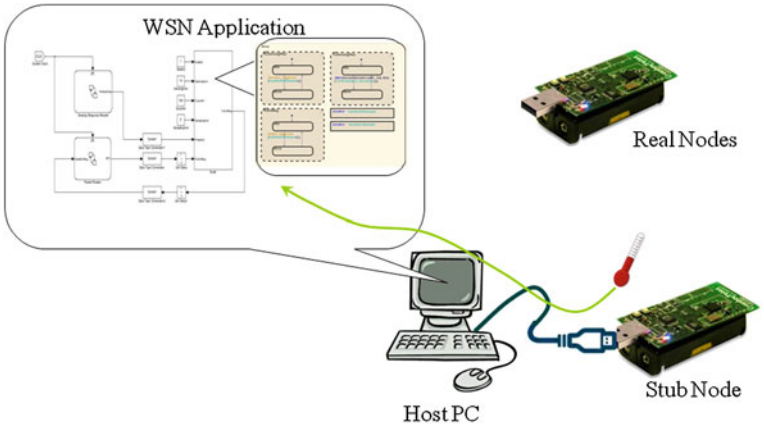


Fig. 12 HIL simulation on TinyOS platform

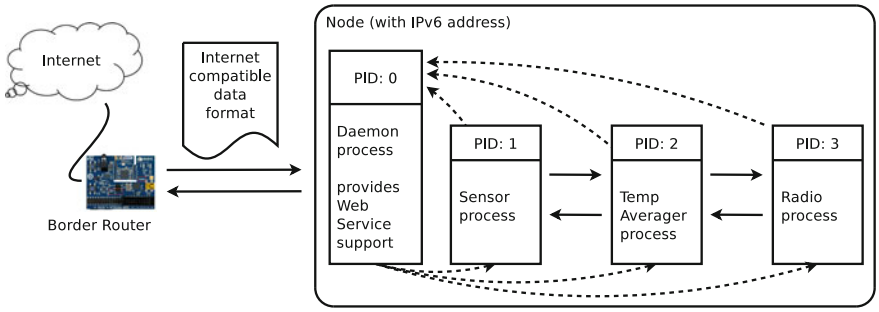


Fig. 13 Deployment on Contiki OS with REST web service support

stub node that is connected to the development framework configures it to act as a gateway bridging the virtual and hardware nodes. It forwards the physical network messages to the framework when queried and broadcasts the virtual network messages to the physical net. The stub can also transfer real sensing samples (e.g., the real temperature value) from its on-board sensors when requested by the virtual nodes.

Contiki OS generation maps each ADM to a Contiki OS protothread process, such as `Sensor`, `TempAverager` and `Radio` Processes in Fig. 13. These run in parallel reacting to internal and external events, processing the service messages using the functions from ADM behavioural description.

Particularly interesting for IoT applications, Contiki OS provides the essential support for web services. The ADM service description WSDL file can be exposed on the Internet to provide a standard remote access to the services, as shown in Fig. 13. The border router is provided by Contiki OS to bridge the Internet and the WSN.

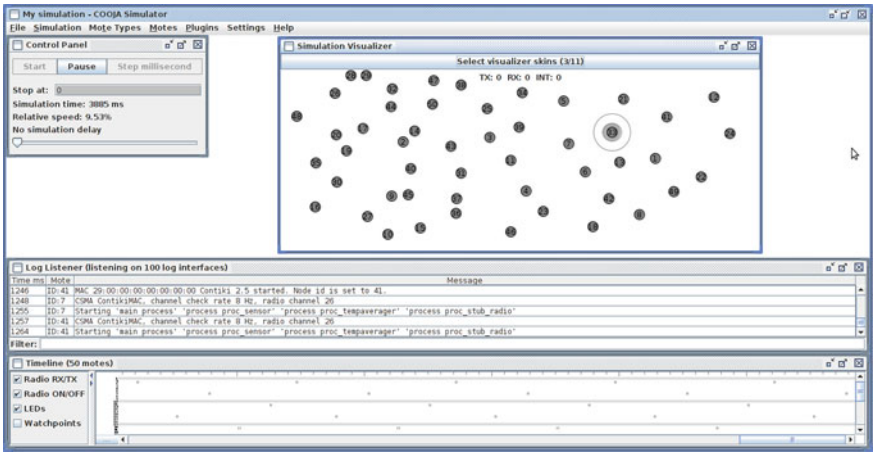


Fig. 14 Large scale network simulation in COOJA

Each Contiki node running applications with REST web service support can be internally decomposed in two type of processes: daemon processes and application processes. A daemon process (identified by PID:0 in Fig. 13) can be automatically generated by the framework. It receives and transforms CoAP Internet requests to internal events, dispatches the events to the application processes, and returns a CoAP response to the Internet requester. Each ADM-exposed service port is considered a resource of the Contiki node and can be reported by accessing the dedicated resource *discover*.

The COOJA simulator can be used to simulate large scale networks of Contiki nodes. It can trace low-level communications among virtual nodes and monitor network traffic, as shown in Fig. 14.

Once the developer validates the simulation results and the deployment performance against the WSN application requirements, the design can be refined and improved.

For instance, the sample application can be used to validate the framework capabilities. The code generation was set to transform the high-level design ADMs to nesC modules, suitable for a simple test-bed set up using Memsic Telos rev. B nodes running TinyOS.

The generated nesC modules (`Radio`, `Sensor` and `TempAverager`) are configured, interconnected and encapsulated in a wrapper nesC module that is then wired in TinyOS to adapt the existing radio communication services.

Table 3 shows the code size and memory usage measured for the binary code generated using the development framework and the same application logic implemented manually. The results show a penalty for the generated code of less than 20 % in code size and less than 7 % in RAM requirements.

Table 3 Code size and the memory usage for the use case application

	ROM [bytes]	RAM [bytes]
Hand-written	17220	492
Framework-generated	20562	526

5 Conclusion

IoT rapid evolution, diversification and pervasiveness benefit from holistic approaches and improvements of the collaboration among domain experts, developers, integrators and operators of pervasive systems.

The development tools that support shared abstractions can play a primal role in IoT application modeling and implementation at all levels.

In this context, we introduced a workflow and toolset aimed to support the definition and implementation of modular IoT systems based on WSNs, enabling hybrid simulation and HiL emulation to accelerate and simplify the evaluation of system behavior in complex, large-scale scenarios.

Acknowledgments Parts of this work were supported by ARTEMIS-JU and Governments of Italy, Spain, and Greece through the ARTEMIS project WSN-DPCM (#269389).

References

1. Ashton, K.: That 'internet of things' thing. In the real world, things matter more than ideas. Expert view. RFID J. (2009). http://www.rfidjournal.com/articles/view?4986
2. Brock, D.L.: The electronic product code (EPC)—a naming scheme for physical objects. MIT Auto-ID Center White Paper (2001)
3. Bushell, S.: M-commerce key to ubiquitous internet. Expert view. Computerworld (2000). http://www.computerworld.com.au/article/84178/m-commerce_key_ubiquitous_internet/
4. Romer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wirel. Commun. **11**(6), 54–61 (2004)
5. Smith, I.G., Vermesan, O., Friess, P., Furness, A. (eds.): The internet of things 2012 new horizons. In: IERC Cluster Book, 3rd edn. Platinum, Halifax (2012)
6. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Generation Comput. Syst. **29**(7), 1645–1660 (2013)
7. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**, 393–422 (2002)
8. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
9. Chong, C.-Y., Kumar, S.P.: Sensor networks: evolution, opportunities, and challenges. Proc. IEEE **91**(8), 1247–1256 (2003)
10. Bandyopadhyay, D., Sen, J.: Internet of things: applications and challenges in technology and standardization. Wirel. Pers. Commun. **58**(1), 49–69 (2011)
11. Buttyán, L., Gessner, D., Hessler, A., Langendoerfer, P.: Application of wireless sensor networks in critical infrastructure protection: challenges and design options [security and privacy in emerging wireless networks]. IEEE Wirel. Commun. **17**(5), 44–49 (2010)

12. Durisic, M.P., Tafa, Z., Dimic, G., Milutinovic, V.: A survey of military applications of wireless sensor networks. In: 2012 Mediterranean Conference on Embedded Computing (MECO), pp. 196–199. IEEE (2012)
13. MacRuairi, R., Keane, M.T., Coleman, G.: A wireless sensor network application requirements taxonomy. In: Second International Conference on Sensor Technologies and Applications, 2008 SENSORCOMM'08, pp. 209–216. IEEE (2008)
14. Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., Razafindralambo, T.: A survey on facilities for experimental internet of things research. IEEE Commun. Mag. **49**(11), 58–67 (2011)
15. Libelium. 50 sensor applications for a smarter world: get inspired! http://www.libelium.com/top_50_iot_sensor_applications_ranking/, Aug 2013
16. Arampatzis, Th., Lygeros, J., Manesis, S.: A survey of applications of wireless sensors and wireless sensor networks. In: Proceedings of the 2005 IEEE International Symposium on Intelligent Control, Mediterrean Conference on Control and Automation, 2005, pp. 719–724. IEEE (2005)
17. Tafich, M.: The internet of things: application domains. In: Proc. of Adv. Media Technol. seminar, pp. 37–45. Technische Universität München (2013)
18. HernÃąndez-MuÃśoz, J.M., Bernat Vercher, J., MuÃśoz, L., Galache, J.A., Presser, M., HernÃąndez GÃşmez, L.A., Pettersson, J.: Smart cities at the forefront of the future internet. In: Domingue, J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., MÃijller, H., Li, M., Schaffers, H., Lotz, V., Alvarez, F., Stiller, B., Karnouskos, S., Avessta, S., Nilsson, M. (eds.) The Future Internet. Lecture Notes in Computer Science, vol. 6656, pp. 447–462. Springer, Berlin (2011)
19. Paek, J., Chintalapudi, K., Caffrey, J., Govindan, R., Sami M.: A wireless sensor network for structural health monitoring: performance and experience. The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II, pp 1–10, Sydney, Australia (2005)
20. Cheung, S.Y., Ergen, S.C., Varaiya, P.: Traffic surveillance with wireless magnetic sensors. In: Proceedings of the 12th ITS World Congress, pp. 1–13. Citeseer (2005)
21. Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys '08, pp. 323–336. ACM, New York, NY, USA (2008)
22. Burke, J.A., Reddy, S., Srivastava, M.B., Estrin, D., Hansen, M., Parker, A., Ramanathan, N.: Participatory sensing (2006)
23. Yu, L., Wang, N., Meng, X.: Real-time forest fire detection with wireless sensor networks. In: Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005, vol. 2, pp. 1214–1217. IEEE (2005)
24. Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., Welsh, M.: Deploying a wireless sensor network on an active volcano. IEEE Internet Comput. **10**(2), 18–25 (2006)
25. Cordova-Lopez, L.E., Mason, A., Cullen, J.D., Shaw, A., Al-Shamma'a, AI.: Online vehicle and atmospheric pollution monitoring using gis and wireless sensor networks. In: Journal of Physics: Conference Series, vol. 76, p. 012019. IOP Publishing (2007)
26. Trinchero, D., Galardini, A., Stefanelli, R., Fiorelli, B.: Microwave acoustic sensors as an efficient means to monitor water infrastructures. In: IEEE MTT-S International Microwave Symposium Digest, 2009. MTT'09. pp. 1169–1172. IEEE (2009)
27. Castillo-Effer, M., Quintela, D.H., Moreno, W., Jordan, R., Westhoff, W.: Wireless sensor networks for flash-flood alerting. In: Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, 2004, vol. 1, pp. 142–146. IEEE (2004)
28. Fan, Z., Kalogridis, G., Efthymiou, C., Sooriyabandara, M., Serizawa, M., McGeehan, J.: The new frontier of communications research: smart grid and smart metering. In: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, pp. 115–118. ACM (2010)
29. Nguyen, N.-H., Tran, Q.-T., Leger, J.-M., Vuong, T.-P.: A real-time control using wireless sensor network for intelligent energy management system in buildings. In: 2010 IEEE Workshop on Environmental Energy and Structural Monitoring Systems (EESMS), pp. 87–92. IEEE (2010)

30. Kahn, J.M., Katz, R.H., Pister, K.S.J.: Next century challenges: mobile networking for âĂIJsmart dustâĂİ. In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 271–278. ACM (1999)
31. Viani, F., Oliveri, G., Donelli, M., Lizzi, L., Rocca, P.,Massa, A.: Wsn-based solutions for security and surveillance. In: Microwave Conference (EuMC), 2010 European, pp. 1762–1765. IEEE (2010)
32. Tuna, G., Gulez, K., Mumcu, T.V., Gungor, V.C.: Mobile robot aided self-deploying wireless sensor networks for radiation leak detection. In: 2012 5th International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. IEEE (2012)
33. Strohbach, M., Martin, M.: Toward a platform for pervasive display applications in retail environments. IEEE Pervasive Comput. **10**(2), 19–27 (2011)
34. Ping, L., Liu, Q., Zhou, Z., Wang, H.: Agile supply chain management over the internet of things. In: 2011 International Conference on Management and Service Science (MASS), pp. 1–4 (2011)
35. Mainetti, L., Patrono, L., Vergallo, R.: Ida-pay: an innovative micro-payment system based on NFC technology for android mobile devices. In: 2012 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6 (2012)
36. Forcolin, M., Fracasso, E., Tumanischvili, F., Lupieri, P.: EuridiceâĂTiot applied to logistics using the intelligent cargo concept. In: 2011 17th International Conference on Concurrent Enterprising (ICE), pp. 1–9. IEEE (2011)
37. Anderseck, B., Hille, A., Baumgarten, S., Hemm, T., Ullmann, G., Nyhuis, P., Potthast, J.-M., Schulz, R., Monecke, J., Zadek, H., et al.: Smarti: deploying the internet of things in retail supply chains. Integration **10–11**, 2013 (2012)
38. Carullo, A., Corbellini, S., Parvis, M., Vallan, A.: A wireless sensor network for cold-chain monitoring. IEEE Trans. Instrum. Meas. **58**(5), 1405–1411 (2009)
39. Brizzi, P., Conzon, D., Khaleel, H., Tomasi, R., Pastrone, C., Spirito, M.A., Pramudianto, F.: Bringing the internet of things along the manufacturing line: a case study in controlling industrial robot and monitoring energy consumption remotely. In: IEEE International Conference on Emerging Technologies and Factory Automation—ETFA 2013. IEEE (2013a)
40. Wark, T., Peter, C., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, Philip, S., Dave, S., Bishop-Hurley, G.: Transforming agriculture through pervasive wireless sensor networks. IEEE Pervasive Comput. **6**(2), 50–57 (2007)
41. Chaudhary, D.D., Nayse, S.P., Waghmare, L.M.: Application of wireless sensor networks for greenhouse parameter control in precision agriculture. Int. J. Wirel. Mob. Netw. (IJWMN) **3**(1), 140–149 (2011)
42. Scalera A., Brizzi, P., Tomasi, R., Gregersen, T., Mertens, K., Maselyne, J., Van Nuffel, A. Hessel, E., Van den Weghe, H.: The pigwise project: a novel approach in livestock farming through synergistic performances monitoring at individual level. In: Proceeding of Conference on Sustainable Agriculture through ICT innovation—EFITA 2013, Jun 2013
43. Brizzi, P., Conzon, D., Pramudianto, F., Paralic, M., Jacobsen, M., Pastrone, C., Tomasi, R., Spirito, M.A.: Bringing the internet of things along the manufacturing line: a case study in controlling industrial robot and monitoring energy consumption remotely. In: IEEE International Conference on Emerging Technologies and Factory Automation—ETFA 2013. IEEE (2013b)
44. Wheeler, A.: Commercial applications of wireless sensor networks using zigbee. IEEE Commun. Mag. **45**(4), 70–77 (2007)
45. Hubert, T., Grijalva, S.: Realizing smart grid benefits requires energy optimization algorithms at residential level. In: Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES, pp. 1–8 (2011)
46. Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., Schreier, G.: The internet of things for ambient assisted living. In: 2010 Seventh International Conference on Information Technology: New Generations (ITNG), pp. 804–809 (2010)
47. Frederix, I.: Internet of things and radio frequency identification in care taking, facts and privacy challenges. In: Wireless VITAE 2009. 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009, pp. 319–323 (2009)

48. Caporusso, N., Lasorsa, I., Rinaldi, O., La Pietra, L.: A pervasive solution for risk awareness in the context of fall prevention. In: 3rd International Conference on Pervasive Computing Technologies for Healthcare 2009, PervasiveHealth 2009, pp. 1–8 (2009)
49. Talzi, I., Hasler, A., Gruber, S., Tschudin, C.: PermaSense: investigating permafrost with a WSN in the Swiss Alps. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets '07, pp. 8–12. ACM, New York (2007)
50. Szlavecz, K, Terzis, A., Ozer, S., Musaloiu-Elefteri, R., Cogan, J., Small, S., Burns, R.C., Gray, J., Szalay, A.S.: Life Under Your Feet: An End-to-End Soil Ecology Sensor Network, Database, Web Server, and Analysis Service. CoRR, abs/cs/0701170 (2007)
51. Wireless Sensor Network Development, Planning, Commissioning, and Maintenance (WSN-DPCM). http://www.wsn-dpcm.eu/. Accessed Oct 2011
52. Hui, J.W.: An extended internet architecture for low-power wireless networks—design and implementation. PhD thesis, EECS Department, University of California, Berkeley(2008)
53. Sugihara, R., Gupta, R.K.: Programming models for sensor networks: a survey. ACM Trans. Sen. Netw. **4**(2), 8:1–8:29 (2008)
54. Mottola, L., Picco, G.P.: Programming wireless sensor networks: fundamental concepts and state of the art. ACM Comput. Surv. **43**(3), 19:1–19:51 (2011)
55. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. SIGPLAN Not. **35**(11), 93–104 (2000)
56. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesc language: aholistic approach to networked embedded systems. SIGPLAN Not. **38**(5), 1–11 (2003)
57. Cheong, E., Liebman, J., Liu, J., Zhao, F.: TinyGALS: a programming model for event-driven embedded systems. In: Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03, pp. 698–704. ACM, New York (2003)
58. Han, C.-C., Kumar, R., Shea, R., Kohler, E., Srivastava, M.: A dynamic operating system for sensor nodes. In: Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, MobiSys '05, pp. 163–176. ACM, New York (2005)
59. Han, C.-C., Goraczko, M., Helander, J., Liu, J., Priyantha, B., Zhao, F.: CoMOS: An Operating System for Heterogeneous Multi-Processor Sensor Devices. Technical Report MSR-TR-2006-177, Microsoft Research (2006)
60. Greenstein, B., Kohler, E., Estrin, D.: A sensor network application construction kit (SNACK). In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pp. 69–80. ACM, New York (2004)
61. Levis, P., Gay, D., Handziski, V., Hauer, J.H., Greenstein, B., Turon, M., Hui, J., Klues, K., Sharp, C., Szewczyk, R., et al.: T2: a second generation os for embedded sensor networks, Telecommunication Networks Group, Technische Universität Berlin. Technical, Report TKN-05-007, (2005)
62. Kasten, O., Romer, K.: Beyond event handlers: programming wireless sensors with attributed state machines. In: Information Processing in Sensor Networks 2005, pp. 45–52 (2005)
63. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation, vol. 1, NSDI'04, pp. 3–3. USENIX Association, Berkeley (2004)
64. Bhatti, S., Carlson, J., Dai, H., Deng, J., Rose, J., Sheth, A., Shucker, B., Gruenwald, C., Torgerson, A., Han, R.: Mantis os: an embedded multithreaded operating system for wireless micro sensor platforms. Mob. Netw. Appl. **10**(4), 563–579 (2005)
65. McCartney, W.P., Sridhar, N.: Abstractions for safe concurrent programming in networked embedded systems. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, pp. 167–180. ACM, New York (2006)
66. Dunkels, A., Gronvall, B., Voigt, T.: Contiki—a lightweight and flexible operating system for tiny networked sensors. Local Comput. Netw. **2004**, 455–462 (2004)
67. Nitta, C., Pandey, R., Ramin, Y.: Y-Threads: supporting concurrency in wireless sensor networks. In: Proceedings of the Second IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS'06, pp. 169–184. Springer-Verlag, Berlin (2006)

68. Levis, P., Culler, D.: Maté: a tiny virtual machine for sensor networks. SIGOPS Oper. Syst. Rev. **36**(5), 85–95 (2002)
69. Levis, P., Gay, D., Culler, D.: Active sensor networks. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation, vol. 2, NSDI'05, pp. 343–356. USENIX Association, Berkeley (2005)
70. Yu, Y., Rittle, L.J., Bhandari, V., LeBrun, J.B.: Supporting concurrent applications in wireless sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, pp. 139–152. ACM, New York (2006)
71. Koshy, J., Pandey, R.: Vmstar: synthesizing scalable runtime environments for sensor networks. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, SenSys '05, pp. 243–254. ACM, New York (2005)
72. Gu, L., Stankovic, J.A.: T-kernel: providing reliable os support to wireless sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, pp. 1–14. ACM, New York (2006)
73. Kothari, N., Gummadi, R., Millstein, T., Govindan, R.: Reliable and efficient programming abstractions for wireless sensor networks. SIGPLAN Not. **42**(6), 200–210 (2007)
74. Newton, R., Arvind, Welsh, M.: Building up to macroprogramming: an intermediate language for sensor networks. In: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN '05, Piscataway, IEEE Press, NJ (2005)
75. Newton, R., Morrisett, G., Welsh, M.: The regiment macroprogramming system. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN '07, pp. 489–498. ACM, New York (2007)
76. Chu, D., Popa, L., Tavakoli, A., Hellerstein, J.M., Levis, P., Shenker, S., Stoica, I.: The design and implementation of a declarative sensor network system. In: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, SenSys '07, pp. 175–188. ACM, New York (2007)
77. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1), 122–173 (2005)
78. Mottola, L., Picco, G.P. : Programming wireless sensor networks with logical neighborhoods. In: Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks, InterSense '06, ACM, New York (2006)
79. Bakshi, A., Prasanna, V.K., Reich, J., Larner, D.: The abstract task graph: a methodology for architecture-independent programming of networked sensor systems. In: Proceedings of the 2005 Workshop on End-to-End, Sense-and-Respond Systems, Applications and Services, EESR '05, pp. 19–24. USENIX Association, Berkeley (2005)
80. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. **36**(SI), 131–146 (2002)
81. Marron, P.J., Karnouskos, S., Minder, D., The CONET Consortium.: Roadmap on Cooperating Objects. Kluwer Academic Publishers, Luxembourg (2009)
82. Song, Z.Y., Lavagno, L., Tomasi, R., Spirito, M.A.: A service-driven development tool for wireless sensor network. In: C. Benavente-Peces, F.H. Ali, J. Filipe (eds.) PECCS, pp. 87–95. SciTePress, Rome, Italy (2012)
83. IBM. Restful web services. http://www.ibm.com/developerworks/webservices/library/ws-restful/
84. Eclipse wsdl editor. http://wiki.eclipse.org/index.php/Introduction_to_the_WSDL_Editor. Accessed 8 Feb 2013
85. Liquid. http://www.liquid-technologies.com/WSDL-Editor.aspx. Accessed 10 Feb 2013

6TiSCH Wireless Industrial Networks: Determinism Meets IPv6

Maria Rita Palattella, Pascal Thubert, Xavier Vilajosana, Thomas Watteyne, Qin Wang and Thomas Engel

Abstract In the last 40 years we have witnessed the emergence of the Operational Technology (OT) and the Information Technology (IT) in parallel, each established with its own scope and range of applications; then, the progressive convergence of IT over an IP infrastructure, imprinted by the birth of the Information and Communications Technology (ICT). Nowadays, we are witnessing the unstoppable evolution of the Internet of Things (IoT), and the upcoming integration of OT and IT. The technologies facilitating such OT/IT convergence only recently commenced to take shape. In detail, existing industrial Wireless Sensor Network technologies have demonstrated that the IEEE802.15.4e Timeslotted Channel Hopping (TSCH) effectively enables industrial-grade deterministic properties for control loops with low latency, ultra-low jitter, ultra-low power consumption and a high reliability. This chapter introduces the work recently started at the IETF by the 6TiSCH working group, and which aims at enabling IPv6 over the TSCH mode of the IEEE802.15.4e standard. In particular,

M. R. Palattella (✉) · T. Engel
University of Luxembourg, Luxembourg, Luxembourg
e-mail: maria-rita.palattella@uni.lu

T. Engel
e-mail: thomas.engel@uni.lu

P. Thubert
Cisco Systems, Issy Les Moulineaux, France
e-mail: pthubert@cisco.com

X. Vilajosana
Universitat Oberta de Catalunya, Barcelona, Catalonia, Spain
e-mail: xvilajosana@uoc.edu; xvilajosana@eecs.berkeley.edu

T. Watteyne
Linear Technology/Dust Networks Product Group, Hayward, CA, USA
e-mail: watteyne@eecs.berkeley.edu; twatteyne@linear.com

Q. Wang
University Science and Technology Beijing, Beijing, China
e-mail: wangqin@ies.ustb.edu.cn; qinwang@berkeley.edu

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_5,

6TiSCH standardizes different mechanisms for allocating link-layer resources and trade off latency and bandwidth with power consumption. Several approaches are supported, based on a combination of centralized and distributed techniques.

1 Introduction

Operational Technology (OT) refers to industrial networks that are typically used for monitoring systems and supporting control loops, as well as movement detection systems for use in Process Control (i.e., process manufacturing) and Factory Automation (i.e., discrete manufacturing).

For the last 40 years or more, industrial networks were developed in parallel to the Public Switched Telephone Network (PSTN), and to early data networks such as Bitnet, which was based on IBM's Systems Network Architecture (SNA). For the last 25 years, the Internet Protocol (IP) [1] has become the *de-facto* standard for the networking layer of the Information Technology (IT) and for the Internet at large.

For the last 15 years, Voice over IP (VoIP) and the emergence of Internet protocols such as the Media Gateway Control Protocol (MGCP) [2] and the Session Initiation Protocol (SIP) [3] enabled a progressive convergence of voice and data networking technologies, over IP. The Real-Time Transport Protocol (RTP) and the Real-Time Transport Control Protocol (RTCP) [4] enabled video streaming and conferencing, leading to the convergence of voice, data, and video over the IP infrastructure.

Information and Communications Technology (ICT) extends the term IT to refer to that convergence, including the software and hardware pieces that enable the manipulation, access, transport and persistence of information over an IP network, be it for voice, data, or video applications.

The next convergence step is the integration of IT and OT technologies, enabling OT traffic to be transported over a shared IT, IP-based, infrastructure. This integration presents a number of new challenges. Due to its different goals, OT has evolved in a manner that is radically different from IT, focusing on highly secure, reliable and deterministic networks, with limited scalability over a bounded area.

Traffic flows such as control loops and motion detection systems are deterministic in that their communication patterns are known a priori. Routing paths and communication schedules can be computed in advance, in a fashion similar to a railway system, to avoid losses due to packet collisions, and to perform global optimizations across multiple flows. Based on that knowledge, a *deterministic network* allocates the required resources (buffers, processors, medium access) along the multi-hop routing path at the precise moment the resources are needed. The forwarding elements can handle data with an amount of jitter that can be negligible for a particular application.

This model is different from the IP Quality of Service (QoS) model, which relies of selective queuing and discarding of packets to achieve end-to-end flow control on statistically multiplexed traffic flows. With strict ingress shaping and resource over-provisioning, QoS can reduce the congestion loss and induced jitter, but never eliminate them completely.

OT is moving gradually towards Ethernet and IP in order to reuse Commercial Off-The-Shelf (COTS) products and reduce cost. In that case, IP technology is only used as yet another fieldbus, with still a clear physical separation between the IT and OT networks and no IP routing between the two domains. End-to-end communication between applications and field devices over IP are (i) mostly a green field (ii) a long term investment for a new factory and (iii) expected to scale to large numbers of devices, possibly tens of thousands in a large plant. It results that IP version 6 (IPv6) [5] is the de-facto IP standard version for the IT/OT convergence. A legacy IPv4 domain can still be reached using Network Address Translation between IPv6 an IPv4 (NAT64) [6] techniques.

To achieve a real convergence, OT needs not only to adopt the formats of IPv6 packets, but also to share the use of the Internet Protocol suite over the common fabric. The required protocol suite for OT includes at a minimum the Internet Control Message Protocol (ICMPv6) [7], the IPv6 Neighbor Discovery Protocol (NDP) [8], the Routing Protocol for Low-Power and Lossy Networks (RPL) [9], and possibly protocols such as the Dynamic Host Configuration Protocol (DHCPv6) [10]and the Multicast Listener Discovery (MLD) protocols [11], as well upper layer protocols such as the User Datagram Protocol (UDP) [5, 12] and the Constrained Application Protocol (CoAP) [13].

Taken separately, those protocols offer many options. The task of selecting and including them all in an overall Machine-to-Machine (M2M) standard can be cumbersome for an industrial Standards Developing Organization (SDO) such as the HART Communication Foundation (HCF), the International Society of Automation (ISA), or the International Electrotechnical Commission (IEC).

It makes sense for the Internet Engineering Task Force (IETF), the SDO that defines the IP protocol suite, to participate in this integration effort, and to propose a simplified bundled suite for M2M that is less cumbersome to include.

The bundled suite needs to provide evolved IETF protocols that match OT requirements and constraints, exploiting the latest technologies from the IEEE for deterministic Media Access Control (MAC), and best practices for (i) network virtualization to achieve strict flow isolation, (ii) high availability to ensure continuous operation, and (iii) security to enable trusted local and remote access [14].

To enable IT and OT technologies to converge over a shared network fabric, there is also a need to adapt the Internet Protocol suite to match the constraints and requirements of automation loops, but also provide components to enable IPv6 operations over medium access technologies such as deterministic Ethernet and IEEE802.15.4e TSCH. Moreover, there is a need to provide an architecture that ties this adapted protocol suite together in order to simplify the adoption of the bundle.

A new Working Group called 6TiSCH [15, 16] is being created at the IETF to enable IPv6 over the TSCH mode of the IEEE802.15.4e standard [17]. The WG considers an architecture in which low-power wireless devices (often called "motes") form a multi-hop Low-power Lossy Network (LLN). This LLN connects to the traditional Internet through one or more LLN Border Routers (LBRs) [18].

At the heart of IEEE802.15.4e TSCH is the notion that the nodes in the LLN communicate by following a schedule. A Time Division Multiple Access (TDMA)

structure instructs each mote what to do in each slot: in other words, if it has to be active and transmit, or receive; or be inactive and thus, sleep. The way this schedule is built determines the amount of traffic that the LLN can produce, the latency of a packet and the redundancy of the network. It also determines the amount of energy each node consumes, hence the lifetime of the network. The goal of 6TiSCH is to develop a standard approach to manage this schedule and match it against the traffic needs in the network [19].

Three different modes are considered for building and maintaining this schedule:

- In the *minimal* case, the schedule is static, either preconfigured, or learnt by a node when joining the network. While it does not exploit the full benefits of IEEE802.15.4e TSCH, it can be used as a "fall-back" mode.
- In the *centralized* case, a specific schedule entity called Path Computation Element (PCE) is located onto the Internet, and continuously gathers network state information and traffic requirements from the motes in the network, adjusting the TSCH schedule accordingly.
- In the *distributed* case, motes in the LLN agree on a schedule by using distributed multi-hop scheduling protocols and neighbor-to-neighbor scheduling negotiation.

The remainder of this chapter is organized as follows. Section 2 introduces the newly published IEEE802.15.4e MAC standard, in particular its "Timeslotted Channel Hopping" (TSCH) mode. After presenting the context in which this standard was developed, Sect. 2 introduces the fundamental concepts of slotframes, time synchronization, channel hopping and schedule. Section 3 presents the architecture envisioned by 6TiSCH. After identifying the scope, Sect. 3 presents the protocol stack, and in particular the 6top sublayer responsible for managing the TSCH schedule. It also covers the different scheduling modes and forwarding mechanisms. Section 4 presents the range of applications enabled by 6TiSCH. Finally, Sect. 5 concludes this chapter by presenting the on-going work and future milestones in the 6TiSCH group.

2 IEEE802.15.4e Timeslotted Channel Hopping

The IEEE802.15 Task Group 4e (TG4e) was created in 2008 to *redesign* the existing IEEE802.15.4-2006 Medium Access Control (MAC) standard to obtain a low-power multi-hop MAC better suitable to emerging needs of embedded industrial applications. The final goal of the TG4e was to overcome the two main drawbacks of the IEEE802.15.4 MAC, consisting in (i) the high energy consumption of router nodes which require their radio to be always on, regardless of their actual traffic; and (ii) the high level of interference and fading, due to the fact that all communication happens on a single frequency.

The IEEE802.15.4e standard [17] has been published in 2012 as an amendment of the IEEE802.15.4-2011 MAC protocol [20]. Three different MACs have been proposed by the 15.4e TG:

- Low Latency Deterministic Network (LLDN)
- Timeslotted Channel Hopping (TSCH)
- Deterministic and Synchronous Multi-channel Extension (DSME)

The Timeslotted Channel Hopping (TSCH) mode facilitates multi-hop operation and deals well with fading and interference. The TSCH mode of the IEEE802.15.4e standard has been the focus of the activities of the 6TiSCH group, and the object of this chapter.

At the core of the TSCH is a medium access technique which uses time synchronization to achieve ultra low-power operation and channel hopping to enable high reliability. This is very different from the "legacy" IEEE802.15.4 MAC protocol.

IEEE802.15.4e does not amend the physical layer. That is, it can operate on any IEEE802.15.4-compliant hardware. The IEEE802.15.4 PHY is arguably the standard with the longest-standing impact in low-power wireless mesh technology, and has been widely used by low-power battery-powered devices to build Low Power and Lossy Networks (LLN).

The need to interconnect IEEE802.15.4-based low power networks to the Internet has triggered the birth of various working groups (WGs) within the IETF, including 6LoWPAN [21], ROLL (the group behind the RPL routing protocol [9]), and CORE (behind the CoAP web transfer protocol [13]) that have defined how to fit an IPv6 protocol stack on top of IEEE802.15.4. Given the appealing features of the IEEE802.15.4e for enabling ultra-low power and reliable LLNs, the 6TiSCH WG aims at building IPv6-enabled LLNs, rooted in the IEEE802.15.4e TSCH MAC layer.

The basic concept of TSCH (i.e. the combination of time synchronization and channel hopping) is not new. It was introduced by Dust Networks in 2006 in its proprietary Time Synchronized Mesh Protocol (TSMP) [22]. The core ideas of TSMP then made it into standards such as WirelessHART (2007) [23] and ISA100.11a (2009) [24, 25]. These standards have targeted the industrial market, which requires ultra-high reliability and ultra-low power. WirelessHART is for instance the wireless extension of HART, a long standing protocol suite for networking industrial equipment.

IEEE802.15.4e TSCH inherits directly from these industrial standards, which are already deployed as commercial products in ten of thousands of networks in operation today. TSCH is thus a proven technology. One important difference with existing industrial standards is that IEEE802.15.4e TSCH focuses exclusively on the MAC layer. This clean layering allows for TSCH to fit under an IPv6-enabled "upper" protocol stack.

2.1 Slotframe and Time Synchronization

All motes in a TSCH multi-hop network are synchronized. Time is sliced up into timeslots. A timeslot is long enough for a MAC frame of maximum size to be sent from mote `A` to mote `B`, and for mote `B` to reply with an acknowledgement (ACK)

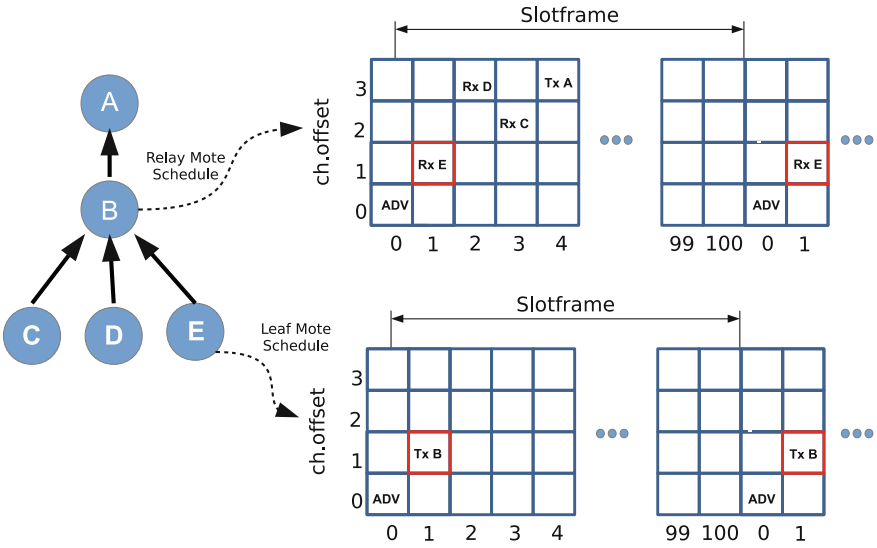


Fig. 1 Slotframe representation for a set of motes. The schedule of mote `B` is configured so it can relay information of the leaf nodes. Leaf node E is configured to be able to send a packet once every slotframe. Empty slots are "sleep" slots. The slotframe is composed of 101 slots of 15 ms each

frame indicating successful reception. The duration of a timeslot is not imposed by the IEEE802.15.4e standard, and can be tuned to fit the application's needs.[1]

TSCH defines a timeslot counter called Absolute Slot Number (ASN). When a new network is created, the ASN is initialized to 0; from then on, it increments by 1 at each timeslot. A mote learns the current ASN when it joins the network. Since motes are synchronized, they all know the current value of the ASN, and any time. The ASN is encoded as a 5-byte number: this allows it to increment for hundreds of years (the exact value depends on the duration of a timeslot) without wrapping. The ASN is used to calculate the frequency to communicate on, jointly with the `channelOffset`, as described in Sect. 2.3.

As shown in Fig. 1, timeslots are grouped into one or more slotframes. A slotframe continuously repeats over time. The IEEE802.15.4e TSCH standard does not impose any slotframe size. Depending on the application needs, these can range from 10 s to 1000 s of timeslots. The shorter the slotframe, the more often a timeslot repeats, resulting in more available bandwidth and lower latency, but also in higher power consumption.

Because of the slotted nature of communication in a TSCH network, motes have to maintain tight synchronization. All motes are equipped with clocks to keep track of time. Because clocks in different motes drift with respect to one another, neighbor

[1] With IEEE802.15.4-compliant radios operating in the 2.4 GHz frequency band, a maximum-length frame of 127 bytes takes about 4 ms to transmit; a shorter ACK takes about 1 ms. With a 10 ms slot (a typical duration), this leaves 5 ms to radio turnaround, packet processing and security operations.

motes need to periodically re-synchronize. Each mote periodically synchronizes its network clock to at least one other mote, and it also provides its network time to its neighbors. IEEE802.15.4e does not define how a mote chooses its "time source neighbor"; this is left to the upper layer, which needs to ensure that the synchronization structure is loop-free.

Nodes already in the network periodically send Enhanced Beacons (EBs) to announce the presence of the network. When a new mote boots, it listens for those to synchronize to the TSCH network. EB frames contain information about the timeslot length, the slotframes and timeslots the beaconing mote is listening on, and a 1-byte join priority (i.e., number of hops separating the node sending the EB, and the PAN coordinator).

IEEE802.15.4e TSCH implicitly adds timing information to all packets that are exchanged. Motes are required to transmit data packets at precise time in the time slot, and timestamp the instant of arrival of received data frames. A node (re-)synchronizes to its time source neighbor using this timing information. IEEE802.15.4e defines two methods for a device to synchronize to its time source neighbor: "acknowledgement-based" and "frame-based" synchronization.

In both cases, the receiver calculates the difference in time between the expected and actual time of frame arrival. In acknowledgement-based synchronization, the receiver writes that measured offset in a field in the acknowledgement frame it sends to the sender. This allows the sender mote to synchronize to the clock of the receiver. In frame-based synchronization, the receiver uses the measured time offset to adjust its own clock. In this case, it is the receiver mote which synchronizes to the clock of the sender.

Regardless the synchronization method adopted, data traffic is used to implictly re-synchronize a node with its time source neighbor. In the absence of data traffic, motes are still required to periodically synchronize to their time source neighbor(s) to account for clock drift. If they have not been communicating for some time, motes can exchange an empty data frame, often referred to as a *Keep-alive* message, to re-synchronize. The frequency at which such messages need to be transmitted depends on the stability of the clock source, and the "guard time" duration (i.e., how "early" each mote starts listening for data). With a 10 ppm clock source and a 1 ms guard time, this period can be up to 100 s.

Different synchronization policies are possible in a TSCH network. Motes can keep synchronization exclusively by (i) exchanging EBs, or (ii) periodically sending valid frames to time source neighbors, or (iii) by using a combination of both. Which solution to use depends on network requirements and operational conditions. For instance, temperature variations might introduce important variations on clock alignment and might require adaptive techniques to maintain synchronization without impacting energy consumption [26]. In any case, the cost of synchronization in a TSCH network is negligible in most applications.

2.2 TSCH Schedule

Timeslotted Channel Hopping is different from traditional low-power MAC protocols because of its scheduled nature. In a TSCH network, each mote follows a **schedule**. This schedule looks like a matrix of width equal to the slotframe size, and of height equal to the number of available frequencies (as shown in Fig. 1).

A single element in the TSCH schedule, identified by a `slotOffset`, and a `channelOffset`, is called a **cell** [27]. A cell can be considered as an atomic unit to be allocated by a scheduling algorithm. Because of the channel hopping nature of TSCH (see Sect. 2.3), the scheduling algorithm does not need to worry about the actual frequency communication happens on, since it changes at each slotframe iteration.

A TSCH schedule instructs each mote what to do in each timeslot: transmit, receive or sleep. A cell can therefore be scheduled or unscheduled. During an unscheduled cell, the node does not communicate. When a cell is scheduled, it is assigned a MAC-layer slotframe identifier, a neighbor MAC address (which can be the broadcast address[2]), and one or more of the following flags: `TX`, `RX`, `shared`, `timeskeeping`, `hard` (see Sect. 3.1.1 for details).

The TSCH schedule contains all the scheduled cells from all the slotframes and is sufficient to qualify the communication in the TSCH network. Once a mote obtains its schedule, it executes it:

- For each `TX` cell, the mote checks whether there is a packet in the outgoing queue which matches the neighbor written in the schedule information for that timeslot. If there is none, the mote keeps its radio off for the duration of the timeslot. If there is one, the mote can ask for the neighbor to acknowledge it, in which case it has to listen for the acknowledgement after transmitting.
- For each `RX` cell, the mote listens for possible incoming packets. If none is received after some listening period, it shuts down its radio. If a packet is received, addressed to the mote, and passes security checks, the mote can send back an acknowledgement, if requested.

Assuming the schedule is well built, if mote `E` is scheduled to transmit to mote `B` at `slotOffset` 1 and `channelOffset` 1, mote `B` will be scheduled to receive from mote `E` at the same `slotOffset` and `channelOffset`, as depicted in Fig. 1.

If there is a lot of data flowing from mote `E` to mote `B`, the schedule might contain multiple cells from `E` to `B` in the same slotframe. Multiple cells scheduled to the same neighbor are typically equivalent, i.e., the MAC layer sends the packet on whichever of these cells happens to show up first after the packet was put in the MAC queue. The union of all these cells identified by different `[slotOffset, channelOffset]`, which are scheduled for a same purpose, with the same neighbor (e.g., between two neighbors, `E` and `B`), with the same flags, and the same slotframe, is called a **bundle** [27]. Since the slotframe repeats over time, each cell gives

[2] A broadcast cell is an alias for "a scheduled cell with neighbor address the broadcast address".

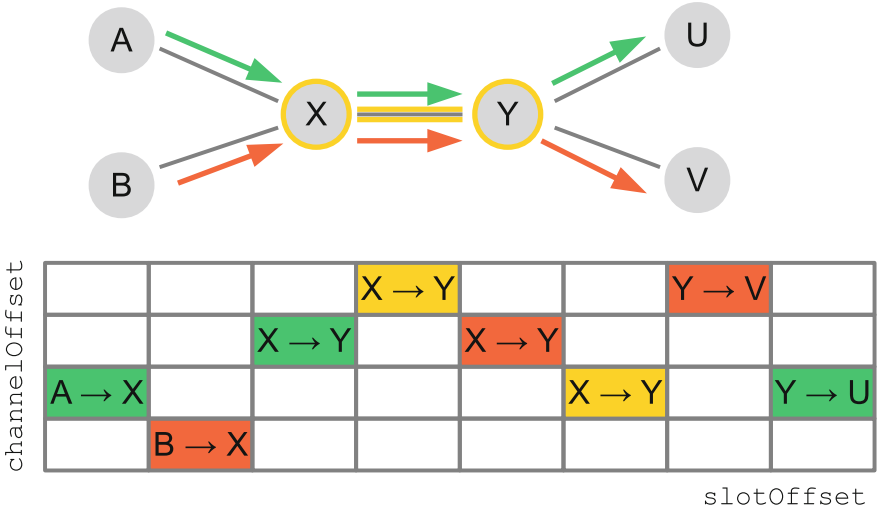


Fig. 2 Example of TSCH schedule including a bundle (in *yellow*), and two distinct tracks (in *green*, and *orange*, respectively) for a simple network scenario

a "quantum" of bandwidth to a given neighbor. Modifying the number of cells in a bundle modifies the amount of resources allocated between two neighbors. An example of bundle is shown in Fig. 2.

By default, each scheduled `TX` cell within the TSCH schedule is dedicated, i.e., reserved only for mote `E` to transmit to mote `B`. IEEE802.15.4e allows also to mark a cell as shared. In a shared cell, multiple motes can transmit at the same time, on the same frequency. To avoid contention, TSCH defines a back-off algorithm for shared cells.

A scheduled cell can be marked as both transmit (`TX`) and receive (`RX`). In this case, a mote transmits if it has an appropriate packet in its output buffer, listens otherwise. Marking a cell as `[TX,shared,RX]` results in slotted-Aloha behavior.

A sequence of cells scheduled along a multi-hop path is called a **track** [27] (see Fig. 2). It is the result of a reservation, and it belongs to the node that initializes the process for establishing the track itself.

The schedule defines entirely the synchronization and communication between motes. By adding/removing cells between neighbors, one can adapt a schedule to the needs of the specific application. It is possible to:

- Make the schedule "sparse" for applications where motes need to consume as little energy as possible, at the price of reduced bandwidth.
- Make the schedule "dense" for applications where motes generate a lot of data, at the price of increased power consumption.
- Add more cells along a multi-hop route over which many packets flow.

TSCH defines the mechanisms to execute a communication schedule. Yet, how the schedule is built, updated and maintained, and and by which entity, is outside of the scope of the IEEE802.15.4e standard. Several scheduling approaches have been investigated by the research community.

A centralized Traffic Aware Scheduling Algorithm (TASA) has been proposed recently [28]. It exploits matching and coloring procedures to schedule cells and tracks to all the motes across the entire network topology graph. In detail, it allows to set up the TSCH schedule based on the network topology and the traffic load. Thus, it uses the information related to the paths, coming from the routing protocol (e.g., RPL), and those related to the traffic (e.g., average traffic load generated by each node) in order to schedule cells and tracks, and provide the required level of QoS (duty cycle, throughput, etc.) to each flow.

Decentralized approaches have been studied in the context of the OpenWSN project [29]. uRes [30] proposes a bargaining based approach where nodes negotiate with their neighbors to allocate cells. uRes allocates cells minimizing the number of collisions as nodes have a certain knowledge of their neighbors schedule. Collisions can still occur; they are resolved by re-allocating the colliding cells [31].

A different decentralized approach has been published recently by Morell et al. [32]. The article introduces the concept of label switching in TSCH networks and proposes the use of reservation to establish and manage tracks between nodes in the network. This approach computes the schedule of the network by collecting information along the track and installing it during the downstream reservation message, in a way similar to the RSVP standard [31].

The aforementioned protocols [28, 30, 32] can be seen as candidate scheduling solutions within the 6TiSCH WG.

2.3 Channel Hopping

The TSCH mode of IEEE802.15.4e combines time synchronization and channel hopping. It is thereby able to provide ultra-high reliability and ultra-low power. For each scheduled cell, the schedule defines a `slotOffset` and a `channelOffset`. The latter is translated by both communicating motes into a frequency using (1).

$$frequency = F(channelOffset + ASN) MOD nFreq \tag{1}$$

The function `F` consists of a look-up table containing the set of available channels. The size of this look-up table is equal to the number of available frequencies, *nFreq*. There are as many `channelOffset` values as there are frequencies available (e.g., 16 when using IEEE802.15.4-compliant radios at 2.4 GHz, when all channels are used). Since both motes have the same `channelOffset` written in their schedule for that scheduled cell, and the same ASN counter since they are synchronized, they compute the same frequency. At the next iteration (cycle) of the slotframe, however,

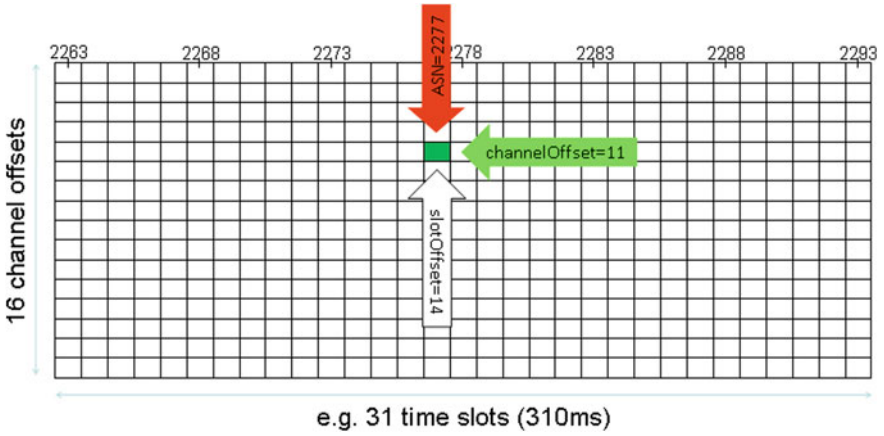


Fig. 3 Channel hopping happens even when the schedule does not change

the `channelOffset` is the same, but the ASN will have changed, resulting in the computation of a different frequency.

Figure 3 illustrates this behavior. It highlights a single cell in a 31-slot long slotframe. This cell is at `slotOffset` 14, at `channelOffset` 11 and at ASN 2277 (at this iteration of the slotframe). When communicating, the neighbor nodes using this cell will apply (1) to obtain a certain frequency. At the next iteration of the slotframe, even is the cell is still at the same `slotOffset` and `channelOffset`, the ASN will have changed and applying (1) will result in a different frequency calculation.

The channel hopping nature of TSCH causes links to be very stable. Wireless phenomena such as multi-path fading and external interference impact a wireless link between two motes differently on each frequency. If a transmission between two mote fails, retransmitting on a different frequency has a higher likelihood of succeeding that retransmitting on the same frequency. As a result, even when some frequencies are "misbehaving", channel hopping averages the contribution of each frequency, resulting in more stable links, and therefore a more stable topology.

2.4 TSCH Deterministic Networks

The IEEE802.15.4e TSCH, due to the combination of Time Division Multiplexing (TDM), time synchronization and the time formatted into slotframes, results in a *deterministic* wireless MAC standard, suitable for deterministic traffic, i.e., traffic flows with an emission rate and routing path patterns that are well-known in advance. For such traffic, a deterministic network allocates the required resources (buffers, processors, medium access) along the multi-hop routing path at the precise moment

the resources are needed. In a TSCH network, the bandwidth is pre-formatted in a TDM fashion. Thus, unlike the traditional CSMA/CA-based networks, there is no contention for gaining access to the channel. In fact, a time slot becomes a unit of throughput that can be allocated to a given deterministic flow. Deterministic networks are also adapted by *Traffic Engineering*, i.e., to support several isolated flows. Different optimized paths and tracks can be built for all the isolated flows, based on their specific requirements.

A deterministic network guarantees timely transmission. A good example of a deterministic network is a railway system, where trains are scheduled periodically to leave a railway station at a certain time, to traverse the stations along a predetermined track at precise times so that, in the end, a given train arrives at its final station at the expected time, with virtually no jitter from a human perspective. Moreover, collisions are eliminated: there is never another train blocking the rail and delaying this train.

The IEEE802.15.4e TSCH, with its deterministic behavior, opens up a set of new applications which require high reliability and low-power. It enables control loops in wireless process control/automation networks, where wired solutions have been used so far. Supporting Traffic Engineering and isolated traffic flows, it enables *umbrella* networks, transporting data from different independent clients. Each flow is isolated and shaped, according to the requirements specified by the client. With its low-power nature and predictable power consumption, TSCH enables networks of energy harvesting motes. Finally, it supports widespread monitoring (like corrosion monitoring or pipe leak detection), which requires slow periodic reporting rates from a large number of sensors and open loop operation. More details about potential 6TiSCH applications are provided in Sect. 4.

These new applications will make a lot more sense if they are able to reuse the same building blocks and thus share a same converged network for IT/OT, based on an IPv6-architecture.

3 The 6TiSCH Architecture

In a large extensive factory floor, hundreds to as many as tens of thousands of constrained field devices might be deployed as a single LLN and share a same physical environment. The logical structure of the radio meshes vary as the conditions change, even in the absence of physical movement.

The memory and processing resources of a constrained device such as a battery-operated sensors or actuators are drastically limited, so all states, including those related to addressing and routing, must be kept to a minimum.

For an IPv6-enabled device, it is commonplace to save an extra identifier and use a (the) IPv6 address of the device as a unique identifier. This implies that the IPv6 address is permanent—the device cannot be renumbered—as long as it keeps playing the role that corresponds to that device identifier.

It results that the device retains its prefix information quasi-permanently, and thus can only be reachable over IP within the range of the network where the IPv6 subnet

associated to that prefix is defined. In order to allow mobility and unhindered reorganization of the routing topology into multiple small mesh networks, it is necessary that the subnet spans the whole factory floor. For a large factory, this means that potentially thousands of field devices will form a single subnet, sharing an IPv6 prefix from which all their global (or unique-local) IPv6 addresses are derived.

A possible model to scale a Low-power Lossy Network (LLN) to the thousands as a single IPv6 subnet consists in laying out a high-speed backbone that spans the area and provides a fast transit facility to federate the network. A mesh protocol partitions the LLN in a collection of logical graphs such as Destination Oriented Directed Acyclic Graphs (DODAGs) oriented towards the backbone. IPv6 routing or ND proxy operation over the backbone enable the overall connectivity effectively forming a so-called multilink subnet.

In existing deployments, the high-speed backbone may effectively be an Ethernet Switched fabric, or a wireless mesh network based on IEEE802.11 technology. A Mesh Access Point connects a specific Gateway that terminates IP flows towards the backbone. The Gateway connects to the field devices over a LLN mesh that is a based on a variation of the TSCH technology that guarantees deterministic transmissions, and may either use IPv6 as a Network Layer or completely bypass that layer.

The natural evolution of that model is to implement the end-to-end principle that sustains the Internet, based in the Internet Protocol. An application agnostic (Layer 3) Router physically replaces the Gateway, so that the end-to-end flows from a field device are no more constrained to terminate at that point, but may flow over a converged IT/OT infrastructure to terminate in servers that are located in the carpeted floor.

6TiSCH will document that evolved model as an open standards-based architecture, highlight best practices, and standardize the missing components to achieve industrial-grade performance in terms of jitter, latency, scalability, reliability and low-power operation for IPv6 over IEEE802.15.4e TSCH [17]. Although not addressed directly by 6TiSCH, it is envisioned that the resulting techniques will be applicable to technologies other than 2.4 GHz IEEE802.15.4 [20].

As illustrated in Fig. 4, the scope of the architecture is a potentially large IPv6 multilink subnet that may span thousands of LLN devices, federated over a backbone, as discussed above.

The architecture will address how multiple BBRs are supported for a higher degree of scalability and reliability, and how nodes maintain synchronization in the presence of multiple BBR[3] [33].

When possible, the architecture will reuse existing protocols such as IPv6 Neighbor Discovery (ND) [8], IPv6 Low power Wireless Personal Area Networks (6LoWPAN) [21], and the Routing Protocol for Low Power and Lossy Networks (RPL) [9], with the minimum adaptation required to meet criteria for reliability and determinism within the mesh, and scalability over the backbone.

[3] This work implies new IPv6 ND proxy operations as illustrated in Sect. 3.6.

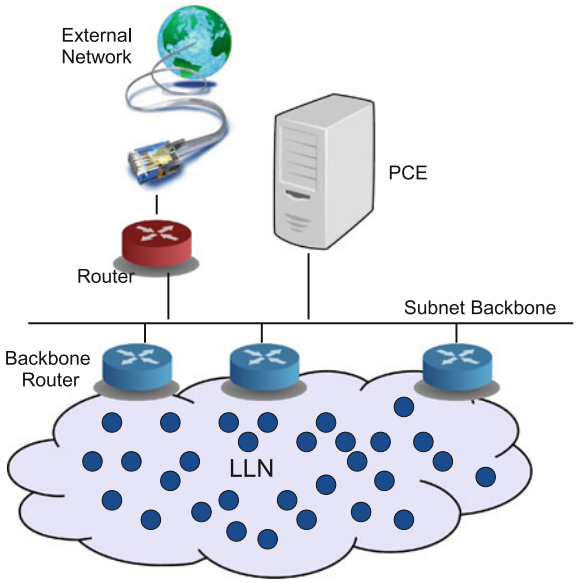


Fig. 4 6TiSCH reference architecture

3.1 The 6TiSCH Stack

The 6TiSCH working group aims at filling the gaps between IEEE lower layers of the protocol stack and the IETF higher layers, to enable an open standards-based protocol stack for deterministic wireless mesh networks. Figure 5 presents a general overview of the integrated protocol stack rooted at the IEEE802.15.4 physical layer and operated by the IEEE802.15.4e TSCH MAC amendment that provides determinism while ensures very low power operation. As described earlier, missing gaps need to be filled by 6TiSCH so that IETF 6LoWPAN Header Compression and RPL, which enables IPv6 packetization and routing, can optimally operate on top of the TSCH MAC layer.

The deterministic nature of IEEE802.15.4e TSCH and its strict requirements in terms of schedule management make it necessary to have a sublayer which enables management entities (MEs) to operate the network. Therefore, one of the main goals of the 6TiSCH WG is to define the 6top sublayer (see Sect. 3.1.1).

The 6TiSCH architecture [34] specifies how packets that belong to a deterministic IPv6 flow are marked and routed or forwarded over the mesh within jitter and latency budgets. The schedule management translates into routes and track management.

Route computation can be achieved either in a centralized fashion by a Path Computation Entity (PCE), which is located either on the backbone or farther in the IPv6 network over a backhaul, or in a distributed fashion using RPL and a multi-path resource reservation protocol.

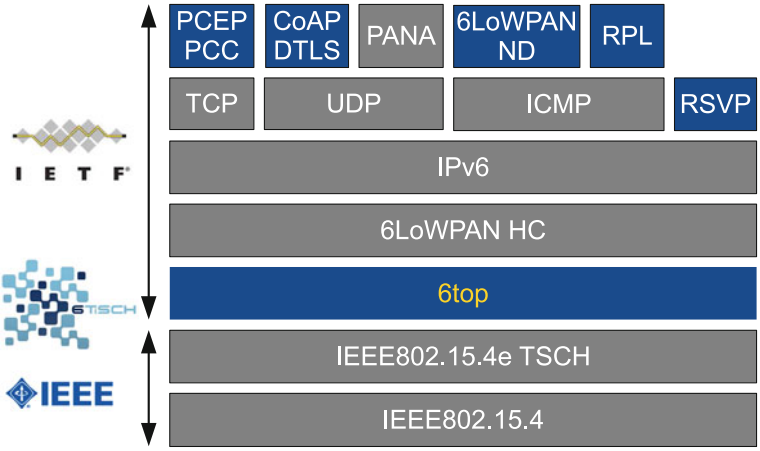


Fig. 5 6TiSCH IPv6-enabled protocol stack for LLNs

Track allocation can be globally optimized and then pushed on the network from the PCE that computes the routes, and/or managed by a distributed scheduling protocol along routes that are computed by RPL.

In detail, as shown in Fig. 5, in centralized approaches, schedule management is handled by a PCE and its control messages are transported by protocols such as PCEP/PCC [35] or COAP/DTLS [13] on top of UDP. In decentralized approaches, the management entities benefits from reservation protocols such as RSVP [31] or NSIS [36] where QoS requirements are transported and installed along a path, and then implemented as cell reservation in the schedule of each node on the path.

The 6TiSCH architecture also covers security. The security requirements will be identified, and then the group will work on a secure and scalable key management framework (and requirements of related protocols) for 6TiSCH networks. The Protocol for carrying Authentication for Network Access (PANA) [37] is a potential candidate as Key Management Protocol for the bootstrapping phase of the 6TiSCH networks. It is a promising protocol since it supports mutual authentication, stateless authentication relay function [38] and encrypted distribution of attributes [39].

3.1.1 The 6top Sublayer

In the 6TiSCH architecture [34], the 6top sublayer [40] is built on top of IEEE802.15.4e TSCH MAC layer, and allows a management entity to drive the TSCH schedule. 6top also includes statistics collection functionality, which an upper layer (including the RPL routing protocol) can use to gather connectivity information. The 6top sublayer offers management commands to upper layers in order to operate, configure and enforce QoS at the MAC layer. Monitoring processes are used to determine that particular cells do not perform as well as expected enabling its rescheduling so deterministic behavior can be maintained.

6top is designed to be used with several scheduling approaches. In a centralized approach, a central PCE collects topology and traffic requirements, used to build a communication schedule, which it then sends to the different nodes in the network. In a decentralized approach, nodes compute their own schedule according to local information or by using a decentralized resource reservation protocol. To enable both approaches, 6top adds a new IEEE802.15.4e `LinkOption` flag [17]; in addition to the `TX`, `RX`, `Shared` and `Timekeeping` flags, a cell is also qualified as either a *hard cell* or *soft cell*. This option is mandatory; all cells are either hard or soft.

- A *hard cell* is a cell that cannot be dynamically reallocated by 6top. This type of cell is typically scheduled by a PCE. Once installed, only the PCE can move it inside the TSCH schedule, or delete it. When installing a hard cell, the PCE indicates the exact `slotOffset` and `channelOffset` of the cell.
- A *soft cell* is a cell that can be reallocated by 6top dynamically. This type of cell is typically scheduled by a distributed scheduling entity in the upper layer of 6top. Instead of specifying the exact `slotOffset` and `channelOffset`, the scheduling entity indicates how many cells must be scheduled to a given neighbors. 6top is responsible for allocating specific `slotOffset` and `channelOffset` values corresponding to the request from the scheduling entity. Furthermore, the monitoring process of 6top keeps tracking the performance of each cell to the same neighbor. If a cell performs significantly worse than others scheduled to the same neighbor, 6top reallocates this cell at different `slotOffset` and `channelOffset` inside the TSCH schedule.

When using a centralized scheduler, the PCE needs a protocol to send schedule updates to the nodes in the network. Candidate protocols include PCEP [35], OpenFlow [41], and ForCES [42].

When using a distributed scheduler, a protocol is needed to reserve MAC-layer resources along the multi-hop path identified by RPL, to satisfy certain QoS constraints (e.g., bandwidth, latency). Candidate protocols include RSVP [31, 32] or NSIS [36]. NSIS provides the semantics for transport layer packets to visit each node along a multi-hop RPL path, and indicating Quality Of Service (QoS) requirements. Upon reception of a QoS request, the 6top layer configures the appropriate MAC layer resources.

6top maintains statistics about the performance of scheduled cells. When using a centralized scheduler, this information is periodically sent to the PCE, which continuously adapts the schedule and sends schedules updates as needed. This information can also be used by the RPL protocol's objective function.

6TiSCH network can transport different types of traffic, possibly for different administrative entities (e.g., lighting and HVAC data in a smart building), possibly with different QoS constraints. Thanks to the slotted nature of IEEE802.15.4e TSCH, 6top can mark different cells with identifiers of those different flows. This can result in perfect isolation. For example, the amount of HVAC traffic has no effect on the latency of the lighting traffic. This allows for true *umbrella* networks, managed by a network operator, and transporting data for different clients. An example is an

Table 1 6top sublayer management interface

ME to 6top	6top to ME
CREATE /DELETE/UPDATE.hardcell	Success/Failure
CREATE/DELETE/REALLOCATE.softcell	Success/Failure
CREATE/DELETE/UPDATE.slotframe	Success/Failure
CONFIGURE.monitoring	Success/Failure
CONFIGURE/RESET.statistics	Success/Failure
CONFIGURE.eb	Success/Failure
CONFIGURE.timesource	Success/Failure
CREATE/DELETE/UPDATE.neighbor	Success/Failure
CREATE/DELETE/UPDATE.queue	Success/Failure
CONFIGURE.security	Success/Failure
CONFIGURE.security.macKeyTable	Success/Failure
CONFIGURE.security.macSecurityLevelTable	Success/Failure
LabelSwitching.map	Success/Failure
LabelSwitching.unmap	Success/Failure
READ.cell	Configuration of a specific cell
READ.slotframe	Configuration of a specific slotframe
READ.monitoring.status	Allocated/provision cells
READ.statistics	Statistic MIB for given parameters
READ.eb	MIB of a specific Enhanced Beacon
READ.timesource	Timesource information in MIB
READ.neighbor	Specific neighbor's MIB
READ.all.neighbor	All neighbors in neighbor table
READ.queue.stats	Queue configuration in MIB

urban network which is used to transport data from weather sensors, and actuation commands for the municipal sprinkler system.

When a packet enters the 6TiSCH network, the 6top layer at the ingress device identifies the service this packet belong to, and marks the packet, possibly by using DSCP field in the 6LoWPAN header. When traveling through the 6TiSCH network, each mote uses that marker to decide on which cell to transmit.

The 6top sublayer provides a management interface to a Management Entity (ME). The interface consists of bidirectional message flows, one is from ME to 6top sublayer (`commands`); and another from 6top sublayer to ME (`status` or values of `MIB attributes`). The messages are defined in Table 1.

The Management Entity (ME) exists in an upper layer, which is responsible for exchanging control messages via network, and mapping the control messages to/from the management interface of 6top. For the centralized scheduling, five different control flows have been defined. In detail, as shown in Fig. 6, there are: *action* flow and *query* flow from PCE to nodes; *report* flow, *event* flow, and *schedule update* flow from nodes to PCE. In order to help the understanding of these flows, and related messages, let us consider the action flow, for instance. The PCE sends a control message to node `A` to create hard cells. Then, the ME in node `A` translates the control message to `CREATE.hardcell command` of 6top, and 6top returns `status`

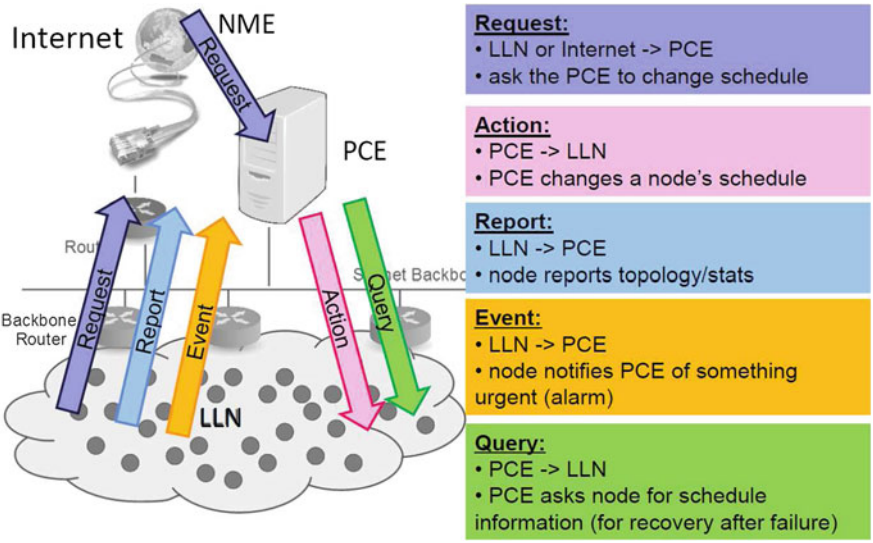


Fig. 6 Control flows defined between the Network Management Entity, NME (i.e, admin console, mobile handheld), or the ME (i.e., PCE) and the nodes in the LLN, for the centralized scheduling

`Success` or `Failure` to the ME. Finally, ME encodes the status into control message to PCE. Similarly, decentralized protocols such as RSVP or NSIS work with 6top through a Management Entity.

3.2 Centralized Scheduling

The 6TiSCH WG will initially focus on the definition of a static *minimal* schedule, which is pre-configured, or learnt by each mote when joining the network. The minimal 6TiSCH configuration draft [43] defines the basic setup for a TSCH network to inter-operate. Such setup includes the definition of a pre-configured schedule, the content of Enhanced Beacons, the TSCH MAC layer slot timing, the policy for selecting the time parent, and some insights for using the RPL Objective Function Zero [44] in a basic TSCH deployment.

Another approach is to use centralized scheduling. In the centralized approach, a key role is played by a Path Computation Element (PCE), which is a gateway connecting the network to the Internet (as shown in Fig. 4). It can be seen as the Management Entity (ME) responsible for building and maintaining the TSCH schedule. The PCE therefore can collect network state information and traffic requirements from the motes. Having a global view of the network, the PCE can build the schedules, making sure that the QoS requirements of all the network traffic flows are properly met.

Two different approaches can be used by the PCE for installing a track in the network. In detail, the PCE can

- talk to each node on the track individually, or
- talk only to the source node, which uses a separate protocol to install the resources along the track.

While the first approach seems to be easier to maintain, and also the one able to better ensure the correct installation of the track, the second one minimizes the amount of control traffic in the network between the PCE and the LLN.

6TiSCH will define the mechanism and format for control messages to be transported between the motes in the network. The 6TiSCH WG will first define the requirements for the protocol used by the PCE to communicate with a 6TSCH network, in order to identify existing protocols which may address (parts of) its needs. Transport protocols such as PCEP/PCC, and COAP/DTLS are possible candidate; they will be investigated and maybe adapted to the needs of the 6TiSCH architecture.

The Path Computation Element Communication Protocol (PCEP), specified in the RFC5440 [35], defines how to set up the communications between a Path Computation Client (PCC) and a PCE, or between two PCEs. In detail, the PCC can ask for path computation to the PCE, using a `PCReq` message that carries the specific requirements (e.g., bandwidth, metric-list, etc.) for the requested path. Upon reception of this request, the PCE replies with a `PCResp` message that specifies if the requirements, and thus, the path request, can be satisfied. Therefore, PCEP could be suitable for carrying the scheduling requirements of each traffic flow from the mote (playing the role of PCC) to the PCE. Security aspects such as data confidentiality, integrity and authentication will have to be taken into account while developing such centralized solution.

CoAP [13] appears as an interesting candidate as a transport mechanism between the central PCE or Management Entity (ME) and the 6top sublayer at the node. CoAP offers REST semantics with delivery guarantees while keeping low energy consumption profile. Operations on the 6top layer can be exposed as CoAP resources and accessed through well understood REST operations matching the operational flows between the ME and 6top as defined in the previous section. CoAP enables *Confirmable* and *Non-Confirmable* messages providing flexibility for those signals that need to be acknowledged and those events that can be transmitted unreliably. In addition, CoAP enables response piggybacking in message confirmation which reduces the cost of end-to-end acknowledgements plus responses. CoAP enables resources to be observed, analogously to an observer pattern, clients (e.g., the ME) can subscribe to resources on the server (e.g., a node in the network) and get notification upon changes. This feature enables the ME or PCE to subscribe to network information at nodes and be updated every time a node records new information.

The IETF DICE working group [45] is defining a subset of functionalities of DTLS to be supported in LLNs and together with CoAP provide a reliable, low footprint and secure framework to transport Management Entity requirements to 6top and vice-versa.

3.3 Distributed Scheduling

Distributed scheduling is analogous to building a set of paths (i.e., tracks) between nodes in the network. The direction of the tracks and the resources allocated for each of them are application-dependent and must be a parameter for the reservation mechanism. As in the case of transport networks (networks operating in the core of Internet), reservation of resources might be carried out by dedicated management protocols such as RSVP [31] or NSIS [36].

The operation of the protocol is driven by a path reservation request that travels along a existing track and ensures that there is connectivity between the two endpoints. At the end of the track, the reservation message is built and forwarded back, ensuring that the resources are available. The overlay used to transport the reservation requests is given by a distributed routing protocol such as RPL. In the 6TiSCH architecture, RPL is used to construct routing topologies using underlying L2 links.

In a distributed scheduling case, a LLN is formed by nodes receiving Enhanced Beacon (EB) messages from neighbors and selecting a best candidate to become its time source neighbor. Enhanced Beacons contain minimal information for nodes to establish a best effort schedule (i.e, shared cells) so they can start communicating [43]. Over these shared cells, RPL DIO and DAO control messages are used to discover and build an overlay topology used for routing. These best effort routes are used initially by nodes to reserve resources along a track (i.e., to a destination node—which usually is the DAGRoot). Once a best effort route between two nodes exists, the distributed Management Entity (an application running on the node), can start requesting resources to build a track.

This process is analogous to the reservation of resources in a wide area transport network. A reservation protocol (or a label distribution protocol) is used to transport QoS requirements (e.g., requested bandwidth) along a track. The 6TiSCH working group is evaluating different approaches to tackle transport of QoS requirements in so constrained devices, including reduced versions of RSVP and NSIS. The idea that the transport protocol, hop by hop, should require 6top sublayer to implement the provision of the required QoS on that hop. Upon a request for bandwidth allocation, for example, 6top can start a pairwise negotiation with the next hop node in order to agree on a set of cells to be used (Fig. 7).

How these cells are allocated is also in scope of 6top sublayer, and a set of recommendations will be defined by the working group, including an evaluation of a simple random cell allocation. Upon the request from a higher layer reservation protocol, the 6top sublayer will maintain the QoS levels as required. This task will also be transparent to the transport protocol, which will ensure that the QoS is enforced along the track. On its behalf, 6top continuously monitors the underlying MAC layer resources ensuring that QoS is met in a hop by hop basis by means of cell reallocation or over-provision.

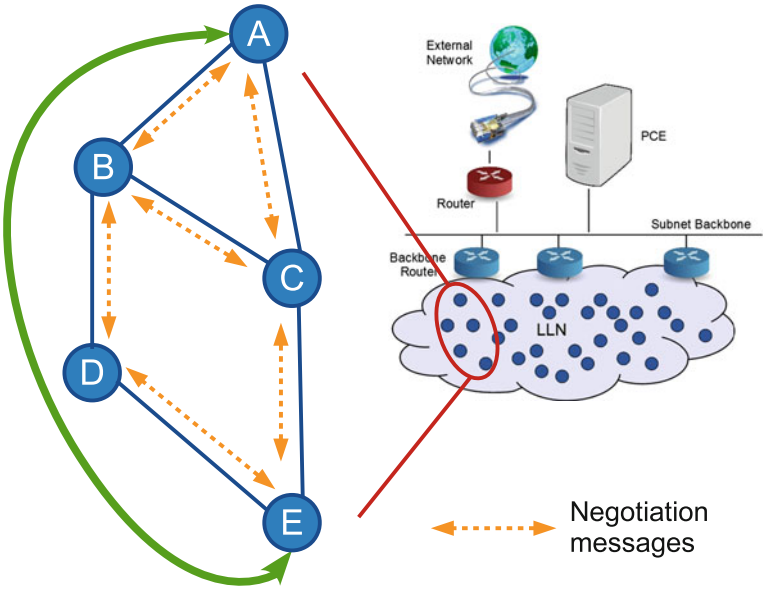


Fig. 7 Distributed scheduling applied to a set of nodes in the LLN. Node A sends a packet along the multi-hop path (track) A–B–D–E. At each hop, neighbor motes negotiate with one another to add cells into their TSCH schedule. A 6top monitoring process is needed to recover from topological changes and collisions

3.4 Routing in the 6TiSCH Architecture

Even though the initial efforts will concentrate on distributed routing over a static schedule, the 6TiSCH architecture ultimately aims at enabling a mix of centralized and distributed routing over a dynamic schedule.

A distributed routing model such as offered by RPL can react rapidly and autonomously to changes in the network, but misses knowledge on the global capacity of the network versus the load of individual links. It will typically direct all flows to a given destination over specific (shortest) paths, adding some excess load on some links when capacity exists elsewhere in the network, which limits its capability to serve optimally the end-to-end requirements of individual flows.

On the other hand, a centralized routing model such as offered by a PCE benefits from an overall perspective (also known as "God's view"), which enables the separate computation of multiple discrete paths that are globally optimized to meet their individual sets of constraints.

Either way, the path computation may be used for traditional hop-by-hop routing at the network layer, or for end-to-end switching at a lower layer along a predetermined track. Tracks can be installed on a per flow basis, so as to match a reservation with particular constraints along a path that is not necessarily shortest, whereas routes are typically shared by all flows as they converge towards a destination. It results that

hop-by-hop operations are generally associated with distributed routing computation, whereas track operations are generally associated with centralized path computation, but that is not necessarily always the case.

3.5 Forwarding in the 6TiSCH Architecture

The 6TiSCH architecture supports traditional IPv6 routing and forwarding, but in order to cover deterministic flows, the architecture also supports lower layer switching operations along predetermined tracks.

IPv6 Forwarding refers to the traditional network layer operation whereby a router selects a next-hop adjacent router for a given packet, typically based on a destination address in the network layer header of the packet. This operation is performed at each hop along a path, using a routing table (also known as Routing Information Base, RIB) that can be programmed by the PCE, computed dynamically through the RPL protocol or a mix of the two. At least one bundle of cells is associated to the link to the next-hop router.

A typical IPv6 Forwarding operation includes the following steps:

- Selection of a next-hop router based on network layer information in the packet (typically the destination IPv6 address) using a routing table;
- Selection of a link (a bundle of cells) to reach that router (if there are multiple, then QoS information may come into play);
- Edition of the MAC layer header to indicate the next-hop;
- Queuing for transmission over the selected bundle based on QoS information;
- (later) Transmission over a cell in the bundle based on queue priority and ageing (this is the actual scheduling operation).

Track Forwarding refers to a deterministic Generalized Multi-Protocol Label Switching (G-MPLS) [46] operation whereby a 6TiSCH node can switch a frame based on the cell of arrival as opposed to, classically, information in a header inside the frame. A track can be installed either by the PCE or by a reservation protocol, in which case the routes that are already present in the RIB will be used to select the path for the track. A Cell Switching Table uniquely binds a set of receive cells to a set of transmit cells, representing a forwarding state that can be used regardless of the upper layer protocol.

A typical Track Forwarding operation includes the following steps:

- Selection of the transmit cell or bundle based on the receive cell or bundle using the Cell Switching Table;
- Queuing for transmission over the bundle based on QoS information;
- (later) Transmission over a cell in the bundle based on queue priority and ageing (this is the actual scheduling operation).

Fragment Forwarding is an additional technique that is used to avoid the reassembly of 6LoWPAN fragments at each hop along a path in the LLN. The first fragment is routed using the IPv6 Forwarding method, and a state is installed for the

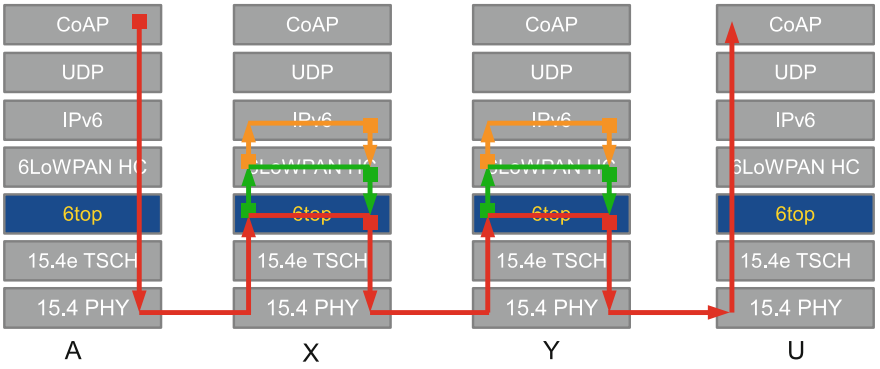


Fig. 8 6TiSCH forwarding mechanisms

subsequent fragments to follow the same route without a need to look up the routing table. The state is indexed by the *datagram tag* that is present in all fragments and identifies uniquely the packet. The *datagram tag* is locally significant; it is attributed at each hop upon the first fragment of a packet and switched in a classical MPLS fashion [47] upon the subsequent fragments [48].

Figure 8 shows the three forwarding techniques supported by the 6TiSCH architecture applied to the track installed between the node `A` and node `U` in Fig. 2. In detail, red, green and yellow arrows correspond respectively to Track, Fragment and IPv6 Forwarding. It has to be noticed that at the relay nodes (i.e., node `X` and node `Y`), the packet (or fragment) will reach a different layer of the protocol stack, according to the specific forwarding approach adopted in the network.

3.6 IPv6 Neighbor Discovery

Reachability within a subnet is classically obtained through the IPv6 Neighbor Discovery Protocol (NDP) [8]. But IPv6 ND relies heavily on multicast signalling messages on the local link, which is workable over the backbone but impractical for operations over a multilink subnet [49, 50]. Mobile IPv6 (MIPv6) [51] introduced a registration protocol to feed a Binding Table and enable reachability to a Mobile Node (MN) over an IP tunnel. In order to attract packets for a registered MN, a Home Agent (HA) performs IPv6 ND proxy operations over a Home Network where the subnet of the mobile node resides.

An IPv6 ND registration mechanism was standardized as Neighbor Discovery Optimization for Low-power and Lossy Networks [52], and extended by wireless ND [53]. The ND registration can also be used to feed a Binding Table for low-power devices that are attached directly to the Backbone Router. In detail, at the core of [53] there is the replacement of the multicast flooding, traditionally required for Duplicate Address Detection (DAD), with a unicast registration to a centralized binding table. The new message to create an entry is a `Duplicate Address Request`

(DAR) that is answered by a Duplicate Address Confirmation (DAC) message. Finally, a new Address Registration Option (ARO) is introduced which contains a unique ID for the device, typically an EUI-64 address.

In a fashion similar to a MIPv6 HA, the BBR can proxy the ND protocol over the backbone on behalf of registered node. In particular, it may advertise its own MAC address in order (i) to avoid leaking additional MAC addresses in the backbone, and (ii) to make sure it receives and stores the packets for a sleeping device till the device is reachable to receive them. For the case of a multilink subnet based on RPL meshes that are interconnected by a backbone, the RIB installed by the RPL protocol can be used to feed the Binding Table that will keep track of which IPv6 address this BBR proxies for.

An issue arises when the same address is registered asynchronously on two different BBRs, as it is unclear whether that is a device that just moved, or if it is an address that is duplicated between two devices. Whether the BBR inter-working is done through a routing protocol or classical IPv6 ND, there is a need for an extension to assert this. RPL, and to that regard any traditional routing protocol, will not consider that two advertisements can represent a duplication, but simply that there are probably two ways to get to the same device. On the other hand, the ARO in [52] can be used to find out when there is a duplication though a device unique ID as illustrated in Fig. 9a, but cannot tell which is the current state that must be conserved from the stale one, that should be cleaned up. RPL uses a sequence counter (called DAOSequence) to detect stale advertisements, and there is probably a need to enhance the ARO to add a similar indication (a Transaction ID, TID) for use within the ND registration mechanism, as illustrated in Fig. 9b.

Once the duplication problem is sorted out, there is still a need to discover and route over the backbone between a device that is attached to the backbone and a wireless device that is located inside a DODAG and reachable over the BBR. It is possible to extend RPL over the backbone and present the subnet as not-onlink in Router Advertisements, so as to always route, over the backbone and then along the DODAG. The alternate is a mixed mode over the backbone that consists in proxying ND operations. Upon a RPL route advertisement (called a DAO), the BBR that acts as root for the DODAG where a given device is located installs a host route towards the device over the LLN. Then, it advertises the device's address over the backbone using classical ND with extensions to check for duplication and movement. In this way, any legacy IPv6 device, using the classical IPv6 ND exchange of a Neighbor Solicitation (NS) and its Neighbor Advertisement (NA) response, resolves that the MAC address for the device is in fact that of the BBR. Then, it passes on the packet, which the BBR finally routes over the DODAG to the wireless destination. This procedure is illustrated in Fig. 10.

There are a number of questions to be answered in this proxy ND operation. In particular, how does this model work with multiple instances that are eventually rooted at different BBRs, and there are well-known possible answers such as the use of VLANs. Regardless of the answer, there is substantial work to be done to extend the simple model in [52] to operate over a backbone, and then enable routing from the backbone towards a LLN device.

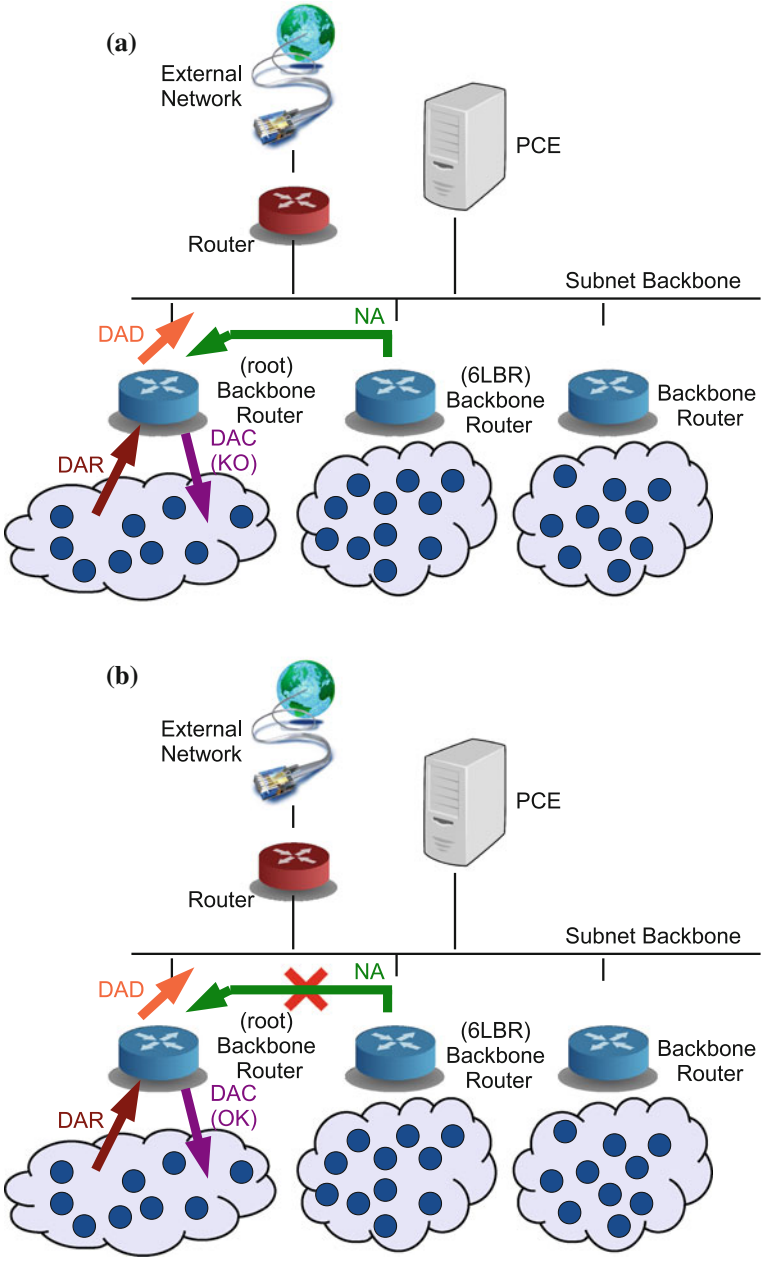


Fig. 9 Determination of duplication versus mobility. **a** Duplication: different EUI-64 in ARO option. **b** Mobility: same EUI-64 (newer TID wins)

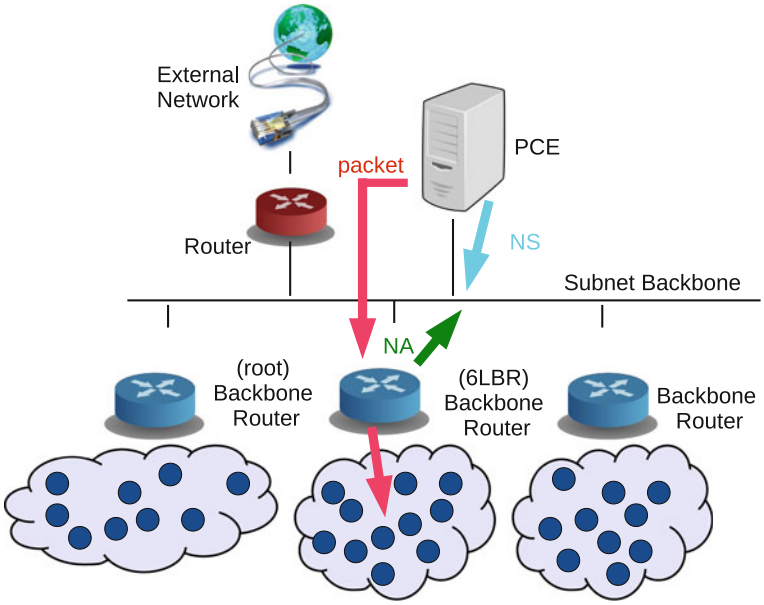


Fig. 10 ND resolution

4 Applications

As of today, several commercial low-power wireless networking providers are offering 99.999 % reliable MAC layers, for instance [54], that provide radio duty cycles well below 1 %, thereby reducing the mote power consumption and increasing the network lifetime [55]. This is facilitating the introduction of new monitoring and actuating devices as tools for operational technologies to improve the security, process automation, efficiency and productivity of the Industries [56, 57], and devices, a clear roadmap to the Industrial Internet paradigm [58].

The 6TiSCH working group is contributing to the consolidation of an open standards based protocol stack which will empower global adaptation of low power wireless technologies and will align Informational Technologies to Operational Technologies enabling the real take off of the Industrial Internet era. Hereafter we present and describe some of the potential applications that will benefit from the 6TiSCH architecture introduced in this chapter. Note that this is a non-exhaustive list.

- **Internet of Factory** WirelessHART and ISA100.11a are two industrial wireless standards, widely used in industrial process automation. Since 6TiSCH architecture brings the deterministic feature of the two aforementioned standards into IPv6 context, it allows to employ those technologies and functionalities provided by Internet community such as *traffic engineering*. The 6TiSCH architecture allows more dynamics in industrial wireless networks while keeping the deterministic

feature; in addition, it allows the industrial wireless networks to become part of the Internet automatically, enabling applications like *remote control*.

- **Smart Cities** Smart city involves many aspects such as smart economy, smart people, smart mobility, smart environment, smart living and smart governance. But, all of them need a common foundation, i.e., IoT infrastructure, which provides the connectivity among physical world, cyber world and people. As part of the IoT infrastructure, wireless sensor/actuator networks play a critical role, e.g., sensing occupancy of parking spots, sensing air quality, sensing traffic condition, alarming accidence, and control street light. These applications can be built on top of a IoT infrastructure based on open standards. By nature, the 6TiSCH architecture and protocol stack is a good solution for the IoT infrastructure in a smart city.
- **IoT Service Provider** Service Providers (SPs) have to deal with the different requests of their customers, and make sure that all of them are satisfied at the same time (whenever possible). The 6TiSCH architecture is very promising for service provider operating an *umbrella* network supporting different types of traffic flow, coming from several distinct customers. In fact, when a new customer asks to be granted access to the network, the network operator can predict with pretty good granularity the impact this new traffic will have on the remaining bandwidth of the network, and on the power consumption of individual motes. Thus, the operator will be able to grant/deny, and probably also charge differently each customer, according to the actual guaranteed service.
- **Internet of Buildings** The core functionality of building automation systems keeps the building climate within a specified range, provides lighting based on an occupancy schedule, monitors system performance and device failures, and provides malfunction alarms to building engineering/maintenance staff. The functionality reduces building energy and maintenance costs. By applying the 6TiSCH architecture and protocol stack, all of the sensors and actuators in the building automation system can be connected to Internet with very low energy consumption. Besides enabling more efficient and lower cost remote building controls, it will potentially let more information from the Internet input the building automation system, and make the buildings smarter.
- **Internet of Vehicles** Vehicular Automation is a system to assist vehicle operator. Manufacturers and researchers subsequently added a variety of automated functions to cars and other vehicles, and are now exploring how to take advantage of wireless networking technologies to integrate the vehicles into the Internet to improve driving safety, convenience and efficiency. There are two potential application fields, one is vehicle internal communication, another is vehicle-to-vehicle or vehicle-to-roadside communication. The functional requirement from the two fields are very different, but they require some features in common, e.g., both of them work in complex radio environment, while requiring highly reliable and deterministic network feature. Thus, the 6TiSCH architecture and protocol stack could be a candidate solution.
- **Internet of Home** Home automation is the residential extension of building automation. It is automation of the home, housework or household activity. Conventional home automation may include centralized control of lighting, HVAC

(heating, ventilation and air conditioning), appliances, security locks of doors, and other systems, to provide improved convenience, comfort, energy efficiency and security. In these home automation systems, different kinds of sensors and actuators play a critical role, and lower power operation is a welcome feature. Like other wireless networks, adopting 6TiSCH can make home automation systems more flexible and easier to be installed. Besides, more importantly, while the sensors and actuators are equipped with the 6TiSCH protocol stack, the conventional home automation system can be extended to a remote monitor and control loop with very low additional energy consumption, which involves the human beings related with the home but being far away. In addition, since the 6TiSCH protocol stack can bridge the sensors in home and monitors in hospitals or some care centers, the home automation for the elderly and disabled can provide more efficient and more economical services from caregivers or institutional care.

5 Conclusion

This chapter has introduced the work recently started at IETF by the 6TiSCH WG, and outlined the challenges that it will have to face when adopting cross IEEE and IETF standards within the IoT Industrial protocol stack for deterministic networks.

We have focused in particular on the 6TiSCH technically viable communication architecture, based on open standards, which will enable a new range of applications in automation (home, city, building), man-to-machine interfaces (cars, planes), and machine-to-machine communication.

The 6TiSCH WG is currently putting effort on developing distributed routing operation over a static TSCH schedule. At the same time, a *Minimal 6TiSCH Configuration* defining how to build a 6TiSCH networks using the Routing Protocol for LLNs (RPL) and a static TSCH schedule, is being produced. Finally, the 6TiSCH WG will also produce an *Information Model* containing the management requirements of a 6TiSCH node. Such model will include a description of the mechanisms to be used by an entity to (i) manage the TSCH schedule in a 6TiSCH node, and (ii) query timeslot information from that node. A *Data Model* mapping for an existing protocol, such as Concise Binary Object Representation (CBOR) over the Constrained Application Protocol (CoAP), will be also provided.

Acknowledgments This publication was supported by the FP7 projects IoT6-288445, CALIPSO-288879, RELYONIT-317826 and SWAP-251557, which are partially funded by the European Community. Xavier Vilajosana is funded by the Spanish Ministry of Education under Fullbright-BE grant (INF-2010-0319).

References

1. Postel, J.: Internet protocol, RFC 791, Internet Engineering Task Force (1981)
2. Andreasen, F., Foster, B.: Media gateway control protocol (MGCP) version 1.0, RFC3435, Internet Engineering Task Force (2003)

3. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session initiation protocol, RFC 3261, Internet Engineering Task Force (2002)
4. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V.: RTP: A transport protocol for real-time applications, RFC3550, Internet Engineering Task Force (2003)
5. Deering, S., Hinden, R.: Internet protocol, version 6 (IPv6) specification, RFC2460, Internet Engineering Task Force (1998)
6. Baker, F., Li, X., Bao, C., Yin, K.: Framework for IPv4/IPv6 translation, RFC6144, Internet Engineering Task Force (2011)
7. Conta, A., Deering, S., Gupta, M.: Internet control message protocol (ICMPv6) for the internet protocol version 6 (IPv6) specification, RFC4443, Internet Engineering Task Force (2006)
8. Narten, T., Nordmark, E., Simpson, W., Soliman, H.: Neighbor discovery for IP version 6 (IPv6), RFC4861, Internet Engineering Task Force (2007)
9. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.P., Alexander, R.: RPL: IPv6 routing protocol for low-power and lossy networks, RFC 6550, Internet Engineering Task Force (2012)
10. Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M.: Dynamic host configuration protocol for IPv6 (DHCPv6), RFC3315, Internet Engineering Task Force (2003)
11. Vida, R., Costa, L.: Multicast listener discovery version 2 (MLDv2) for IPv6, RFC3810, Internet Engineering Task Force (2004)
12. Postel, J.: User datagram protocol, RFC768, Internet Engineering Task Force (1980)
13. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained application protocol (CoAP), IETF CoRE Working Group (2011)
14. Palattella, M.R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L.A., Boggia, G., Dohler, M.: Standardized protocol stack for the internet of (important) things. IEEE Commun. Surv. Tut. **15**(3), 1389–1406 (2012)
15. 6TiSCH Mailing list available at: https://www.ietf.org/mailman/listinfo/6tsch
16. 6TiSCH homepage available at: https://bitbucket.org/6tsch/
17. IEEE802.15.4e: IEEE standard for local and metropolitan area networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer, Institute of Electrical and Electronics Engineers Std., April (2012)
18. Thubert, P., Watteyne, T., Palattella, M.R., Vilajosana, X., Wang, Q.: IETF 6TSCH: combining IPv6 connectivity with industrial performance. In: Proceedings of International Workshop on Extending Seamlessly to the Internet of Things (esIoT), Taiwan, July (2013)
19. Watteyne, T., Palattella, M.R., Grieco, L.A.: Using IEEE802.15.4e TSCH in an LLN context: overview, problem statement and goals. draft-watteyne-6tsch-tsch-lln-context-01 (work in progress), Feb (2013)
20. IEEE802.15.4: IEEE standard for local and metropolitan area networks - Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), Standard for Information Technology Std., Sept (2011)
21. Kushalnagar, N., Montenegro, G., Schumacher, C.: IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals, RFC 4919, Internet Engineering Task Force (2007)
22. Pister, K., Doherty, L.: TSMP: Time synchronized mesh protocol. In: Proceedings of International Symposium on Distributed Sensor Networks (DSN), Florida, USA (2008)
23. Highway Addressable Remote Transducer, a group of specifications for industrial process and control devices administered by the HART Foundation. Available at http://www.hartcomm.org
24. ISA, ISA100, Wireless systems for automation. Available at http://www.isa.org/Community/SP100WirelessSystemsforAutomation, May (2008)
25. Nixon, M.: A Comparison of WirelessHART and ISA100.11a (2012), July, white paper
26. Stanislowski, D., Vilajosana, X., Wang, Q., Watteyne, T., Pister, K.: Adaptive synchronization in IEEE802.15.4e networks. IEEE Trans. Industr. Inf. PP(99), 1 (2013)

27. Palattella, M.R., Thubert, P., Watteyne, T., Wang, Q.: Terminology in IPv6 over time slotted channel hopping. draft-palattella-6tsch-terminology-00 (work in progress), Mar (2013)
28. Palattella, M.R., Accettura, N., Grieco, L.A., Boggia, G., Dohler, M., Engel, T.: On optimal scheduling in duty-cycled IoT industrial applications using IEEE 802.15.4e TSCH. IEEE Sens. J. **13**(10), 3655–3666 (2013)
29. Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., Pister, K.: OpenWSN: a standards-based low-power wireless development environment. Trans. Emerg. Telecommun. Technol. **23**(5), 480–493 (2012)
30. uRes, available at https://openwsn.atlassian.net/wiki/display/OW/uRES
31. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource reservation protocol (RSVP) : version 1 functional specification. RFC 2205, Internet Engineering Task Force (1997)
32. Morellk, A., Vilajosana, X., Vicario, J.L., Watteyne, T.: Label switching over IEEE802.15.4e networks. Trans. Emerg. Telecommun. Technol. **24**(5), 458–475 (2013)
33. Thubert, P.: 6LoWPAN backbone router. draft-thubert-6lowpan-backbone-router-03 (work in progress), Feb (2013)
34. Thubert, P., Assimiti, R.A., Watteyne, T.: An architecture for IPv6 over time synchronized channel hopping. draft-thubert-6tsch-architecture-00 (work in progress), March (2013)
35. Vasseur, J.P., Le Roux, J.L.: Path computation element (PCE) communication protocol (PCEP) RFC 5440, Internet Engineering Task Force (2009)
36. Hancock, R., Karagiannis, G., Loughney, J., Van den Bosch, S.: Next steps in signaling (NSIS): framework. RFC 4080, Internet Engineering Task Force (2005)
37. Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., Yegin, A.: Protocol for carrying authentication for network access (PANA), RFC 5191, Internet Engineering Task Force (2008)
38. Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Yegin, A.: Protocol for carrying authentication for network access (PANA) relay element, RFC 6345, Internet Engineering Task Force (2011)
39. Yegin, A., Cragie, R.: Encrypting the protocol for carrying authentication for network access (PANA) attribute-value pairs, RFC 6786, Internet Engineering Task Force (2012)
40. Wang, Q., Vilajosana, X., Watteyne, T.: 6tus adaptation layer specification. draft-wang-6tsch-6tus-00 (work in progress), March (2013)
41. The OpenFlow Switch Specification. Available at http://OpenFlowSwitch.org
42. Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang, W., Dong, L., Gopal, R., Halpern, J.: Forwarding and control element separation (ForCES) protocol specification RFC 5810, Internet Engineering Task Force (2010)
43. Vilajosana, X., Pister, K.: Minimal 6TSCH configuration, draft-vilajosana-6tsch-basic-01 (work in progress), July (2013)
44. Thubert, P.: Objective function zero for the routing protocol for low-power and lossy networks (RPL), RFC 6552, Internet Engineering Task Force (2012)
45. IETF Working Group: DTLS In Constrained Environments (DICE), - charter at http://datatracker.ietf.org/wg/dice/charter/
46. Mannie, E.: Generalized multi-protocol label switching (GMPLS) architecture, RFC3945, Internet Engineering Task Force (2004)
47. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol label switching architecture, RFC3031, Internet Engineering Task Force (2001)
48. Thubert, P., Hui, J.W.: LLN fragment forwarding and recovery, draft-thubert-roll-forwarding-frags-02 (work in progress), Sept (2013)
49. Thaler, D., Huitema, C.: Multi-link subnet support in IPv6, draft-ietf-ipv6-multilink-subnets-00.txt, Internet Draft, Internet Engineering Task Force (2002)
50. Thaler, D.: Multi-link subnet issues, RFC4903, Internet Engineering Task Force (2007)
51. Perkins, C., Johnson, D., Arkko, J.: Mobility support in IPv6, RFC 6275, Internet Engineering Task Force (2011)
52. Shelby, Z., Chakrabarti, S., Nordmark, E., Bormann, C.: Neighbor discovery optimization for IPv6 over low-power wireless personal area networks (6LoWPANs), RFC 6775, Internet Engineering Task Force (2012)

53. Chakrabarti, S., Nordmark, E., Wasserman, M.: Efficiency aware IPv6 neighbor discovery optimization draft-chakrabarti-nordmark-6man-efficient-nd-01 (work in progress), Nov (2012)
54. Doherty, L., Lindsay, W., Simon, J.: Channel-specific wireless sensor network path data. In: Proceedings of IEEE ICCN 2007 Conference, pp. 89–94 (2007)
55. Dust Networks Linear Technology (2013) Smart mesh ip
56. Vilajosana, I., Llosa, J., Martinez, M., Pacho, J.C.: Wireless sensors helps monitoring one of world most advanced load and unload harbor terminals, White Paper available at http://www.loadsensing.com (2012)
57. Emerson: Emerson wireless technology helps RWE maximize gas storage capacity and improve efficiency and safety, White Paper available at http://www.emersonpress.com (2013)
58. Evans, P.C., Annunziata, M.: Industrial internet: Pushing the boundaries of minds and machines, White Paper available at http://www.ge.com (2012)

Vehicular Ad Hoc Networks and Trajectory-Based Routing

Yanmin Zhu, Yuchen Wu and Bo Li

Abstract With the rapid advance in short-range radio communications for vehicles (e.g., IEEE 802.11p), vehicular ad hoc networks have emerged as a new paradigm of mobile ad hoc networks. Within a vehicle ad hoc network, there are several types of wireless communications, including vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and infrastructure-to-vehicle (I2V). A wide variety of existing applications are being developed based on vehicular ad hoc networks, such as traffic safety, transport efficiency, and entertainment on the move. A key enabling technology for vehicular ad hoc network is message routing among vehicles. Efficient message routing is particularly challenging for vehicular ad hoc networks because of frequent network disruption, fast topological change and mobility uncertainty. The vehicular trajectory knowledge plays a key role in message routing. By extracting mobility patterns from historical vehicular traces, we develop trajectory predictions by using multiple order Markov chains. We then present routing algorithms taking full advantage of predicted probabilistic vehicular trajectories. We carry out extensive simulations based on large datasets of real GPS vehicular traces. The simulation results demonstrate that the trajectory based routing algorithms can achieve higher delivery ratio at lower cost.

1 Introduction

With the rapid advance in short-range radio communications for vehicles (e.g., IEEE 802.11p), vehicular ad hoc networks have emerged as a new paradigm of mobile ad hoc networks. A vehicular ad hoc network is a network of vehicles which communicate with each other via short-range wireless communications [1]. Within a vehicle ad hoc network, there are several types of wireless communications,

Y. Zhu (✉) · Y. Wu · B. Li
Department of Computing Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
e-mail: yzhu@cs.sjtu.edu.cn

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_6,

including vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and infrastructure-to-vehicle (I2V). Vehicular ad hoc networks have many appealing applications, such as driving safety [2], intelligent transport [3, 4], and entertainment on the move.

Efficient message routing is of central importance to vehicular ad hoc networks [5–11]. Message routing in a vehicular ad hoc network may introduce nonneligible delivery latency because of frequent network topology disconnections. Thus, we should stress that the message routing in vehicular ad hoc network is suitable for those applications which can tolerate certain delivery latency. There are many examples of such applications, including peer-to-peer file sharing, entertainment, advertisement, and file downloading.

The knowledge of future vehicular traces plays a key role for optimal routing. Existing routing algorithms heavily rely on prediction of vehicle mobility. However, they have adopted only simple mobility patterns, such as the spatial distribution and inter-meeting time distribution, which support coarse-grained predictions of vehicle movements. Some algorithms [9, 12] assume random mobility in which vehicles move randomly in an open space or a road network. This model is simple but far from the reality. Some other algorithms assume simple mobility patterns such as exponential inter-meeting times and regular spatial distributions. As a result, prediction results based on these simple patterns are of limited value to efficient routing in vehicular ad hoc networks. In addition, many of existing algorithms ignore the fact that links in a vehicular ad hoc network have unique characteristics [11]. On the one hand, a link is typically short-lived. This suggests that the capacity of the link is limited. Thus, the order for forwarding packets becomes important. On the other hand, links in a densely populated area may interfere with each other. This indicates that link scheduling becomes necessary.

To overcome the limitations of existing algorithms, this chapter presents an approach to taking full advantage of trajectory predictions. By analyzing an extensive dataset of real vehicular traces, we find that there is strong spatiotemporal regularity with vehicle mobility. More specifically, our results based on conditional entropy analysis demonstrate that the future trajectory of a vehicle is greatly correlated with its previous trajectory. Thus, we develop multiple order Markov chains for predicting future trajectories of vehicles. With the available future trajectories of vehicles, we present an analytical model and theoretically derive the delivery probability of a packet. We develop an efficient global algorithm and a fully distributed algorithm which needs only localized information. The two algorithms jointly consider packet scheduling and link scheduling. We evaluate the algorithms with extensive trace driven simulations, based on the trace dataset collected in Shanghai and Shenzhen. The results demonstrate that our algorithm considerably outperforms other algorithms in terms of delivery probability and delivery efficiency.

The rest of the chapter is organized as follows. In Sect. 2, we briefly introduce vehicular ad hoc networks. Next, we present the network model and formulate the problem of message routing in vehicular ad hoc networks. In Sect. 4 we analyze vehicle traces and present a mobility model based on Markov chain. Section 5 describes a global routing algorithm and distributed routing algorithm. Simulation results are presented in Sect. 6. The chapter is concluded in Sect. 7.

2 Vehicular Ad Hoc Networks

As illustrated in Fig. 1, a vehicular ad hoc network is simply a system of vehicles that communicate with each other with short-range radios, such as 5.9 GHz DSRC. Besides inter-vehicle communications, vehicles can also communicate with the infrastructure that is comprised by roadside units (RSUs).

Conceptually, inexpensive wireless local area network (WLAN) technologies, such as WiFi, can be used for V2V, V2I, and I2V communications. However, new challenges arise by high vehicle speeds and highly dynamic network environments. In addition, new requirements are imposed by applications of vehicular ad hoc networks, especially by those safety-critical applications, e.g., high reliability of message delivery, low packet latency.

In response to the new challenges and new requirements, new standards have been developed. The American Society for Testing and Materials (ASTM) subcommittee E17.51 examined issues concerning vehicle roadside communications and proposed DSRC. It is largely based on the IEEE 802.11a standard, including the physical (PHY) layer and the media access control (MAC) layer. Currently, the IEEE 802.11p working group continues the main tasks of DSRC.

Compared with traditional mobile ad hoc networks, vehicular ad hoc networks face several new challenges.

First, vehicular ad hoc networks are subject to frequent disruptions. It is difficult to find a connected path between a pair of source and destination in vehicular ad hoc networks. This is caused by high mobility and uneven distribution of vehicles over the network.

Second, vehicles usually move at a high speed of up to 80 Km/h. Two vehicles can communicate only when they are within the communication range. Recent study has shown that the contact duration in case of a vehicle and a static access point is as short as 10 s on average [13].

Finally, there is a great deal of uncertainty associated with vehicle mobility. Vehicles move at their own wills. It is difficult, if not impossible, to gain the complete knowledge about the vehicular trace of future movement, i.e., the position of the vehicle at a given point in time. For message routing in a vehicular ad hoc network, a relay node must decide how long a packet should be kept and which node a given packet should be forwarded to. Existing study [14, 15] shows that it is possible to find an optimal routing path when the knowledge of future node traces is available, which is NP-hard though. However, it is impractical to have prior knowledge about future traces of nodes.

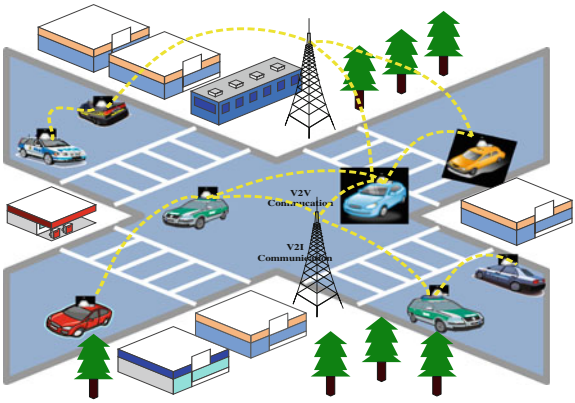


Fig. 1 The illustration of vehicle-to-vehicle communications and vehicle-to-infrastructure communications in a road intersection

3 Network Modelling and Routing Complexity

3.1 Network Model

The vehicular ad hoc network is modeled as a set of nodes, N. When two nodes, i and j, are in the communication range (denoted by D), i.e., $d_{ij} < D$, there is a link between the two nodes and they can communicate with each other while the link exists.

The position of node n at time τ is denoted by $p_n(\tau)$. The time is slotted. Therefore, the trajectory of node n is a sequence of positions, denoted by

$$T_n = \langle p_n(0), p_n(1), ..., p_n(\tau) \rangle . \tag{1}$$

The distance between two nodes, i and j, at time τ is denoted by $d_{ij}(\tau)$

Links in the network are changing with the relative positions of the nodes. The set of all possible links at time τ is denoted by $L(\tau)$,

$$L(\tau) = \{l_{i,j} | d_{i,j}(\tau) \leq D; \forall i, j \in N\}. \tag{2}$$

Links are assumed to have the same capacity in theoretical analysis.

The vehicular ad hoc network tries to deliver a set of packets, denoted by Φ, and the packets are of equal size. A packet has a source, $\delta(p)$, and a destination, $\psi(p)$. A packet is copied and forwarded from one node to another if there is a link between them. A packet group, $\theta(p)$, is introduced to denote all the copies of packet p in the network. The size of packet group, $|\theta p|$, increases when there is a new copy-forward process of packet p in the network.

3.2 Formulation of Routing in Vehicular Ad Hoc Networks

The main goal of message routing is to move the packets from their sources to respective destinations. The delivery performance objectives include delivery ratio, delay and efficiency. In the following, we first analyze the properties of individual packets, and then show the objectives in a global view.

The delivery probability is the chance of a packet successfully delivered from its source to its destination. The delivery probability of packet p, denoted by ρ_p, depends on the routing strategy, Υ, and the trajectories of nodes,

$$\rho_p = f_\Upsilon(\delta(p), \psi(p), \{T_n | n \in N\}). \tag{3}$$

With given trajectories, ρ_p can be calculated at the beginning and it does not change over time. But in the real world where the future traces are uncertain, this delivery probability can be considered as a random variable.

The delay of a packet, denoted by σ_p, is defined as the total transmission time spent for the network to deliver the packet to its destination. Analysis of delay is similar to that of delivery probability. The delay consists of two parts. One part is the time already spent, and the other is time to spent,

$$\sigma_p(\tau) = \tau + \Gamma_\Upsilon(\delta(p), \psi(p), \{T_n\}). \tag{4}$$

When $\sigma_p \tau = \tau$, the delivery of the packet is complete.

The cost of the transmission of a packet, denoted by ζ_p, is defined as the total number of times that a packet is forwarded. A new copy is created when a packet is forwarded. Thus, the total number of copies indicates the delivery cost. The cost is therefore defined as

$$\varsigma_p = |\theta(p)|\,. \tag{5}$$

One of the common objectives of vehicular ad hoc networks is to maximize *delivery ratio*. The delivery ratio is defined as the proportion of successfully delivered packets to the total packets to be transmitted. The number of total packets is often omitted in calculations because it is fixed and not affected by routing strategies. Thus, one objective can be given by

$$\max \sum\nolimits_{p \in \Phi} \mathbb{E}[\rho_p]. \tag{6}$$

Thus, the problem is to find the optimal routing of the packets through the vehicular ad hoc network that meets (6).

Two other common objectives are minimization of delay and minimization of total cost, which are given respectively by

$$\min \sum\nolimits_{p \in \Phi} \mathbb{E}[\sigma_p], \text{ and } \min \sum\nolimits_{p \in \Phi} \mathbb{E}[\varsigma_p]. \tag{7}$$

The efficiency is defined as the ratio of total successfully delivered packets to the total cost, given by

$$\ell = \sum\nolimits_{p\in\Phi} \mathrm{E}[\rho_p] / \sum\nolimits_{p\in\Phi} \mathrm{E}[\varsigma_p]. \tag{8}$$

A high efficiency indicates that one algorithm achieves high delivery ratio at a low cost.

3.3 Analysis of Routing Complexity

The optimal routing for achieving the objectives mentioned above is extremely difficult to design. For analysis simplification, we first assume that the routing algorithm have the complete knowledge of all vehicles and their future traces, and the set of packets to be transmitted. Note that this is impractical in the real world.

Without loss of generality, we take the objective of maximizing the delivery ratio for example. Objective (6) can be written as,

$$\max \sum\nolimits_{p\in\Phi} \mathbb{E}[f_\Upsilon(\delta(p), \psi(p), \{T_n | n \in N\})]. \tag{9}$$

Theorem 1: This routing problem with objective (9) when the vehicular traces and the set of packets to transmit are given is NP-hard.

The concise proof for the theorem is as follows. The well studied edge-disjoint path (EDP) problem is known to be NP-hard, which tries to find the maximal number of edge-disjoint paths. By reducing the EDP problem to the ideal routing problem, we can prove the ideal routing problem is also NP-hard [8]. The basic reduction procedure is as follows. The vertices of EDP are mapped to the vehicles, and each edge is mapped to a contact of vehicles. The source destination pairs are mapped to the source and destination pairs of the packets to transfer. As a result, the routes are valid edge-disjoint paths.

In practice, the complete knowledge is hard to be obtained because the future knowledge of vehicular traces is usually unavailable. Following this practical constraint, the trajectory of a node is divided into two parts: the past and the future, denoted by

$$\{T_n\} = \{T_n|_{\tau\le t}\} \cup \{T'_n|_{\tau>t}\}. \tag{10}$$

Let $\lambda_t(p)$ to denote the current position of the packet. The objective at time t for routing becomes

$$\max \sum\nolimits_{p\in\Phi} \mathbb{E}[\rho_p] = \max \sum\nolimits_{p\in\Phi} \mathbb{E}[f_\Upsilon(\lambda_t(p), \psi(p), \{T_n\}, \{T'_n\})]. \tag{11}$$

The second part of the trajectory is different from the first part. The first part has been fixed while the second part is not fixed and unknown at the time of being.

Since the objective of delay minimization has similar formation, we generalize (11) to the following,

$$\max \sum\nolimits_{p\in\Phi,n\in N} \Theta(\lambda_t(p), \psi(p), \{T_n|_{\tau\le t}\}, \{T'_n\}). \tag{12}$$

where Θ denotes the global metric for the whole network. If we omit the inputs, the objective becomes

$$\max \sum\nolimits_{p\in\Phi} \Theta_{p,N,t}. \tag{13}$$

This is the practical objective with the limited knowledge beyond time t.

4 Mobility Analysis and Modelling

In this section, we quantitatively reveal the spatiotemporal regularity with vehicle mobility. The quantitative analysis is based on mining the large dataset of vehicular traces of more than 4,000 taxis in Shanghai, China, which have a duration of more than two years. The GPS trace of a taxi, equipped with a GPS receiver, was collected by the vehicle periodically sending its instant position to a data collection center at an interval from 10 s to several minutes. The taxis operates in the whole urban area of Shanghai, which covers an area of over 120 km^2.

For simplification of discussion, the whole space is divided into Q small grids., and is denoted by S,

$$S = \{s_0, s_1, \ldots, s_{Q-1} | s_i \cap s_j = \emptyset\}. \tag{14}$$

where Q is the total number of grids. The time is slotted. The location of a vehicle at a given time is considered as a random variable which takes state values from the grid space. Let S_i denote the random variable for vehicle i. We reveal the spatiotemporal regularity by computing the marginal and the conditional entropies of S_i given the previous K states.

For vehicle i, suppose we have observed its states for L time slots. The state sequence of the vehicle can be denoted by a vector $T_i = < s_0, s_1, \ldots, s_{L-1} >$, where $s_j \in S$, $0 \le j \le L-1$ is the positional state of vehicle i at time slot j. Suppose that s_j appeares o_j times in the vector of T_i, $0 \le j \le L-1$. Thus, the probability of vehicle i taking state s_j can be computed as o_j/L. Then, the marginal entropy of S_i is

$$H(S_i) = \sum\nolimits_{j=0}^{Q-1} (o_j/L) \times \log_2 \frac{1}{o_j/L}. \tag{15}$$

Next, we compute the conditional entropy of S_i given its immediately previous state S_i^1 which has the same distribution with S_i. The conditional entropy is,

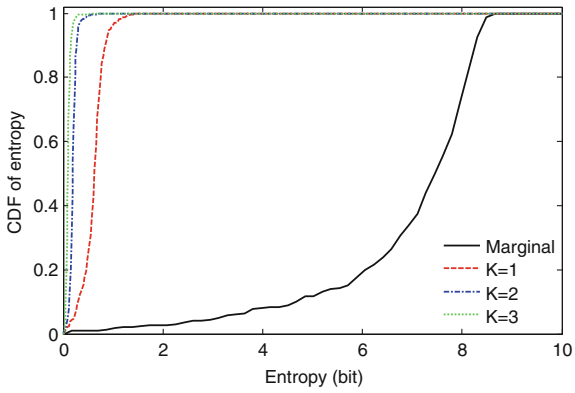


Fig. 2 The CDFs of marginal entropy and conditional entropies of a vehicle positional state. The grid size is 200 m × 200 m, and the number of vehicles is 500

$$H(S_i|S_i^1) = H(S_i, S_i^1) - H(S_i). \tag{16}$$

To derive the conditional entropy, we have to derive the entropy of the joint random variable (S_i, S_i^1) By using the state sequence T_i, we derive a sequence of 2-tuples, $T_i^1 = < (s_0, s_1), (s_1, s_2), \ldots, (s_{L-2}, s_{L-1}) >$. By counting the occurrences of (s_{j-1}, s_j), denoted by o_{j-1}, j, we can get its probalibty. Thus, the joint entropy of the two dimensional random variable of (S_i, S_i^1) is,

$$H(S_i, S_i^1) = \sum_{j=1}^{Q-1} \frac{o_{j-1,j}}{L-1} \times \log_2 \frac{1}{o_{j-1,j}/(L-1)}. \tag{17}$$

By generalizing the previous computation, we can compute the conditional entropy of S_i given its immediately previous K states $S_i^1, S_i^2, \ldots, S_i^K$

$$H(S_i|S_i^1, S_i^2, \ldots S_i^K). \tag{18}$$

In essence, by showing the marginal entropy we reveal spatial regularity, which characterizes the uncertainty of a vehicle residing a location in the space. By showing the conditional entropy given the previous states, we reveal the spatiotemporal regularity of vehicle mobility. This characterizes the uncertainty of a vehicle residing a specific location when its previous states are given. This demonstrates the correlation of a vehicle's positional states over different times.

In Fig. 2, the cumulative distribution functions (CDFs) of the marginal and the conditional entropies for are shown. The conditional entropies $K = 1, 2, 3$ are significantly smaller than the marginal entropy. This implies that the uncertainty of the positional state becomes smaller when the previous states are known. We also find that when K becomes larger, the entropy continues to decrease. This suggests that more previous states help further reduce uncertainty. However, the improvement quickly stalls K as increases. This gives the guidance to the order selection for trajectory prediction using multiple order Markov chains.

4.1 Modelling Vehicular Mobility with Markov Chain

A mobility pattern of a node characterizes the regularity of its mobility.

Definition 1. The mobility pattern of node n, denoted by MP_n, is defined as a pair of random variables, $MP_n = (\alpha, \beta)$. The probability distribution of the mobility pattern characterizes the regularity of the vehicle's mobility.

We develop the following mobility pattern for characterizing the spatiotemporal regularity of vehicle mobility. For vehicle i, its mobility pattern is defined as $M_i = (H, F)$, where H is a vector of past positional states, $H = < H_0, H_1, \ldots, H_{K-1} >$, $K \geq 1$, and F is the future positional state. The mobility pattern characterizes the frequencies of F following H in the vehicle's traces by computing the probability distribution of the two-dimensional random variable M_i. This can be achieved by analyzing the historical traces of the vehicle, as discussed in Sect. 3.

The notion of mobility pattern is general enough to cover the regularities that have been studied. For the study of inter-meeting times, the mobility pattern is

$$MP_n = (A, B), \tag{19}$$

where A is a random variable representing the inter-meeting time of vehicle n and a second vehicle, and B is the random variable denoting the second vechile. Based on this mobility pattern, we are able to describe the distribution of inter-meeting times between n and any other vehicle.

For the study of spatial distribution of a vehicle, the mobility pattern is

$$MP_n = (S, \emptyset)\,, \tag{20}$$

where S is the random variable representing the spatial state and ϕ denotes a null random variable.

To predict the trajectory of a vehicle, exiting simple patterns are inadequate. In the next subsection, we present our method of trajectory prediction by developing multiple order Markov chains based on the spatiotemporal mobility pattern.

4.2 Predicting Future Trajectories

The problem of predicting the trajectory of vehicle i at time t is to compute the future trajectory $T_i|_{\tau>t}$, given the trajectory before t, $T_i|_{\tau<t}$. A trajectory, T, can be described by a sequence of positional states,

$$T = < s^0, s^1, \ldots, s^\tau, \ldots > . \tag{21}$$

Because of the uncertainty associated with vehicle mobility, there may exist a number of possible future trajectories.

Definition 2. The set of all possible trajectories of node n is defined as a trajectory bundle, denoted by TB_n, which can be characterized by

$$TB_n = < D^n(1), D^n(2), \ldots, D^n(\tau), \ldots > . \tag{22}$$

where $D^n(\tau)$ is the probability distribution of spatial states at future time τ.

$$D^n(\tau) : P_s^n(\tau) | s \in S. \tag{23}$$

where $P_s^n(\tau)$ is the probability of the node n appearing in state s at time τ.

To predict the trajectory of a vehicle, it is essentially to calculate the trajectory bundle of the vehicle. For trajectory prediction, we develop multiple order Markov chains for predictions. The key is to establish the matrix of transition probabilities. The transition probabilities are computed on an individual vehicle's basis, since different vehicle possesses different regularities.

For a vehicle, we can easily derive its transition matrix (denoted by X) from its mobility pattern. When we are using a K-order Markov chain, an element, $x_{ij} \in X$, represents the transition probability from H_i to F_j, where H_i is a sequence of positional states, $H_i = < h_0^i, h_1^1, \ldots, h_{k-1}^i >$, and F_j is a single state. Then, $x_{ij} = Pr\left(H_i, F_j\right)$, where $Pr(\cdot)$ is the probability distribution function of the mobility pattern of the vehicle.

By applying the K-order Markov chain, we can compute the trajectory bundle as follows. Given the current trajectory of node n is $T_c^n = < s_{-k+1}, s_{-k+2}, \ldots, s_0 >$, the initial distributions for T_c^n are

$$D^n(\tau) : \begin{cases} P_{s_\tau}^n(\tau) = 1 \\ P_s^n(\tau) = 0, s \neq s_\tau \end{cases}, \quad (\tau \leq 0). \tag{24}$$

The distributions of spatial state at future times can be iteratively calculated. For a single trajectory (denoted by $<< s_1, \ldots, s_k >, s >$, its probability is

$$P^n_{<s_1,..,s_K>,s}(\tau) = x_{<s_1,..,s_K>,s} \times \prod_{i=1}^{K} P_{s_i}^n(\tau - K + i). \tag{25}$$

$P_{si}^n(t)$ are derived from the previous distributions $x_{<s_1,\ldots,s_k}, s$ and is defined in the transition matrix. Then, $D_n(\tau)$ can be derived by

$$D^n(\tau) : P_s^n(\tau) = \sum\nolimits_{all<s_i>} P_{<s_i>,s}{}^n(\tau), \ \tau > 0. \tag{26}$$

4.3 Computing Inter-Vehicle Delivery Probability

With the predicted trajectories, we are able to derive the encounter probabilities which are required for computing the eventual delivery probabilities.

Given the trajectory bundles of node i and j, $D^i(\tau)$ and $D^j(\tau)$ are known. Let $\varepsilon_{i,j}(\tau)$ denote the encounter probability of the two nodes at time τ. It can be calculated by

$$\varepsilon_{i,j}(\tau) = \sum\nolimits_{s \in S} P_s^i(\tau) \times P_s^j(\tau). \tag{27}$$

Then, the encounter probability, $\varepsilon_{i,j}$ is given by

$$\varepsilon_{i,j} = 1 - \prod\nolimits_{\tau=t}^{T} (1 - \varepsilon_{i,j}(\tau)), \tag{28}$$

where T denotes the max prediction range of time.

If the overall objective is to minimize the delivery delay, the estimated encounter time for the two nodes can be derived by

$$\eta_{i,j} = \sum_{\tau=t}^{T} \tau \times \varepsilon_{i,j}(\tau) / \sum_{\tau=t}^{T} \varepsilon_{i,j}(\tau). \tag{29}$$

Therefore, the estimated delay of a packet can be calculated through trajectories in a similar way with delivery probability computation.

5 Trajectory-Based Routing

In this section we first present the global routing algorithm and the distributed routing algorithm.

5.1 Global Algorithm

Designing the optimal routing for achieving (13) is difficult, especially without any knowledge about future vehicular traces. We present a global routing algorithm with only predicted trajectories of vehicles.

The global metric of a packet changes after intermediate transfers over time. Consider that a packet, p, is forwarded from node i to j at time t. After the forwarding, the increment in the global metric becomes

$$\Delta\Theta_{p,N.t} = \Theta_{p,N,t}(j) - \Theta_{p,N,t}(i). \tag{30}$$

And the original objective becomes

$$\max \sum\nolimits_{p\in\Phi} (\sum\nolimits_{\tau=0}^{t-1} \Delta\Theta_{p,N,\tau} + \Delta\Theta_{p,N,t}). \tag{31}$$

Since the sum of increments before time t has been fixed, it can further be written as

$$\max \sum\nolimits_{p\in\Phi} \Delta\Theta_{p,N,t}. \tag{32}$$

Item $\Delta\Theta$ can be considered as the current metric for packet p.

At time t, there may exist a number of links that may inference with each other, and it is impossible for all the links to be active simultaneously. The routing algorithm must schedule the links, i.e., choose a subset of links from all possible links to maximize the sum of current metrics. Thus, the objective becomes

$$\max \sum\nolimits_{p\in\Phi,\ l\in L(t)} (\Delta\Theta_{p,N,t} \times l_{i,j}). \tag{33}$$

The optimal routing performs two tasks, i.e., link scheduling and packet scheduling. Before the routing decision can be made, we have to obtain the metric increment of every packet over all possible transfers. When all the increments are available, the optimal routing chooses such a set of links and a set of packet transfers that meet (33).

We should emphasize that even when all $\Delta\Theta_{p,N,t}$ are available, finding the best links and packet transfers is still an NP-hard problem. A brief proof is as follows. This problem can be considered as a weighted maximum independent set problem that has been proved to be NP-hard. To solve this problem, a number of existing heuristic algorithms [16] can be used, in which $\Delta\Theta_{p,N,t}$ are considered as link weights.

As mentioned before, we take delivery probability maximization as example. The delivery probability of packet, ρ_p, depends on the encounter probability of every two nodes, denoted by $\epsilon_{i,j}$, $\forall i, j \in N$. In the following we derive the relationship between metric increments and encounter probability.

The probability for the packet to be delivered in no more than one hop is

$$\rho_p^1 = \varepsilon_{\lambda(p),\psi(p)}. \tag{34}$$

Then, for a two hop delivery with the two-hop route of $< \lambda(p), n_1, \psi(p) >$, the probability is

$$\rho_p^{<\lambda(p),n_1,\psi(p)>} = \varepsilon_{\lambda(p),n_1} \times \varepsilon_{n_1,\psi(p)}. \tag{35}$$

Thus, the probability for packet being delivered in two hops is

$$\begin{aligned} \rho_p^2 &= 1 - \prod\nolimits_{n\in N, n\neq\lambda(p),\psi(p)} (1 - \rho_p^{<\lambda(p),n,\psi(p)>}) \\ &= 1 - \prod\nolimits_{n\in N, n\neq\lambda(p),\psi(p)} (1 - \varepsilon_{\lambda(p),n} \times \varepsilon_{n,\psi(p)}). \end{aligned} \tag{36}$$

The delivery probability of the packet with the h-hop route $< \lambda(p), n_1, \ldots, n_{n-1}$ $\psi(p) >$ is

$$\rho_p^{<\lambda(p),n_1,..,n_{h-1},\psi(p)>} = \varepsilon_{\lambda(p),n} \times \prod_{i=1}^{h-2} \varepsilon_{n_i,n_{i+1}} \times \varepsilon_{n_{h-1},\psi(p)}. \tag{37}$$

Then the set of all h-hop routes (denoted by Ξ) which starts from λ and ends at ψ is denoted by

$$\Xi^h(\lambda, \psi) = \{< \lambda, n_1, .., n_{h-1}, \psi > | \, n_i \in N; n_i \neq \lambda, \psi\}. \tag{38}$$

Thus, the h-hop delivery probability is calculated by

$$\rho_p^h = 1 - \prod_{r \in \Xi^h(\lambda(p),\psi(p))} (1 - \rho_p^r). \tag{39}$$

Therefore, the total delivery probability is

$$\rho_p = 1 - \prod_{0<h<H} 1 - \rho_p^h \quad . \tag{40}$$

Constant H acts as the hop limit.

From the previous analysis, we can find that encounter probability is the key to computing the overall delivery probability of a packet. Since it is impossible to know future movements of vehicles, the knowledge of encounter probability of each pair of vehicles is not immediately available.

Fortunately, we have observed that there is strong spatiotemporal regularity with vehicle mobility. Based on this observation, we present mobility patterns to characterize this regularity. With the mobility pattern, we are enabled to predict the trajectory of a vehicle. With trajectories of vehicles, we can effectively compute the encounter probability of two vehicles.

We should stress that the encounter probability computed based on this method is more accurate than those derived from simple patterns, such as inter-meeting times and spatial distribution. Our prediction effectively makes use of both the current state information and the historical information of vehicle movement. In addition, the historical information is purely based on individual vehicles. This overcomes the problem with traditional methods that require historical information about any pair of vehicles, which introduces additional overhead.

The global algorithm is described as follows. According to Sect. 4.2, each vehicle which has a transition matrix obtained from historical data can calculate its future trajectory bundle using its current position. With the trajectory bundles, the encounter probability of every two vehicles is known from (27). Then, the delivery probability of each packet in the network through each possible path can be calculated by (28) and (40). With these probabilities, the global algorithm solves an optimization problem to determine which the best next hop for each packet is. The algorithm repeatedly schedules the delivery until all the packets are delivered.

5.2 Distributed Algorithm

The global algorithm requires the complete knowledge of all packets and vehicles. More specifically, for each packet, the knowledge of its source and destination, and the current position must be known. For each vehicle, the knowledge of its position and the mobility pattern must be known. These pieces of knowledge are not available in a distributed setting. Thus, we design a practical, distributed algorithm with which a vehicle requires only limited and localized knowledge.

The distributed algorithm consists of two fundamental building blocks. In the first block, the new objective for each individual vehicle is designed, since it is impractical for individual vehicles to compute the global objective. In the second block, we define the metadata for vehicles to exchange with each other at meetings.

In the distributed setting, the knowledge of nodes and packets is usually incomplete. Therefore, a global optimization (32) is hard for it is impossible to compute the global metric Θ and the current metric $\Delta\Theta$ (defined in (11), (12), and (30)). In this case, we have to design a new metric based on which an individual vehicle makes routing decisions. Let the incomplete set of nodes and packets be denoted by and , respectively. Then, the new objective becomes

$$\max \sum\nolimits_{p\in\Phi\prime} \Delta\Theta_{p,N',t}. \tag{41}$$

$\Delta\Theta_{p,N',t}$ becomes the new metric of packet p at time t, which is local and can be computed by individual vehicles for each packet. When making routing decisions, a node maximizes the sum of local metrics of all the packets it knows.

It would be better for a node to have more knowledge about nodes and packets. Since it is difficult for a node to have the complete knowledge, we design a distributed protocol for the nodes to exchanging information when they meet each other. By this way, the knowledge can be propagated throughout the network.

For exchanging information, we define metadata, which include two parts of information. The first part is about the mobility pattern and the most update position of each vehicle (time stamped). The second part is about the packets that the node carries. Note that for a relatively stable set of vehicles, the mobility pattern reflects the regularity of a vehicle's mobility. Thus, it is relatively stable and therefore it is no need to update the mobility patterns frequently.

As a distributed algorithm, each vehicle executes the routing algorithm independently. For each vehicle, a routine procedure is invoked each time it finds a new communication neighbor. For neighbor discovery, it is required that every vehicle periodically broadcasts hello messages so that other vehicles can discover it when it enters their communication ranges.

In the following we describe the procedure, supposing that vehicle n finds another vehicle, n', entering its communication range. The Pseudo code description of this procedure is shown in Fig. 3.

First, node n updates its neighbor set (Λ_n). Then, node n and n' exchange their metadata. The metadata of a node, i, denoted by $\mathcal{M}_i$ include two metadata sets, $\mathcal{P}_i$

Procedure 1 $(n, n', \Lambda_n, \mathcal{M}_n, \mathcal{M}_{n'}, t)$
Variables
n: the node; n': new neighbor; $\mathcal{M}_n$, $\mathcal{M}_{n'}$: metadata;
Φ_n: packets of node n; Λ_n: neighbors of n;
t: current time; N: total nodes
1. $\Lambda_n = \Lambda_n \cup \{n'\}$,
2. n exchanges metadata with n': $n \leftarrow \mathcal{M}_{n'}$, $n' \leftarrow \mathcal{M}_n$
3. n calculates $\Delta\Phi_{n'} = \Phi_n - \Phi_{n'}$ from $\mathcal{M}_n, \mathcal{M}_{n'}$
4. For every packet $p \in \Delta\Phi_{n'}$, calculate metric $\Delta\Theta_{p,N,t}(n') = \rho_p\ n' - \rho_p(n)$
5. Sort all packets $p \in \Delta\Phi_i$ according to metric $\Delta\Theta_{p,N,t}(i)$ for all $i \in \Lambda_n$ in the decreasing order
6. Transmit the packets in this order

Fig. 3 Pseudo code of the procedure of the distributed algorithm, which is executed upon each contact with a vehicle

and $\mathcal{V}_i$. Set $\mathcal{P}_i$ contains the metadata about the packets that node carries, including identification and source-destination pair. Set $\mathcal{V}_i$ contains the metadata about the vehicles known to node i, including ID, most update position and mobility pattern.

Next, it recalculates the metrics for all the packets it carries, and sort the packets respect to the new metrics in the decreasing order. The packets will then be transferred in the sorted order. Note that if the node has already been transferring a packet, the transmission of this packet is not interrupted. After its completion, the packets shall be transmitted according to the new order. By ordering the packets, we are essentially doing the link scheduling in a distributed way. However, before a packet transfer can be started, a vehicle has to follow media access control to avoid potential collisions.

As packets are transferred between vehicles, the packet set, $\mathcal{P}_i$, is updated whenever the vehicle receives a new packet from its neighbors.

6 Simulation Study

We evaluate our algorithms with the performance metric of delivery ratio, delay, cost and efficiency. These metrics have been defined in Sect. 2. We compare our algorithms with several other algorithms, which is to be introduced shortly.

The simulations are conducted with the three datasets of real GPS vehicular traces. One dataset includes the vehicular traces of more than 4,000 taxis collected in Shanghai, the second dataset is the traces of more than 2,000 buses in Shanghai and the third dataset consists of vehicular traces of more than 12,000 taxis collected in Shenzhen. The Shanghai dataset covers a duration of two years and the Shenzhen dataset a duration of one month. The whole urban area of Shanghai is 133 km in length and 69 km in width, and that of Shenzhen is about 27 km in length and 97 km in width. Table 1 shows the summary of the three trace datasets.

The whole space is divided into grids, and the grid size is three kilometers by default and will be varied in the impact study of grid size. The communication range

Table 1 Summary of trace datasets

Trace	Shanghai taxi	Shanghai bus	Shenzhen taxi
# of nodes	4,000	2,000	12,000
Contacts (per hour)	352	891	425
Granularity (second)	60	60	60
Duration (hour)	17,280	17,280	720
Range (km × km)	133 × 69	133 × 69	27 × 97

is 300 m. We consider link interfaces, and each node can communicate with one neighbor at any time. We randomly select a subset of 400 taxis or buses from the complete trace for simulations.

For each packet, we randomly select its source and destination. The packets are injected at different times. Every packet has the same size and priority. The maximum hop and the maximum time-to-live (TTL) of each packet is set to 20–2 h, respectively. The number of packets is varied from 100 to 800 to study different loads of the network.

The order of Markov chains K is set to two. From conditional entropy analysis, we have already found that a higher order beyond two gives little reduction in uncertainty. To build the mobility pattern for each vehicle, its vehicular trace of the past 15 days is used. Each data point is the average over five different runs of trace-driven simulations.

We compare our routing algorithms with the following algorithms

- **Flooding** [17], also known as epidemic routing, is a simple algorithm. Each node forwards all the packets it carries to any node it meets. This algorithm provides an upper bound on delivery ratio and a lower bound on delivery delay. It introduces very high cost, which is the major defect.
- **P-Random** [15] is an opportunistic routing algorithm, which randomly decides whether to forward a packet to another node. In simulations, the probability is set to 0.4. This algorithm represents the algorithms without predictions.
- **MobySpace** [9] is a routing algorithm for delay tolerant networks. It obtains the probability of each node residing in each possible location by analyzing the historical trace data. It assumes that every node has the full knowledge of such probabilities of all vehicles. It then computes the contact probability of every two vehicles based on the probability distribution, and then routing decisions can be made.
- **Max-Contribution** [11] is a routing algorithm using a simple mobility pattern that only considers the inter-meeting times of exponential distribution. This pattern makes prediction independent of the current location of a vehicle.

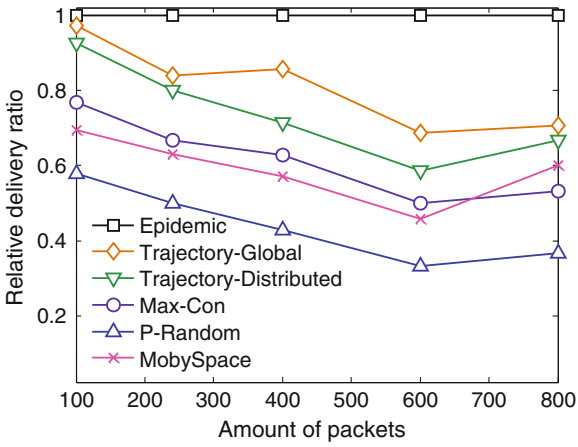


Fig. 4 Delivery ratio versus amount of packets, with Shanghai taxi dataset

6.1 Performance Results

We present performance results of our algorithms compared against other algorithms. First, we show the results using the Shanghai dataset of real taxi traces. To characterize the mobility pattern, the vehicular trace of the recent 15 days is used.

We vary the amount of packets and compare the six algorithms in terms of delivery ratio, average delay and total cost under different loads of network. For a given setting of amount of packets, the same set of packets and the set of nodes are used for all the six algorithms.

Since the delivery ratio is largely affected by the simulated time length, we use the metric of relative delivery ratio instead of bare delivery ratio, which is the delivery ratio of each algorithm normalized by that of Flooding. In Fig. 4, the performance of the six algorithms in terms of relative delivery ratio is shown. We can see that our algorithms perform better than P-Random, MobySpace and Max-contribution. Among the six algorithms, Flooding performs the best, as expected. In general, the algorithms that use predictions produce better delivery ratios than the algorithms that make no predictions. Our algorithms are better than MobySpace and Max-Contribution because our algorithms use not only the historical traces information but also the current state and the previous states. Predicted trajectories of vehicles help to find better routing paths. The global algorithm is better than the distributed one since it has the global, complete network information. We can also find that when the amount of packets increases, the overall delivery ratios of all algorithms decrease. The reason is that the overall capacity of the network is limited. By injecting more packets, the packets may compete for network resources and as a result, fewer packets can be delivered in the end.

In Fig. 5, average delay against amount of packets is plotted. Our algorithms have a lower delay than P-Random, MobySpace and Max-Contribution. As expected, Flooding has the smallest delay. The average delays of our algorithms are slightly

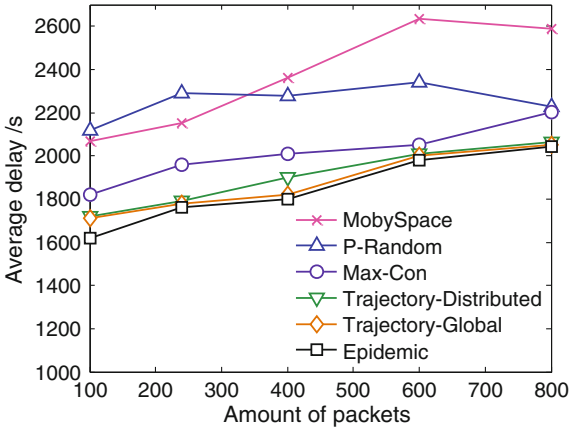


Fig. 5 Average delay versus amount of packets, with Shanghai taxi dataset

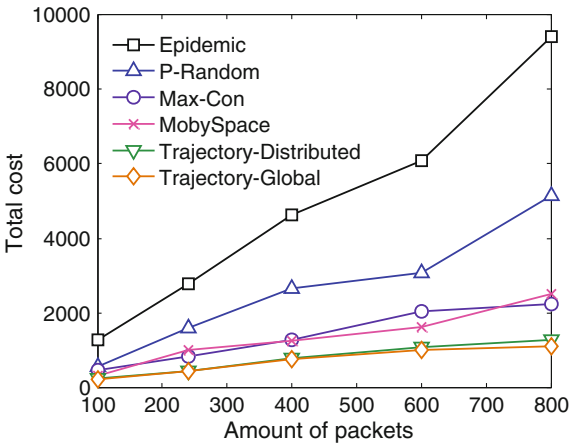


Fig. 6 Total cost versus amount of packets, with Shanghai taxi dataset

larger than that of flooding. This performance gain of our algorithms is mainly due to the fact that routing paths with high delivery probabilities usually lead to shorter delays. Since our algorithms can select routing paths of high delivery probabilities, the resultant delay is low.

Figure 6 shows the costs of the six algorithms. We can find that our algorithms have lower costs, better than all the other algorithms. The main reason is that by effectively predicting the trajectories of vehicles, we only consider the paths that lead to eventual delivery with high probability. Therefore, unnecessary packet transfers are greatly reduced.

In Fig. 7, we compare the efficiencies of the six algorithms. We can see that our algorithms have higher efficiency, better than all the other algorithms. Flooding and P-Random have a similar efficiency and are much worse than the rest four. This is due to the blindness of Flooding and P-Random when they are making transfer decisions. Max-Contribution and MobySpace have larger efficiency than Flooding

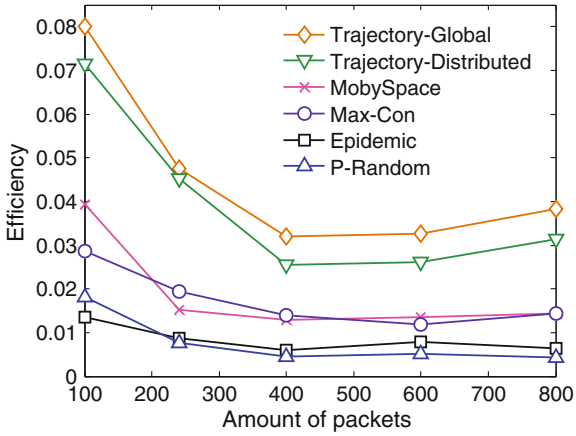


Fig. 7 Efficiency versus amount of packets, with Shanghai taxi dataset

and P-random, but have lower efficiency than our algorithms, since Max-Contribution and MobySpace merely uses a simple mobility pattern of inter-meeting time.

6.2 Impact of Grid Size

We study the impact of grid size. According to the design of the algorithms, two vehicles residing in the same grid are considered to encounter with each other, but they may be unable to communicate in reality. Meanwhile, a larger grid size results in a lower algorithm complexity. The simulation is performed with the Shanghai taxi dataset with the Shanghai bus dataset.

In Fig. 8, the delivery ratios of the two algorithms are shown for different grid sizes. We find that when the grid size increases, the relative delivery ratio of the two algorithms increases almost linearly. This confirms that a larger grid size makes the algorithms more aggressive, thus injecting more packets into the network. As a result, the delivery ratio becomes higher.

In Fig. 9, we show the performance of delivery delay of the two algorithms when the grid size is varied from 500 to 6,000. We can see that when the grid size increases, the average delays of both the algorithms slightly decrease Shanghai Bus dataset, but increase for Shanghai taxi dataset. Buses have much stronger regularity than taxis. For Shanghai bus dataset, a larger gird size leads to a more aggressive forwarding strategy and helps find shorter routing paths. For Shanghai taxi dataset, however, a more aggressive forwarding strategy size does not lead to short routing paths. The main reason is that predictions for taxis are less effective than for buses and the more aggressive forwarding results in less effective utilization of the limited forwarding opportunities in vehicular ad hoc networks.

In Fig. 10, we look at the performance of total cost of the two algorithms when the grid size varies. It is apparent from the result that the costs of the two algorithms

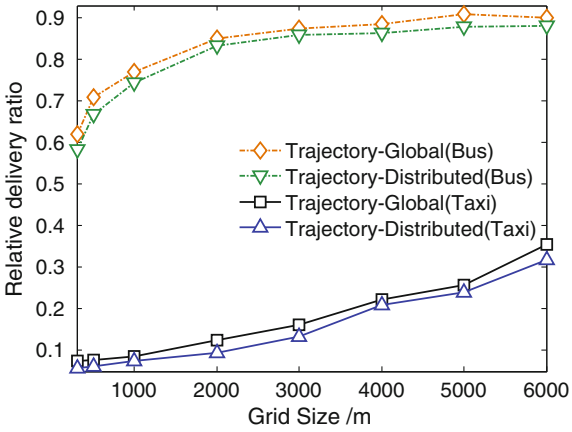


Fig. 8 Delivery ratio versus grid size

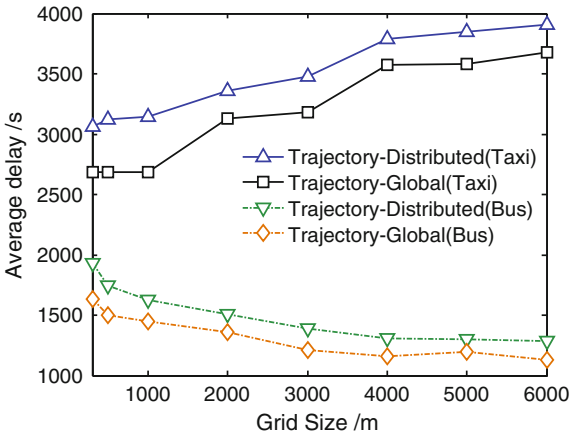


Fig. 9 Average delay versus grid size

quickly increase as the grid size becomes larger. This is easy to understand since with a larger grid size, the algorithms intend to forward more packets and thus inject more packets into the network. Thus, a higher cost is observed when the grid size is larger.

The performance of efficiency for the two algorithms for different grid sizes is shown in Fig. 11. The performance results match our expectation that the efficiency of the algorithms quickly decreases as the grid size becomes larger. This results from the fact that a larger grid size allows the algorithms to make more aggressive data forwarding.

In summary, the grid size controls the tradeoff between delivery ratio and cost (and also efficiency). And importantly, although a grid size introduces higher cost, the delivery delay may not be decreased.

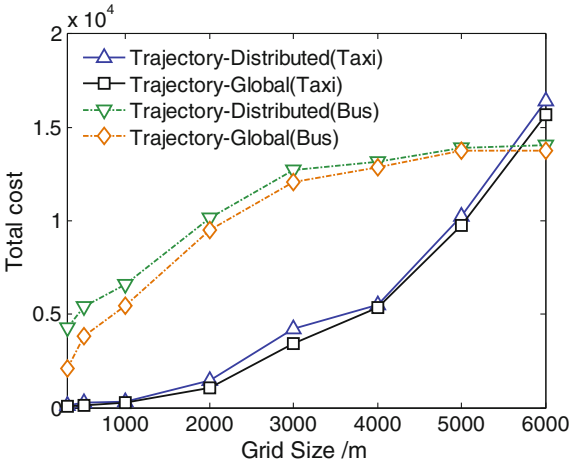


Fig. 10 Total cost versus grid size

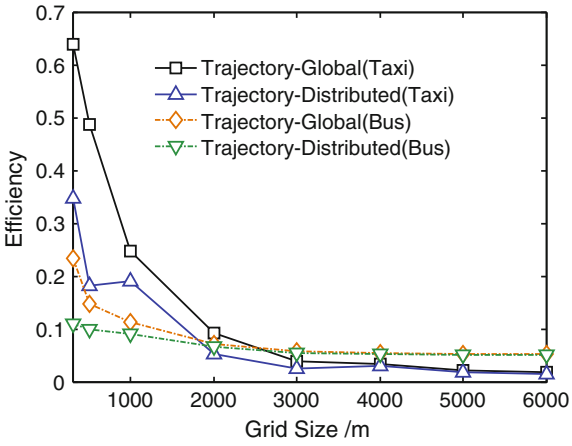


Fig. 11 Efficiency versus grid size

6.3 Impact of Historical Traces

The impact of the use of historical traces is studied. To this end, we vary the length of the historical trace that is used for building the mobility pattern of a given vehicular node. The length is varied from 2 to 16 days. To offset the effect of grid size, we study the performance under two configurations of grid size, 500 and 2,000 m. The simulation is performed with the Shanghai taxi dataset.

In Fig. 12, the performance of relative delivery ration of the trajectory based algorithms is shown for different lengths of historical trace. We can see that the relative delivery ratio increases quickly when a longer length of historical trace is used for both the global and the distributed algorithms. This strongly demonstrates that a longer trace builds a better mobility model and is beneficial to prediction of

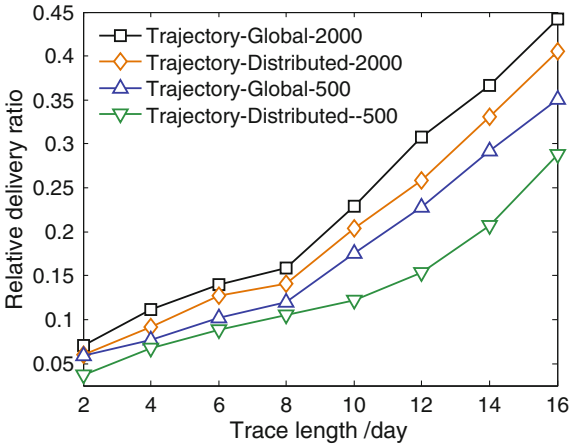


Fig. 12 Delivery ratio versus length of historical trace

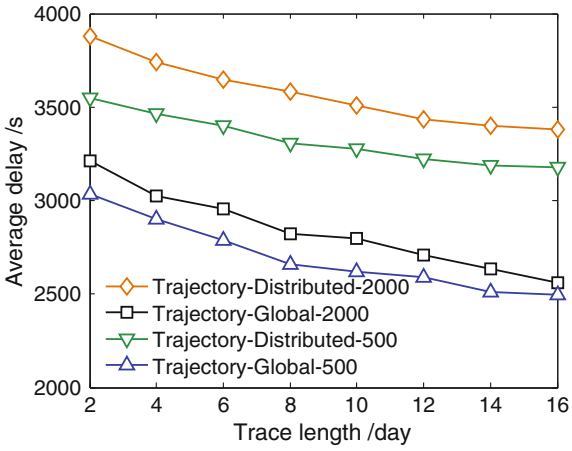


Fig. 13 Average delay versus length of historical trace

vehicular trajectory. This makes the routing process more effective by making use of the predicted trajectory. In addition, we find that when the grid size is 2,000 m, the delivery ratio performance becomes better, as observed previously in the study on the impact of grid size.

In Fig. 13, we show the performance of average delay of the two algorithms for different lengths of vehicular trace. We find that when the length of vehicular trace increases, the average delay drops slightly for both the algorithms. This shows that a longer historical trace is also beneficial to reduction of delivery delay.

Figure 14 shows the performance of the two algorithms in terms of total cost for different lengths of historical vehicular trace. The results show that the cost for each of the two algorithms also increases when the length of historical trace becomes larger. However, it should be noted that the increase in cost when the grid size is 500 m is much slower than the one when the grid size is 2,000 m. Thus, we can

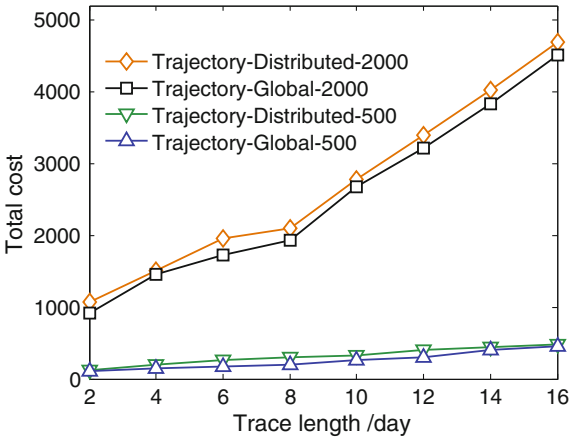


Fig. 14 Total cost versus length of historical trace

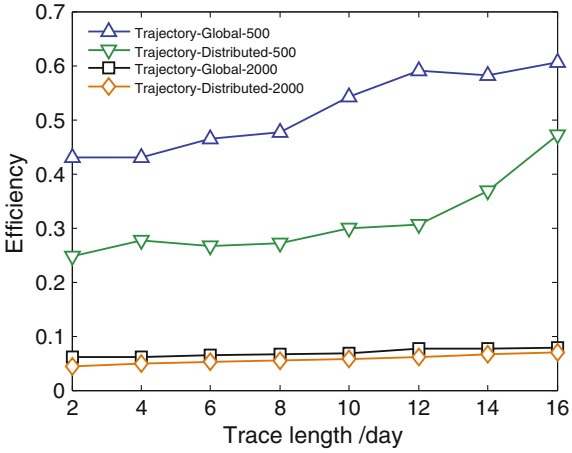


Fig. 15 Efficiency versus length of historical trace

conclude that when a small grid size, e.g., 500 m, is used, a longer historical trace only introduces a modest increase in cost. But when a large grid size is used, the cost rises quickly as the length of historical trace increases.

In Fig. 15, we show the performance of efficiency of the two algorithms for different lengths of historical trace. We can see that as the length of historical trace increases from 2 to 16 days, the efficiency of each algorithm slowly increases. In addition, the efficiency with a grid size of 500 m is much higher than that with a grid size of 2,000 m. And, the efficiency with a grid size of 2,000 m almost does not change as the length of historical trace increases.

In summary, a longer historical trace helps improve the performance of the trajectory based algorithms.

7 Conclusion

Vehicular ad hoc networks are a new type of mobile ad hoc networks. Efficient message routing is of significant importance to applications of vehicular ad hoc networks. However, different from traditional mobile ad hoc networks, existing routing algorithms for mobile ad hoc network cannot directly be used. Although routing of vehicular ad hoc networks has been studied, existing work either assumes the availability of future vehicular traces, e.g., through navigation systems, or fails to make effective use of the vehicular traces. By developing multiple order Markov chains, we predict future vehicle trajectories. The proposed trajectory based algorithms take full advantage of predicted probabilistic vehicle trajectories. Performance results verify that our algorithm outperforms other algorithms. This demonstrates that predicted trajectories do help improve message routing performance in vehicular ad hoc networks.

References

1. Bai F., Stancil, D.D., Krishnan, H.: Toward understanding characteristics of dedicated short range communications (DSRC) from a perspective of vehicular network engineers. Presented at the ACM MOBICOM, (2010)
2. Chisalita, L. Shahmehri, N.: A peer-to-peer approach to vehicular communication for the support of traffic safety applications. Presented at the The 5th IEEE Conference on Intelligent Transportation Systems (2002)
3. Li, Z., Zhu, Y., Zhu, H., Li, M.: Compressive sensing approach to urban traffic sensing. Presented at the IEEE ICDCS (2011)
4. Zhu, H., Zhu, Y., Li, M., Ni, L.M.: SEER: metropolitan-scale traffic perception based on lossy sensory data. In: Proceeding of the IEEE INFOCOM (2009)
5. Leontiadis, I., Mascolo, C.: Geopps: geographical opportunistic routing for vehicular networks. In: Proceedings of the IEEE WoWMoM (2007)
6. Marfia, G, Roccetti, M., Palazzi, C.E., Amoroso A.: Efficient vehicle-to-pedestrian exchange of medical data: an empirical model with preliminary results. In: Proceedings of the ACM MobiHoc Workshop on Pervasive Wireless (2011)
7. Leontiadis, I., Costa, P., Mascolo, C.: Extending access point connectivity through opportunistic routing in vehicular networks. In: Proceedings of the IEEE INFOCOM (2010)
8. Balasubramanian, A., Levine, B., Venkataramani, A.: DTN routing as a resource allocation problem. In: ACM SIGCOMM (2007)
9. Leguay, J., Friedman, T., Conan, V.: DTN routing in a mobility pattern space. In: proceedings of the ACM SIGCOMM Workshop on Delay-tolerant Networking (2005)
10. Krumm, J., Horvitz, E.: Predestination: where do you want to go today? Computer **40**, 105–107 (April 2007)
11. Lee, K., Yi, Y., Jeong, J., Won, H., Rhee, I., Chong S.: Max-contribution: on optimal resource allocation in delay tolerant networks. In: IEEE INFOCOM (2010)
12. Chen, Z., Kung, H., Vlah, D.: Ad hoc relay wireless networks over moving vehicles on highways. In: ACM Mobihoc (2001) p. 250
13. Bychkovsky, V., Hull, B., Miu, A., Balakrishnan, H., Madden, S.: A measurement study of vehicular internet access using in situ Wi-Fi networks. In: Proceedings of the ACM MOBICOM (2006)

14. Burgess, J., Gallagher, B., Jensen, D., Levine, B.N.: Maxprop: routing for vehicle-based disruption-tolerant networks. In: Proceedings of the IEEE INFOCOM (2006)
15. Jain, S., Fall, K., Patra, R.: Routing in a delay tolerant network. In: Proceedings of the ACM SIGCOMM (2004)
16. Yi, Y., Proutière, A., Chiang, M.: Complexity in wireless scheduling: impact and tradeoffs. In: Proceedings of the ACM MobiHoc (2008)
17. Ramanathan, R., Hansen, R. Basu, P., Rosales-Hain, R., Krishnan, R.: Prioritized epidemic routing for opportunistic networks. In: ACM MobiSys workshop on Mobile Opportunistic Networks (MobiOpp) (2007), p. 66

Internet of Things Low-Cost Long-Term Environmental Monitoring with Reusable Wireless Sensor Network Platform

Mihai T. Lazarescu

Abstract The Internet of Things (IoT) provides a virtual view, via the Internet Protocol, to a huge variety of real life objects, ranging from a car, to a teacup, to a building, to trees in a forest. Its appeal is the ubiquitous generalized access to the status and location of any "thing" we may be interested in. Wireless sensor networks (WSN) are well suited for long-term environmental data acquisition for IoT representation. Application requirements, such as low cost, high number of sensors, fast deployment, long lifetime, low maintenance and high quality of service need to be considered from the beginning to achieve a WSN platform optimized for a range of long-term environmental monitoring IoT applications. Also important, the low-effort platform reuse should be considered through all design levels, for the platform as a whole and for all its components.

1 Introduction

More than a decade ago, the Internet of Things (IoT) paradigm was coined [1] in which computers would be able to access data about objects and environment without human interaction. This aimed to complement human-entered data, seen as a limiting factor to acquisition accuracy, pervasiveness and cost.

From the start, two technologies were considered key enablers for the paradigm: the radio-frequency identification (RFID) and the wireless sensor networks (WSN). While the former is well established for low-cost identification and tracking, WSNs bring IoT applications richer capabilities for both sensing and actuation. WSN solutions already cover a very broad range of applications, and research and technology advances continuously expand their application field.

Mihai T. Lazarescu (✉)
Politecnico di Torino, Turin, Italy
e-mail: mihai.lazarescu@polito.it

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_7,

But the sheer diversity of WSN applications makes increasingly difficult to define "typical" requirements for their hardware and software [2]. Generic WSN components often have to be adapted to specific application requirements and environmental conditions [3]. Such ad hoc changes tend to adversely impact the overall solution complexity, cost, reliability, and maintenance. In turn, these curtail WSN adoption, including their use in IoT applications [4].

Reusable WSN platforms are receiving a growing interest since they attempt to address these issues. These platforms are typically optimized by leveraging knowledge of the target class of applications (e.g., domain, WSN devices, phenomena of interest) to improve key WSN application parameters, such as cost, productivity, reliability, interoperability, maintenance.

Among the IoT application domains, the environmental/earth monitoring receives a growing interest as environmental technology becomes a key field of sustainable growth worldwide. WSN environmental monitoring includes both indoor and outdoor applications. The later can fall in the city deployment category (e.g., for traffic, lighting or pollution monitoring) or the open nature category (e.g., chemical hazard, earthquake and flooding detection, volcano and habitat monitoring, weather forecasting, precision agriculture).

The open nature environmental monitoring is especially challenging technically and logistically because of, among others, the typically harsh operating conditions and difficulty and cost of physical access to the field for deployment and maintenance. Moreover, the typical IoT applications requirements for low cost, high service availability and low maintenance add further challenges.

Generic WSN platforms can be used with good results in a broad class of IoT environmental monitoring applications [5]. However, many IoT applications (e.g., those operating for long time in open nature) may have stringent requirements, such as very low cost, large number of nodes, long unattended service time, ease of deployment, low maintenance, which make these generic WSN platforms less suited.

To be cost-effective, the sensor nodes often operate on very restricted energy reserves. Premature energy depletion can severely limit the network service [6–9] and needs to be addressed considering the IoT application requirements for cost, deployment, maintenance, and service availability. These become even more important for monitoring applications in extreme climatic environments, such as glaciers, permafrosts or volcanoes [3, 10–14]. The understanding of such environments can considerably benefit from continuous long-term monitoring, but their conditions emphasize the issues of node energy management, mechanical and communication hardening, size, weight, and deployment procedures.

Open nature deployments [15–19] and communication protocol developments and experiments [9, 20] show that WSN optimization for reliable operation is often time-consuming and costly. Reusable hardware and software platforms [21–26] are available to better satisfy IoT applications requirements for long-term, low-cost and reliable service. These include flexible Internet-enabled servers [27–29] for collection, storage and processing of field data.

In the following we will discuss and define the typical requirements for a class of environmental monitoring applications. Then we will explore possible solutions and

practical realizations of a full-custom, reusable WSN platform suitable for low-cost long-term IoT environmental monitoring applications.

For design consistency will be defined a real-life representative application for the domain. Its requirements for low-cost, fast-deployment of large number of sensors and reliable and long unattended service will drive the decisions at all design levels. Several trade-offs between platform features and specifications will be identified and analyzed to guide the design. Their suitability will be then checked against the experimental results. Moreover, the development methodology presented can be reused for platform design for other application domains, or evolutions of this platform.

The most important contributions to WSN development and use can be summarized as: (1) detailed specifications for a demanding WSN application for long-term environmental monitoring that can be used to analyze the optimality of novel WSN solutions, (2) specifications, design considerations, and experimental results for platform components that suit the typical IoT application requirements of low cost, high reliability, and long service time, (3) specifications and design considerations for platform reusability for a wide range of distributed event-based environmental monitoring applications, and (4) a fast and configuration-free field deployment procedure suitable for large scale IoT application deployments.

2 Platform Role in WSN Application Development

Since the early days of the IoT paradigm, the rich features of the WSNs placed them among the best candidates to bridge the physical and digital worlds. With small dimensions, on-board processing power, long unattended exploitation time, and ease of deployment due to multihop failure-resilient wireless communications, they can play this role in the most unobtrusive ways, as a natural evolution of the *smart dust* concept [30].

However, the WSN applications are yet to achieve their full potential. The multiple and variated technologies found in most WSN platforms, as well as the increasing diversity of application domains and specifications still represent a barrier to mass adoption due to high cost of the required skills and development time.

Figure 1 shows the typical flow of value in the development of a WSN application. At the top of the WSN value chain are the WSN application *end users*. They are providing the application field, the conditions to monitor, and the user interface on which to display the field status and alerts.

The *system integrators* leverage on the existing WSN technologies to satisfy the application requirements. Often under time and cost pressure, they ideally look after complete WSN platforms which can fully cover, end-to-end, the WSN applications needs.

However important, developing and maintaining a complete, flexible, and reliable WSN platform is costly and involves a broad range of expertise, advanced web and UI design, low level firmware and IDE, and code generation techniques for high

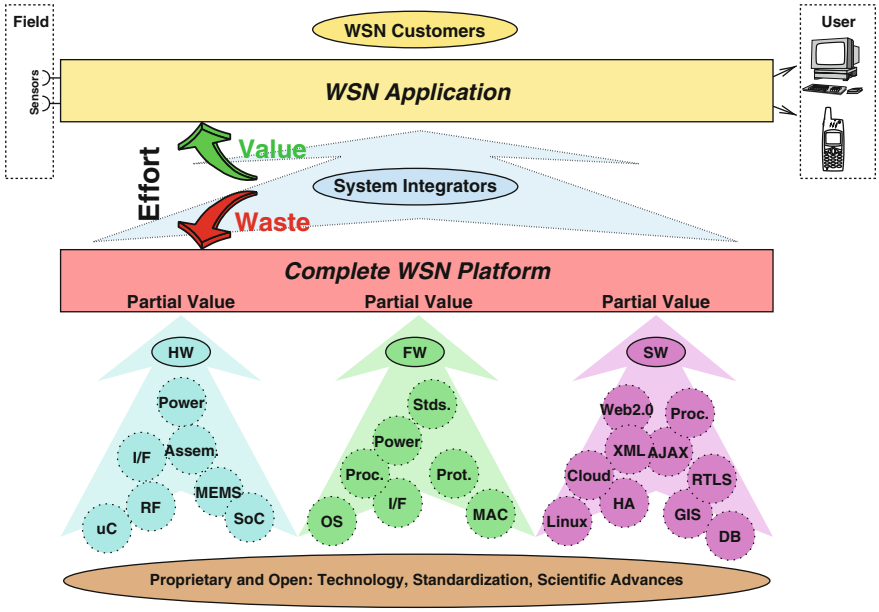


Fig. 1 Main value flow in WSN application developments

level application definition. Quality assurance is also very important, since overall unreliability perception is still a limiting factor to WSN wider adoption.

The commercial WSN development platforms are usually provided by the hardware vendors for their devices. These focus typically on exploiting the features of the vendor's hardware, often being hardware-centric rather than WSN application-centric. As such, they may require significant extension or adaptation to properly cover a broader range of applications. They also tend to significantly lag the WSN field state of the art on directions different than the progress of one specific hardware producer.

Most existing embedded (TinyOS, freeRTOS, Contiki) or server (GSN) open projects are largely vendor- and hardware-independent, but they also do not address the complete WSN application needs, requiring significant adaptation and integration for an effective use.

Moreover, the up-front integration effort into existing development flows and the proprietary solutions lock-in to vendor's hardware often lead to significant hold-up problems hampering usage and business potential. All these aspects finally translate into wasted system integrators' effort, missed or underperformed delivery opportunities, higher development costs and risks. For new players, they add to existing entry and differentiation barriers, effectively limiting the adoption of new WSN solutions.

A complete, sensor-to-user interface WSN platform would allow the system integrators to direct most of their resources to satisfying their users' needs. Ideally,

a WSN platform would be flexible, reliable, hardware independent (no lock-in), and constantly updated to WSN field state of the art.

However, the cost to build and maintain such a platform is high and requires complex interdisciplinary skills that can hardly be afforded by the system integrators, its main beneficiaries. Although the underlying technology providers can cover well some specific segments, their focus is usually too narrow to successfully address a complete, end-to-end platform.

Nevertheless, the proprietary or open contributions of technology providers are essential for building a good quality, complete platform. For instance, the hardware support of the WSN platform is important for both the system integrators and the hardware vendors. Thus, as hardware ports add value to all parties, they can be sponsored by the system integrators, by the hardware vendors, or simply be developed out of genuine interest by the research or open source community for the most popular and trendy hardware.

In fact, university research and open source community already offer valuable elements for a complete WSN platform. On the firmware and software side there are development tools, communication protocols, embedded OSs, security features, and some standards implemented for the field devices, while the server side benefits from the advanced Linux, user interface, and web tools and techniques.

3 IoT Environmental Monitoring Requirements

WSN data acquisition for IoT environmental monitoring applications is challenging, especially for open nature fields. These may require large sensor numbers, low cost, high reliability, and long maintenance-free operation. At the same time, the nodes can be exposed to variable and extreme climatic conditions, the deployment field may be costly and difficult to reach, and the field devices weight, size, and ruggedness can matter, e.g., if they are transported in backpacks.

The well-known application of wildfire monitoring using in situ distributed temperature sensors sums up most of these requirements and conditions. In its simplest event-driven form, each sensor node performs periodic measurements of the surrounding air temperature, process them on-board to reduce energy consumption, and sends alerts to surveillance personnel if they exceed a threshold. Figure 2 shows a typical deployment pattern of the sensor nodes that achieves a good field coverage [31]. For a fast response time, the coverage of even small areas requires a large number of sensor nodes, making this application representative for cost, networking and deployment issues of the event-driven high-density IoT application class.

In the simplest star network topology, the sensor nodes connect directly to the gateways, and each gateway connects autonomously to the server. Ideally, a deterministic field deployment procedure ensures that each sensor node can communicate with more than one gateway to avoid the single points of failure.

This application can be part of all three WSN categories [22]: event-driven (as we have seen), time-driven (e.g., if the sensor nodes periodically send temperature

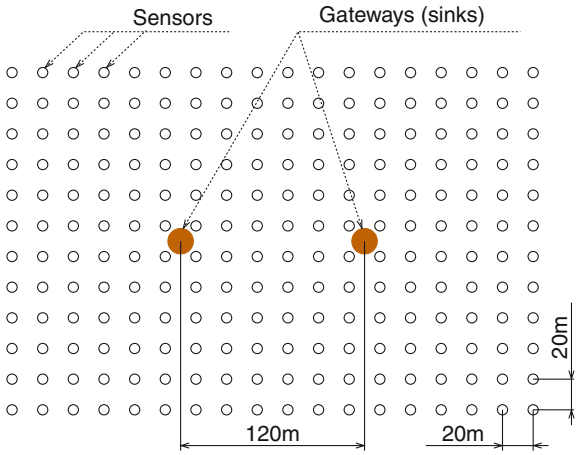


Fig. 2 Ideal WSN deployment for in situ wildfire detection applications

readings or statistics) and query-driven (e.g., if the current temperature can be requested by the operator). This means that the infrastructure that supports the operation of this application can be reused for a wide class of similar long-term environmental monitoring applications like:

- *water level* for lakes, streams, city sewages;
- *gas concentration in air* for cities, laboratories, deposits;
- *soil properties*, e.g., humidity;
- *inclination* for static structures, e.g., bridges, dams;
- *position changes* to monitor land slides, for example;
- *lighting conditions* either as part of a combined sensing or standalone, e.g., to detect intrusions in dark places;
- *infrared radiation* for fire, or detection of human or animal presence;
- *distance* for level or passage monitoring.

Since these and many related applications typically use fewer sensor nodes, they are less demanding on the communication channels (both in-field and with the server), and for sensor node energy and cost. Consequently, the in situ wildfire detection application can be used as reference for the design of a WSN platform optimized for IoT environmental monitoring and the platform should be easily reusable for a broad class of related applications.

Thus, the most significant requirements of a WSN platform for IoT long-term environmental monitoring can be defined as follows:

- low-cost, tiny sensor nodes with on-board processing, self-testing and error recovery capabilities;
- low-cost, small-size gateways (sinks) with self-testing, error recovery and remote update capabilities, supporting several types of long-range communication;
- adequate gateway hardware and software resources to support specific application needs (e.g., local transducers, and data storage and processing);

- detection of field events on-board the gateway to reduce network traffic and energy consumption;
- fast detection and relaying to field operators of field alerts;
- field communication protocol efficiently supporting:
  - from few sparse to a very large number of nodes;
  - low data traffic in small packets;
- fast and reliable field node deployment procedure;
- remote configuration and update of field nodes;
- high availability of service of field nodes and servers, reliable data communication and storage at all levels;
- node ruggedization for long-term environment exposure;
- extensible server architecture for easy adaptation to different IoT application requirements;
- multiple-access channels to server data for both human operators and automated processing;
- programmable multichannel alerts;
- automatic detection and report of WSN platform faults (e.g., faulty sensor nodes) within hours, up to a day;
- 3–10 years of maintenance-free service.

These requirements will be used in the following sections to define the specifications and analyze the design alternatives for the nodes, networking, deployment procedure and operation of a WSN platform suitable to support a significant set of IoT applications for environmental monitoring.

4 Platform Structure

The main purpose of a WSN for environmental monitoring for IoT is to gather, process, and convey field data and events to the users of the application (humans or computer systems).

The tiered structure of the used platform was introduced by one of the first long-term outdoor WSN experiments [15] (see Fig. 3) and it allows:

- a good functional separation of platform components for optimization according to application requirements;
- a cloud-based field data access to bridge the latency-energy trade-offs of the low power communication segments and the ubiquitous and fast access to field data for end users (either humans or IoT applications).

The *sensor nodes* are optimized for field data acquisition using on-board transducers, processing, and communication to gateways using short-range RF communications, either directly or through other nodes. The *gateways* process, store, and

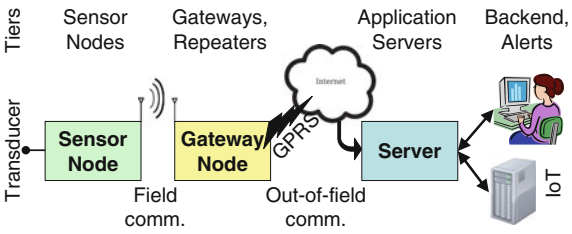


Fig. 3 Tiered structure of the WSN platform

periodically send the field data to the application server using long-range communication channels. The *application server* provides long-term data storage, and interfaces for data access and process by end users (either human or other applications).

Also, the platform should be flexible to allow the removal of any of its tiers to satisfy specific application needs. For instance, the transducers may be installed on the gateways for stream water level monitoring since the measurement points may be spaced too far apart for the sensor node short-range communication capabilities. The gateways may not be needed in other applications, such as seismic reflection geological surveys. In these applications, the sensor nodes may be required to connect directly to an on-site processing server. And when the gateways can communicate directly with the end user, e.g., by an audible alarm, the application server may be omitted.

Besides the functional elements described above, the platform often includes an *deployment device*. Its purpose is to assist the field operators when they are looking for a suitable installation place for the platform nodes, thus reducing the deployment time, cost and errors.

5 Design of the WSN Nodes

The specifications of the WSN platform defined in Sect. 3 will be refined to derive the specifications of the nodes of the WSN platform, to perform design space exploration, analysis of possible solutions, and to support most important node design decisions.

5.1 Design of the WSN Sensor Nodes

Since the IoT applications for environmental monitoring may require large numbers of sensor nodes (e.g., for the in situ distributed wildfire detection), the specifications of these nodes are very important for the overall application functionality, performance and cost.

For such applications, one of the most important requirements is the reduction of the cost of the sensor nodes. Also, for a low deployment and operation cost of

the application, the sensor nodes should provide a long maintenance-free service time and support a simple and reliable deployment procedure. The physical size and weight of the nodes is also important, for instance if they are transported in backpacks to the deployment site.

The energy source can influence several characteristics of node operation and perceived reliability. Batteries can provide a steady energy flow but limited in time and their replacement may be costly for remote areas and large number of nodes. Energy harvested from environment can be potentially endless, but is unpredictable in time. This may impact node and network operation, as well as the perceived reliability of the application. Also, these sources may have specific packaging and deployment requirements that may increase the overall application costs.

Considering all these, the nodes powered by batteries may reduce the application cost and improve the service reliability if their energy consumption allows using a small battery that lasts for the whole node lifetime.

The energy consumption of the sensor node has the following components:

- *RF communication* (data transfer, network maintenance, service);
- *processing* (transducer data, self-checks, RTC);
- *sensing* (transducer supply, calibration);
- *safety devices* (watchdog timer, brown-out detector);
- *power down*, the energy drawn by the node components in their lowest power consumption mode.

The last two (safety devices and power down) depend mostly on component selection and their drivers, while sensing, energy for RF communication, and processing depend mainly on the application and communication protocol [32].

One of the most important energy drains on WSN nodes is the RF communication and over the years have been explored various trade-offs between communication parameters. A large variety of solutions was proposed, optimized for the general communication requirements and patterns specific for classes of application domains [33, 34].

However, to optimize the cost/performance ratio for a specific application domain its communication requirements should be carefully considered for the selection and optimization of its communication protocol. For instance, the typical needs of the sensor nodes in event-driven environmental monitoring applications are:

- report alerts for field events as soon as they are detected;
- periodically report their health status and other statistics.

Messages in the former class need to be delivered timely and reliably to the server, but the latter can sustain higher latencies and packet loss without a significant impact on overall application reliability.

Moreover, the mandatory information flow for this application class is from sensor nodes to gateways and server, to forward the field data. The sensor nodes are not usually *required* to communicate directly with their peers in field, nor to *receive* communications from the gateways.

However, both these communication flows may be necessary to fulfill specific application requirements. For instance, the former allows to create more complex field network topologies that can add network redundancy to relax to some degree the deployment specifications and offer increased resilience to communication failures. The latter may be needed to remotely configure or reprogram the sensors during network exploitation.

The specifications of the reference application, of in situ distributed wildfire detection, are to minimize the sensor node cost and maximize its operational lifetime (see Sect. 3).

The energy reserves are very important for the sensor node lifetime, maintenance requirements, quality of service (QoS), and cost. To select the best suited for the application, a first decision to make is whether to use energy harvesting with limited on-board storage capacity or to use non-regenerative solutions (e.g., a primary battery).

Energy harvesting solutions require outside energy sources that can be harvested, for instance vibration, fluid flow, temperature gradients, light. However, it is difficult to estimate the cost and efficiency of these solutions in the general case. Although combined energy harvesting solutions [35] increase harvesting availability, an on-board energy storage should be sized based on average and peak power consumption of the sensor node, as well as on the expected maximum interruptions in external energy (e.g., variations or absence of environmental illumination). Moreover, from the average energy and the expected operation between maintenances can be obtained the primary battery capacity, thus its cost. This cost can be used as higher bound for the evaluation of energy harvesting-based solutions.

Besides the cost of the energy source, its physical dimensions, packaging requirements, and other ways it can affect sensor node cost or its suitability for the application should be evaluated. For instance, when considering light harvesting the requirements for the node package (e.g., to provide an impermeable transparent window) should be carefully weighed, as well as the sensor node placement and orientation in the field during deployment, to receive enough average light. These may increase the node cost and the deployment time, cost, and risk for errors.

Considering all these, the best way to supply large numbers of small, cheap and easy to deploy sensor nodes, as required by the reference application, is using a *small* internal primary battery that lasts for the entire expected service lifetime of the nodes. This solution can be cheaper than harvesting-based solutions [36] and the flow of energy is stable, improving the overall node service reliability.

Since the cost of batteries tends to increase proportionally with their capacity, this solution is cost-effective if it is small. This means that the average energy requirements of the node need to be kept as low as possible to ensure a depletion time of a small capacity battery compatible with the expected application lifetime (see Sect. 3).

Radio communication is usually among the highest energy drain on the WSN nodes [37]. If the application requires sensor nodes with bidirectional communication capabilities, these should be equipped with a transceiver. This can increase their cost and, typically, the energy consumption, which may lead to higher application

cost, reduced service time due to faster battery depletion or higher maintenance requirements for battery replacement.

In these cases, the receiver should operate on a very low duty cycle to reduce its idle energy consumption. A widely used technique is low power listening (LPL, that can lower the receiver duty cycle down to 1–3 % [38]). Lower duty cycles and improved channel utilization can be achieved using synchronized networks [26, 39]. However, these require the nodes to keep track of the time using a constantly running accurate oscillator, which adds to node cost and to idle energy consumption. Moreover, the oscillator thermal drift in extreme environmental conditions may require more frequent synchronization messages or even repeated energy-intensive network re-registrations if synchronization is lost [11, 26]. The protocol complexity uses more resources (e.g., program and data memory, and processing time), which further increase node cost and energy consumption.

Other mechanisms can be used to reduce the radio energy consumption. For instance, the transmission power can be lowered if the communication range decreases by using a mesh topology instead of a star topology. In this case, however, it should also be considered the impact on the node resources of the additional traffic due to peer message forwarding. Also, message routing in large mesh networks can lead to undesired dynamic effects, some difficult to foresee and avoid, like bottlenecks or instabilities [19, 20]. These may reduce the network service reliability and require additional analysis and maintenance, that can increase the costs.

A receiver on the sensor nodes can also help reducing packet collisions (e.g., by using clear channel assessment (CCA) techniques), thus lowering the wasted energy due to message retransmissions. In this case, higher reception error rates can also be accepted if they can be compensated by an automatic retransmission of lost messages. This also allows to shorten the messages (e.g., by increasing the symbol rate, reducing the preamble, removing the FEC), thus effectively reducing the energy spent per message.

However, we mentioned that the sensor nodes in environmental monitoring WSN applications are not generally *required* to sustain bidirectional communications and that the reference application is particularly sensitive to sensor node lifetime and cost. With these requirements in mind and the above considerations on node features, energy and cost, we will explore in the following whether the sensor node communication can be implemented most energy- and cost-effective using a transmit-only radio device.

If the sensor nodes have no radio receive capabilities, they cannot form mesh networks, thus they need to communicate directly with the receiver (gateway), in a star topology. Also, they cannot receive acknowledgments or prevent packet conflicts, thus the protocol has to include redundant retransmissions at the source to keep the message loss acceptable for the application. Moreover, the sensor nodes have to periodically communicate their status through heartbeat messages. Their absence marks very likely a node failure.

Note that there is a distinction between *packet* and *message*. A *message* is the data to be conveyed and fits into a packet. Due to packet loss, the *same message* can be

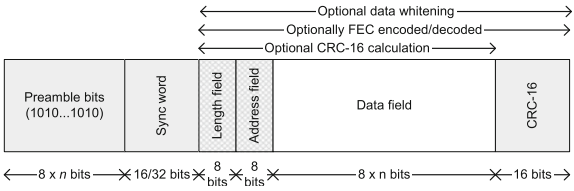


Fig. 4 Texas Instruments CC1150 packet format (*source* device data sheet)

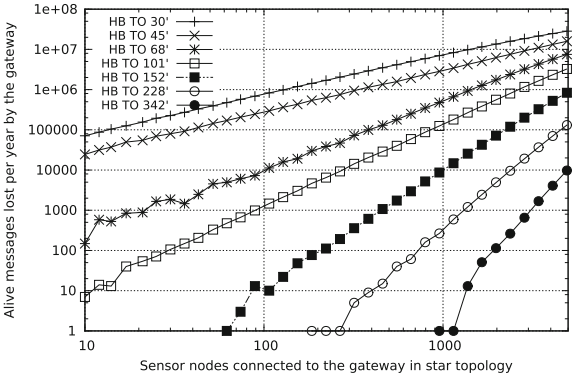


Fig. 5 Lost alive messages per year (280 ms packets, TX rate 1/h) for one gateway in star topology as function of node density and heartbeat timeout

repeatedly sent using *different packets*. The message is received if *at least* one packet carrying it is properly received and the receiver should discard message duplicates.

We will examine the redundancy necessary for such a network considering a popular off-the-shelf radio device and tuning the transmission parameters to optimize the RF propagation over long distances in forest conditions. The packet has a long duration for just a few bytes payload to minimize the receiver error rate (e.g., using 24 bytes preamble, 4 bytes sync word, 4 data bytes, 1200 Baud symbol rate, FEC and CRC for the packet structure of the Texas Instruments CC1150 device, see Fig. 4).

Figure 5 shows the simulation results, in these conditions, of the number of "alive" nodes lost by a gateway monitoring up to 5,000 sensor nodes in a star topology, over one year of operation. Each node sends one heartbeat *packet* per hour and the gateway considers that a node is missing if no heartbeat *message* is received within a given timeout (the plots are for timeouts from 30 minutes to almost 6 hours).

It can be seen that enough transmission redundancy can keep very low the rate of false node missing reports due to message loss. For instance, if the application can accept to detect missing (faulty) nodes with a delay of about 6 h, then the false report rate (report of normally operating nodes as missing) due to heartbeat message loss can be around 1/year for a field with about 1,000 sensor nodes. More generally, a message loss rate of about 1/year can be assumed for an RF space utilization, due to concurrent transmissions, of about 8 % ($\approx$ 1000 packets $\times$ 0.28 s/packet/3600 s) if the message is sent with a redundancy of 6 (the number of packets that repeat the same message).

It is worth noting that even with a low payload-to-packet size ratio (see the settings above), the sensor node radio operates with a *very low* duty cycle, of less than 0.01% (1 packet of 0.28 s sent every hour).

5.2 Gateway Node Design

The main role of the gateways in a WSN application is to collect, process, and forward to an Internet-connected server the field data received from the sensor nodes. They are fit with an in-field communication interface for sensor nodes and an out-of-field communication channel for the application server.

The long range communication, e.g., a GPRS modem (see Fig. 3) can be responsible for most energy consumption of the gateway. Although carefully optimized gateways can reach several years of operation using large battery packs [3], they are usually fit with energy harvesting devices (or mains power supply) to reduce their cost and maintenance requirements.

The gateway node shapes and sizes are typically less constrained by the applications than for sensor nodes. Nevertheless, a small form factor and adequate I/O resources generally help satisfy the application needs, especially those that require transducers be connected directly to gateway I/O.

According to considerations on sensor node optimization for the reference application from Sect. 5.1, the field communication protocol is typically chosen to optimize the sensor nodes. The impact of the field protocol on gateway resources (e.g., as processing, code size, data memory) is usually less important than for sensor nodes. Also, peer gateways may communicate on a different channel than that of the sensor nodes, to prevent its congestion. This is especially useful when the MAC of the sensor nodes cannot resend lost messages to improve communication reliability (e.g., when they have no receiving capabilities).

For star topology deployments in difficult propagation conditions it may be necessary to extend the reachability of the gateways using *repeater* nodes. Their basic operation is to forward to the gateways in range all sensor node messages they receive, so that the gateways can process them as if they were received directly from the sensor nodes.

In an event-driven application, the gateways reduce the communication with the server by processing the field data on-board to detect significant events. For instance, they can autonomously detect faulty nodes or analyze the data from several sensor nodes for events that cannot be detected at single sensor node level.

The gateways usually allow to remotely change their parameters, processing flows, or update their entire program [40–43] to improve their fit to (changing) application requirements. For a high quality of service over long periods of time it is also important to perform frequent self-checks and trigger error recovery mechanisms in case of anomalies. At the same time, they should also preserve as much as possible the data collected from the field across the error recovery procedures.

5.3 Deployment Device Design

The WSN deployment personnel typically make use of a handheld interactive device that assist them in the node deployment process. It can help with:

- the orientation in the field to find the designated node deployment position;
- assessing the suitability of a specific deployment location;
- identifying and configuring the nodes;
- recording the actual deployment data.

These are among the most time-consuming and error-prone parts of the deployment procedure and is important to provide the personnel with suitable means to speed up the process and avoid errors. All these are especially relevant for applications with a large number of nodes (as the reference application), whose quality and cost are particularly sensitive to deployment issues.

Once the desired field location is found, the device assists the operator assessing whether a node is functional and can properly operate in a specific installation spot. For a sensor node, this usually means assessing the suitability of its connection with peer nodes (for mesh networks) or with the gateways/repeaters. For a gateway node, it means assessing its connection with peers, the long range connection quality (e.g., GPRS coverage), and the energy harvesting quality (e.g., the proper illumination of its solar panels).

To assist in these operations, the deployment device should be able to communicate with the nodes in range and display the relevant data in a format easy to see and understand in open nature conditions and by personnel that may have limited training on WSN operation.

Besides these basic functions, the deployment device can also have localization capabilities (e.g., GPS) that can guide the field operators to the predefined node deployment positions and record the actual node deployment locations. It may also be able to connect to the application server over a long range communication channel to receive configuration and upload the deployment data.

6 Implementation of the WSN Devices

In the following are presented the most important implementation choices for all devices of the WSN platform. These are derived from the requirements in Sects. 3 and 5, that make them suitable for long-term environmental monitoring IoT applications.

The implementation of the WSN nodes considered especially the low cost requirement for all nodes, particularly for the sensor nodes of the reference application. Among the major factors contributing to application cost are: node components, assembly, packaging, size/weight (impact on deployment time and cost), and node robustness for long term deployments in open nature.

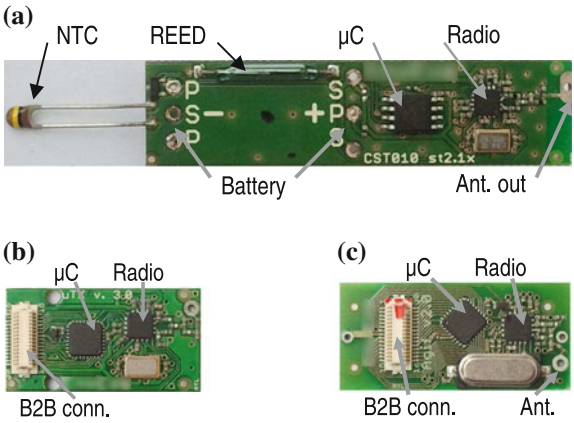


Fig. 6 PCB of sensor nodes for environmental monitoring: **a** For in situ wildfire detection, **b** For mostly analog and **c** For mostly digital applications (scale ≈1:1)

The optimization of these and other factors will be exemplified by the node implementations described in the following sections. We use low cost off-the-shelf components and minimal hardware resources: e.g., the hardware of the gateway nodes is similar to that of the commercial or research sensor nodes [44], while our sensor nodes have much less resources to lower the cost.

6.1 Implementation of the Sensor Nodes

Figure 6 shows several sensor nodes designed and optimized for long-term environmental monitoring applications. The node for in situ wildfire monitoring (see the PCB in Fig. 6a) is optimized for cost since this application typically makes use of a high number of nodes (up to tens of thousands). The communication protocol is unidirectional, not synchronized, since the node has no RF receive capability. As discussed in Sect. 5.1, this solution minimizes the node cost and energy requirements, and can also simplify the network operation and maintenance.

The microcontroller is an 8 bit Atmel AVR ATtiny25 with 2 KB for program and 128 bytes for data memory, clocked by its internal 8 MHz RC oscillator. The reduced accuracy of the oscillator is sufficient for node requirements and this way the node cost and energy consumption are lower.

The radio transmitter is a Texas Instruments CC1150 that requires an external 26 MHz quartz crystal. The TI CC family was selected because it implements a featureful packet modem in hardware, thus reducing the power consumption, CPU load, and program size. Also, the microcontroller and radio were selected among the smallest devices in their class that satisfy node requirements. Systems-on-Chip (SoC) with an integrated RF core may occupy less space on the PCB, but tend to have too many features for the node requirements, thus higher costs.

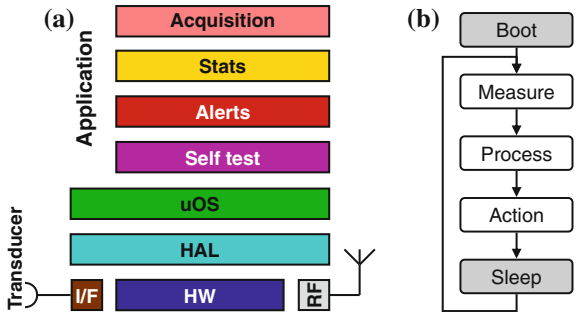


Fig. 7 Sensor node: **a** Firmware structure for reference application and **b** Operation state flow diagram

The full custom 2 KB C program has the structure in Fig. 7a. A minimal operating system supports the operation of the main program loop shown in Fig. 7b and provides the necessary interface with node hardware, support for node self-tests, and the communication protocol.

The NTC transducer needed for the in situ wildfire monitoring application is connected as a voltage divider. Every measurement cycle (once per second) it is supplied with a 0.02 % duty cycle, sufficient for the acquisition of a temperature reading using the 10-bit ADC on board the microcontroller. Each new measurement updates the stored values of the instantaneous and average temperature, and its variation speed and sign. These values are then compared with specific patterns that are considered to be closely correlated to wildfire conditions. The node sends a specific alert message for any match of these combinations.

To improve the continuity of service and lower the maintenance requirements, the sensor program periodically performs a suite of self-checks for hardware, software and configuration errors. Any anomaly is signalled to gateways, if it does not impair the radio device.

The normal sensor node activity consists of sampling the temperature every second, processing the sample, and sending one 0.28 s packet/h. The node average current consumption is about 4.7 μA, largely due to the microcontroller watchdog timer. This is kept always active to provide timing during the deep sleep periods and to attempt to recover from errors. Both the microcontroller and the transducer are active with duty cycles below 0.05 %. The microcontroller consumes about 600 μA in active state and the transducer a few tens of μA, depending on the temperature. The current consumption during packet transmission does not exceed 30 mA (less than 0.01 % duty cycle).

In these conditions, the sensor node has an unattended theoretical service time in excess of 16 years on a 1/2 AA-size 1 Ah lithium battery. Deployments for wildfire monitoring applications with up to 1,000 nodes, some operational since 2008, continue to operate without requiring battery replacements.

In emergency conditions (e.g., wildfire for the reference application), the sensor node operation mode switches from energy conservation to timely and reliably

propagation of alert messages. These are thus repeated for the whole duration of the alert condition at random intervals, between 1–3 s.

The sensor nodes use a channel in the 433 MHz ISM band to achieve a better propagation in forest environment and a longer range for a given RF power [45, 46]. These are necessary to ensure a good gateway reachability in a star topology with a minimum energy consumption for message transmission. A normal mode helical antenna (NMHA, approx. 2.4 × 1.1 cm) emerged from field tests as a good compromise between cost, size, and RF efficiency. It achieves a gain of about −9 dBi operating in close proximity to the tree trunk and protected inside the plastic package of the node.

The node PCB is a double sided FR-4 with components mounted on one side to reduce the production cost. It is finished with conformal coating to withstand long environment exposure. The sample shown in Fig. 6a requires only the battery and the NMHA (co-linear to the PCB, on the right side) to be operational. Figure 13a shows the node deployed on a tree for an application, housed in a custom, low-cost plastic case. The NTC is in good thermal contact with the surrounding air through an aperture at the lower end of the package, to prevent rain water infiltration.

Figure 6 b and c shows derivative sensor node platforms designed for fast development of related environmental monitoring applications (see Sect. 3). Their structure is similar to the temperature sensor in Fig. 6a. The NMHA is mounted on the right, while they can be plugged in application-specific boards using the B2B connector on the left. The application board typically hosts the application transducers, their interface circuitry, and the power supply. The microcontroller power supply and most of its I/O pins are routed to the connector so that it can receive power and provide processing, I/O, and networking to the application.

These nodes use Atmel AVR ATtiny261 microcontrollers (Fig. 6b, best suited for analog applications) and ATtiny48 (Fig. 6c, with more digital interfaces). Both have up to 8 KB program and 1 KB data memory. The size of the protocol stack (largely that of the temperature sensor) is less than 1 KB. With no application transducer, the average idle current consumption is about 4.8 µA and up to 30 mA during message transmission. As for the temperature sensor, these are consistent with the long unattended service duration needed by the IoT applications. Various sensor nodes (for monitoring applications for dam stability, water level, chemical gases, etc.) deployed since 2009 either operate regularly on their original battery or had lifetimes consistent with theoretical calculations similar to the one above, adjusted for application-specific sensor energy requirements and operating conditions (e.g., the ambient temperature, which can influence battery capacity).

6.2 Implementation of the Gateway Node

Figure 8a shows the PCB of the gateway node. It is a double side FR-4 with components mounted on one side to lower its production cost.

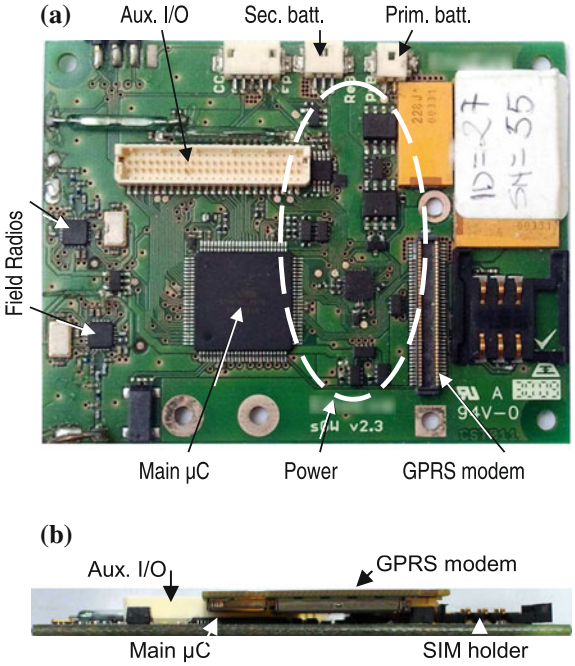


Fig. 8 PCB of the gateway node for environmental monitoring: **a** *Top view* and **b** *Side view* with the GPRS modem mounted on *top* (scale ≈1:1)

Gateway processing and storage is provided by an Atmel AVR ATmega1281 microcontroller with 128 KB program and 8 KB data memory.

In-field communications are handled by two transceivers operating in the 433 MHz band: one for the sensor nodes (receive-only, since the sensor nodes have no receiver) and one for the peer gateways. They operate on different channels to avoid the RF congestion, as discussed in Sects. 5.1 and 5.2.

A Cinterion TC63i GPRS modem with embedded TCP/IP stack provides the gateway long range communication and Internet connectivity. The modem and its SIM interface to the gateway through the connectors on the right.

The power supply module on-board the gateway can power the gateway and the GPRS modem. It makes use of three connectors on the top side: one for an external Li-ion rechargeable battery (2 Ah), one for an external lithium primary battery (3.6 V, for backup power), and one for an optional external unregulated energy supply (e.g., a solar panel) that can recharge the secondary battery.

For proper operation, the gateway PCB in Fig. 8b needs: one of the batteries, the modem and a SIM card, and the antennas for the in-field and long-range communications.

The basic operation of the gateway consists in continuously monitoring the field channel for incoming sensor node messages, using LPL to reduce the consumption of the radio receiver. Inter-gateway communications use an unscheduled protocol similar to that of the sensor nodes, since the traffic on this channel largely mirrors

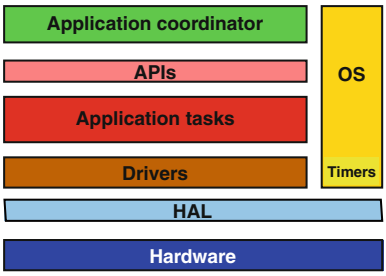


Fig. 9 Gateway firmware block diagram. At less than 64 KB, it can run on most commercial and research WSN *sensor* node resources

the sensor node traffic—the field messages that are received from the sensors are resent to the neighbour gateways for redundant storage to avoid data loss in case of gateway failure.

The small dimensions of the gateway (7.5 × 5.7 × 7 mm including the GPRS modem and excluding the batteries and antennas) facilitate its integration in application-specific cases or boards, and field deployment. Nevertheless, the application may require to embed the gateway in larger structures, e.g., at the bottom of a birdhouse (as shown in Fig. 13b), with solar panels for energy harvesting attached to the roof.

Several mechanisms contribute to the high availability of service of the gateway required for IoT applications. The power manager switches automatically to the primary battery whenever the rechargeable battery becomes depleted, e.g., after extended periods of low energy harvesting. The gateway software also implements several run-time self- and peer-assisted checks and error recovery mechanisms for its most important functions. E.g., the gateway detects when the field receivers do not receive messages for a few minutes. Conversely, the gateways will notify their peers when they fail to receive messages from them for more than a few minutes.

The implementation of the error recovery procedures attempt to limit service disruptions and avoid field maintenance operations. These include run-time reconfigurations, auto-resets that preserve the data collected from field, or, if it cannot recover, by automatically switching to boot loader mode to allow remote control or firmware update. Moreover, most configuration parameters can also be remotely changed during normal operation through remote procedure calls (RPC).

Figure 9 shows the layers of the full custom software structure of the gateway. The top-level gateway operation is controlled by an *application coordinator*. Inspired by declarative programming paradigm, it queues asynchronous service requests from various gateway tasks (e.g., as reaction to internal or external events, such as message queue nearly full or alert message received from field sensors, respectively). These are serviced based on their priority and the gateway state by triggering the needed low level tasks in a predefined order. The servicing of the queue events is preemptive. High priority requests can suspend the execution of lower priority ones to free the gateway resources they need. This addresses the platform requirement of minimum forwarding latency for field alert messages (see Sect. 3).

The *application tasks* implement generic or specific functionality needed by the WSN application, such as message queue, field message handling, sensor node status, field message postprocessing, RPC, etc. They expose an API that can be used to exchange asynchronous commands and data, e.g., to be triggered by the application coordinator.

All tasks and the application coordinator are periodically activated by a round-robin scheduler and are implemented as co-routines. This reduces data memory consumption allowing the gateway to handle up to 1,000 sensor nodes and 10 peer gateways using the 8 KB of internal RAM of the microcontroller.

Field deployment of sensor nodes and gateways does not require any manual configuration. Node IDs in communication range (for both sensors and gateways) are automatically mapped to available internal representations using memory-efficient associative arrays. The resources used by obsolete or old IDs are automatically reused when needed so that a gateway can always handle up to 1,000 sensor nodes and 10 peer gateways.

The gateway average current consumption in normal operation is 1.6–1.8 mA, depending on sensor node and peer traffic. It can rise to almost 500 mA during GPRS traffic in worst propagation conditions. Nevertheless, the gateway can operate for about one year on a D-size lithium battery in some applications, e.g., when it receives field data from only one sensor node and without peer gateways. In such cases, the average idle current decreases to 1.2 mA with the radio for the peer channel turned off, which amounts to $0.0012\,\text{A} \times 24\,\text{h} \times 365\,\text{days} \approx 10.5\,\text{Ah}$ in a year time. From the 19 Ah charge of a D-size lithium battery (e.g., Tadiran TL-5930) this leaves about $19\,\text{Ah} - 10.5\,\text{Ah} = 8.5\,\text{Ah}$ for GPRS traffic per year. At the maximum rate of 480 mA this means more than $2'50''$ of GPRS data transfer daily, which is more than enough to upload the data collected from one sensor.

Applications using gateways deployed inside sewages for level monitoring and that receive data from one sensor node and no peers demonstrated that the gateways can operate for one year on 19 Ah batteries, which is consistent with the theoretical calculations above.

Gateway idle average current can be further reduced using the hardware SPI port to interface with the radio devices and by programming the latter to autonomously scan for incoming packets instead of the software-controlled LPL using a software SPI port emulation that we used.

The gateways perform field data aggregation in a buffer (up to about 400 messages in the microcontroller internal data memory) and connect to the server either periodically (time-driven behaviour, to save energy), or when the buffer becomes full or as soon and as long alert messages are received from field (event-driven behaviour). Since the sensor nodes transmit mostly heartbeat messages that are used to update their state on the gateway, the message buffer fills gradually, unless the gateway is configured to collect variable field data (such as temperature readings) that the sensor nodes piggyback on the heartbeat messages.

T he repeater node uses the gateway design with the unused hardware and software components removed.

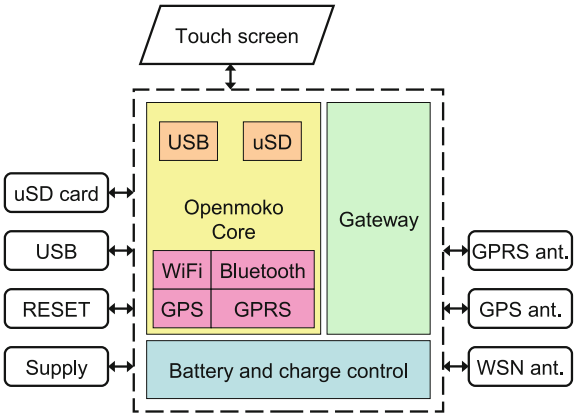


Fig. 10 Block structure of the deployment device

6.3 Implementation of the Field Deployment Device

The deployment device is made of a gateway device connected through a serial port to an Openmoko smartphoneplatform[1] that runs a Linux operating system (see Fig. 10). The gateway interfaces with the field nodes, while the field data processing and the graphical user interface (GUI) and touch screen are handled by an application running on the Openmoko Linux OS.

The Openmoko GPS can be used to assist field orientation of the operator and node localization during deployment.

Both Openmoko and the embedded gateway are supplied by a rechargeable Li-ion battery. The charging is controlled by Openmoko, while its USB port is used to power the gateway.

7 Implementation of the Application Server

The main purpose of a WSN application server is to receive, store, and provide access to field data. It bridges the low power communication segments that operate under tight latency-energy trade-offs, and the fast and ubiquitous end user field data access (for humans or IoT applications).

The full custom server software has the structure shown in Fig. 11. It provides interfaces for:

- field nodes (gateways);
- the operators and supervisors for each field;
- several alert channels;
- external access for other IoT systems.

[1] http://wiki.openmoko.org/

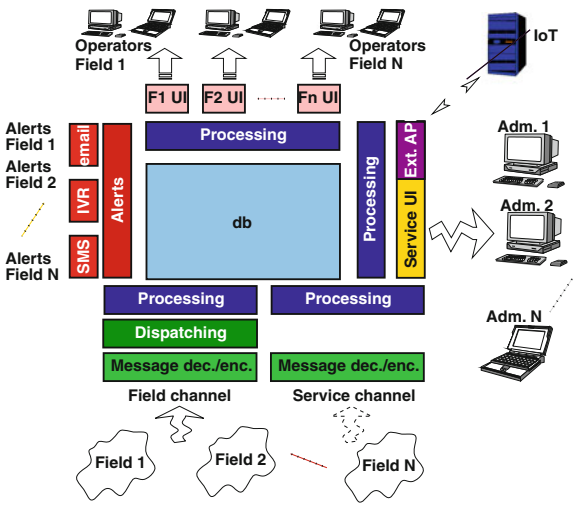


Fig. 11 Application server interfaces

Each interface has a processing unit that includes, e.g., the protocol drivers. A central engine controls the server operation and the access to the main database. The whole server is written in Java, uses a MySQL database and runs on a Linux operating system.

Two protocols are used to interface with the field nodes (gateways): normal and service (boot loader). Both are optimized for energy-efficient communication over unreliable connections.

The *normal operation protocol* acknowledges each event upon reception for an incremental release of gateway memory even if the communication is interrupted prematurely. Messages and acknowledges are sent asynchronously to improve the utilization of high latency communication channels.

The time synchronization overhead is avoided at every communication level. The sensor nodes do not timestamp their messages. The heartbeats are not time-sensitive, while the protocol ensures a timely propagation of the alert messages to the gateways. These timestamp the messages when they queue them, using their on-board relative time that is converted by the server in real-world time using a gateway-specific offset that is calculated at the begin of each gateway communication session.

The protocol for the boot loader mode is stateless, optimized for large data block transfers and does not use acknowledges. The gateway maintains the transfer state and incrementally checks and builds the firmware image. An interrupted transfer can thus be resumed with minimal overhead.

IoT environmental monitoring applications often produce large amounts of data that are typically synthesized in synoptic views by the servers [27, 28] (see Fig. 12). The server also uses high-availability techniques for a high quality of service, e.g., a shadow server automatically takes over the service if the main server fails.

The integration with other IoT applications is supported through an API for remote access to historic and real-time field data.

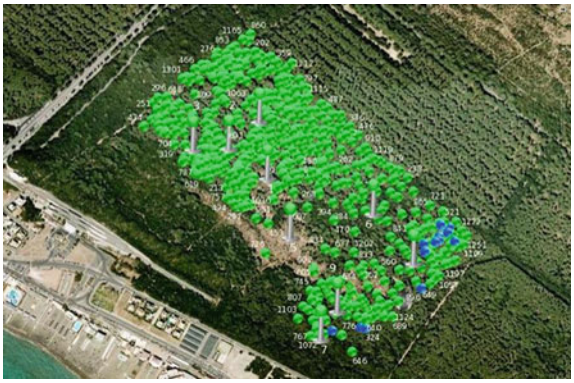


Fig. 12 Display of the status of a 1,000-sensor node field

8 Field Deployment Procedure

The node deployment procedure of the WSN platform aims to install each node in a field location that is both close to the application-defined position and ensures a good operation over its lifetime. For example, Fig. 13 shows some typical deployment solutions for the reference application sensor and gateway nodes.

Node deployment can be complex, time-consuming, error-prone, and manpower-intensive, especially for applications with many nodes. Thus, the deployment needs to be guided and have automatic checks to be able to provide a quick and easy to understand feedback to field operators, and to avoid deployment-time sensor or gateway node configuration.

The check of node connectivity with the network is important, especially for transmit-only nodes (like the reference application sensor nodes). These nodes cannot use alternative multihop message routing if the direct link with the gateway is lost or becomes unstable.

The deployment procedure of the sensor node of the reusable WSN platform takes into account the unidirectional communication capabilities of the sensor nodes. It is also designed to avoid user input and deployment-time configurations on the one hand, and a fast automatic assessment of the deployment position and reliable concurrent neighbour node deployment on the other hand.

One of the issues during node deployment is to selectively and reliably address the node under deployment. The procedure should be fast and error-free, since it is usually repeated a few times for each node, thus for many times for the deployment of WSN applications with a large number of sensor nodes, as the reference application.

For reasons of cost and hardening to prolonged exposure to open nature environment, the sensor nodes are not usually fit with power switches or other means that can be used to communicate directly with the deployment operator. Thus, a form of low-cost near field communication (NFC) between the deployment device and the node under deployment should be used. In this case, a 1-bit NFC communication

Fig. 13 Typical deployment for the reference application nodes: **a** Sensor and **b** Gateway at the *bottom* of a birdhouse

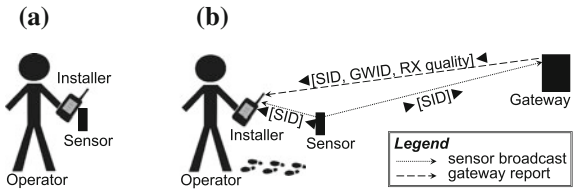


Fig. 14 Field deployment of sensor nodes: **a** Use deployment device magnet to set to deployment state, **b** Display position suitability

was implemented using a REED switch on the sensor nodes (see Fig. 6a) that can be activated in close proximity by a magnet installed inside the deployment device.

Using this NFC channel, the sensor nodes are temporarily switched to deployment operation, as shown in Fig. 14a. The node ID is collected by the deployment device that selects only strong deployment messages. These correspond to nodes within at most a few meters radius, thus providing an effective insulation from overhearing messages from nearby concurrent deployments.

The gateways that receive the node deployment messages report the link quality (see Fig. 14b). The deployment device collects all the data, computes and displays an assessment of deployment position suitability.

No gateway or node configuration is required and the procedure can be repeated until a suitable deployment position is found.

9 Conclusion

WSNs are traditionally considered key enablers for the IoT paradigm. However, due to the widening variety of applications, it is increasingly difficult to define common requirements for the WSN nodes and platforms.

We analyzed and described the requirements, specifications, design space exploration, and implementation for all the phases of the practical development from scratch of a full custom WSN platform for environmental monitoring for IoT applications. We start from the analysis of the application requirements and we define a set of specifications for the platform. Then we select a real-life, demanding application as reference which we use to guide most of the exploration of node and platform solutions, and the implementation decisions.

All aspects of the WSN platform are considered: platform structure, flexibility and reusability, optimization of the sensor and gateway nodes, optimization of the communication protocols for both in-field and long range, error recovery from communications and node operation, high availability of service at all levels, application server reliability and the interface with other IoT applications. Of particular importance are the IoT requirements for low cost, fast deployment, and long unattended service time.

All platform components are implemented and support the operation of a broad range of indoor and outdoor field deployments with several types of nodes, based on the generic node platforms that we have presented. This demonstrates the flexibility of the platform and of the solutions proposed.

Moreover, we consider that the flow presented in this paper can be used to guide the specification, optimization and development of different WSN platforms that are suitable for other IoT application domains.[2]

Acknowledgments Parts of this work were supported by ARTEMIS-JU and Governments of Italy, Spain, and Greece through the ARTEMIS project WSN-DPCM (#269389). Most of the work was done in collaboration with and included in the product lines of Minteos s.r.l., which contributed to functional specification, experimentation and production.

References

1. Ashton, K.: That 'Internet of Things' thing. Expert view, RFID J (2009). http://www.rfidjournal.com/articles/view?4986
2. Romer, K., Mattern, F.: The design space of wireless sensor networks. IEEE Wireless Comm. **11**(6), 54–61 (2004)
3. Talzi, I., Hasler, A., Gruber, S., Tschudin C.: PermaSense: investigating permafrost with a WSN in the Swiss Alps. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets '07, pp. 8–12. ACM Press, NY, USA (2007)
4. Harrop, P., Das, R.: Wireless Sensor Networks 2010–2020. Report, IDTechEx Ltd, Downing Park, Swaffham Bulbeck, Cambridge, CB25 0NW, UK, (2010)

[2] http://www.minteos.com/

5. wang, H.F.: Environmental monitoring system based on ZigBee wireless sensor and low-power technology. Adv. Mater. Res. **662**, 701–704 (2013)
6. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: ultra-low power data gathering in sensor networks. In: Information Processing in Sensor, Networks, pp. 450–459. ACM Press, New York (2007)
7. Dietrich, I., Dressler, I.: On the lifetime of wireless sensor networks. ACM Trans. Sen. Netw. **5**(1), 5:1–5:39 (2009)
8. Bashir, Y., Jalel B.-O.: Towards a classification of energy aware MAC protocols for wireless sensor networks. Wireless Commun. Mob. Computing, **9**(12), 1572–1607 (2009)
9. Yang, J., Li, X.: Design and implementation of low-power wireless sensor networks for environmental monitoring. In: Wireless Communications, Networking and Information Security, pp. 593–597 (2010)
10. Martinez, K., Padhy P., Elsaify A., Zou, G., Riddoch A., Hart J.K., Ong, H.L.R.: Deploying a sensor network in an extreme environment. In: Sensor Networks, Ubiquitous, and Trustworthy Computing, vol. 1, p. 8 (2006)
11. Hasler, A., Talzi, I., Tschudin, C., Gruber, S.: Wireless sensor networks in permafrost research – concept, requirements, implementation and challenges. In Proceedings of 9th International Conference on Permafrost, vol. 1, pp. 669–674, (2008)
12. Beutel, J., Gruber, S., Hasler, A., Lim, R., Meier, A., Plessl, C., Talzi, I., Thiele, L., Tschudin, C., Woehrle, M., Yuecel, M.: PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In: Information Processing in Sensor, Networks, pp. 265–276 (2009)
13. Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., Welsh, M.: Fidelity and yield in a volcano monitoring sensor network. In: Proceedings of the 7th symposium on Operating systems design and implementation, OSDI '06, pp. 381–396. USENIX Association, Berkeley, CA, USA, (2006)
14. Barrenetxea, G., Ingelrest, F., Schaefer G., Vetterli M.: The hitchhiker's guide to successful wireless sensor network deployments. In: Proceedings of the 6th ACM conference on Embedded network sensor systems, SenSys '08, pp. 43–56. ACM Press, New York, USA, (2008)
15. Szewczyk, R., Polastre, J., Mainwaring, A., Culler, D.: Lessons from a sensor network expedition. In: Wireless Sensor Networks, pp. 307–322 (2004).
16. Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., Hong, W.: A macroscope in the redwoods. In: Proceedings of the 3rd international conference on Embedded networked sensor systems, SenSys '05, pp. 51–63. ACM Press, New York, USA, (2005)
17. Bencini, L., Chiti, L., Collodi, G., Di Palma, D., Fantacci, R., Manes, A., Manes,G.: Agricultural monitoring based on wireless sensor network technology: real long life deployments for physiology and pathogens control. In: Sensor Technologies and Applications, pp. 372–377 (2009)
18. Verma, S., Chug, N., Gadre, D.V.: Wireless sensor network for crop field monitoring. In: Recent Trends in Information, Telecommunication and Computing, pp. 207–211 (2010)
19. Liu, Y., He, Y., Li, M., Wang, J., Liu, K., Mo, L., Dong, W., Yang, Z., Xi, M., Zhao J., Li, X.-Y.: Does wireless sensor network scale? A measurement study on GreenOrbs. In: INFOCOM, 2011 Proceedings IEEE, pp. 873–881 (2011)
20. Kuorilehto, M., Kohvakka, M., Suhonen, J., Hmlinen, P., Hnnikinen, M., Hmlinen, T.D.: Ultra-Low Energy Wireless Sensor Networks in Practice. Wiley, New York (2007)
21. Hartung, C., Han, R., Seielstad, C., Holbrook, S.: FireWxNet: a multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, MobiSys '06, pp. 28–41. ACM Press, New York, USA (2006)
22. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M., Couach, O., Parlange, M.: SensorScope: out-of-the-box environmental monitoring. In: Information Processing in Sensor, Networks, pp. 332–343 (2008)

23. Kotamäki, N., Thessler, S., Koskiaho, J., Hannukkala, A.O., Huitu, H., Huttula, T., Havento, J., Järvenpää, M.: Wireless in-situ sensor network for agriculture and water monitoring on a river basin scale in southern Finland: Evaluation from a data user's perspective. Sensors **9**(4), 2862–2883 (2009)
24. Burgess, S.S.O., Kranz, M.L., Turner, N.E., Cardell-Oliver, R., Dawson, T.E.: Harnessing wireless sensor technologies to advance forest ecology and agricultural research. Agric. For. Meteorol. **150**(1), 30–37 (2010)
25. Tennina, S., Bouroche, M., Braga, P., Gomes, R., Alves, M., Mirza, F., Ciriello, V., Carrozza, G., Oliveira, P., Cahill, V.: EMMON: a WSN system architecture for large scale and dense real-time embedded monitoring. In: Embedded and Ubiquitous, Computing, pp. 150–157 (2011)
26. Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., Pister, K.: OpenWSN: a standards-based low-power wireless development environment. Trans. Emerg. Telecommun. Technol. **23**(5), 480–493 (2012)
27. Aberer, K., Hauswirth, M., Salehi, A.: Global Sensor Networks. Technical report LSIR-2006-001, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland (2006)
28. Corra, M., Zuech, L., Torghele, C., Pivato, P., Macii, D. Petri, D.: WSNAP: a flexible platform for wireless sensor networks data collection and management. In: Environmental, Energy, and Structural Monitoring Systems, pp. 1–7 (2009)
29. Jardak, C., Rerkrai, K., Kovacevic, A., Riihijarvi, J.: Design of large-scale agricultural wireless sensor networks: email from the vineyard. Int. J. Sens. Netw. **8**(2), 77–88 (2010)
30. Peterson, I.: From microdevice to smart dust. Sci. News **152**(26), 62–63 (1997)
31. Fierens, P.I.: Number of wireless sensors needed to detect a wildfire. Int. J. Wildland Fire, **18**(7), 625–629 (2009)
32. Pantazis, N.A., Vergados, D.D.: A survey on power control issues in wireless sensor networks. IEEE Commun. Surveys Tutorials **9**(4), 86–107 (2007)
33. Ali, M., Bohm, A., Jonsson, M.: Wireless sensor networks for surveillance applications—a comparative survey of MAC protocols. In: Wireless and Mobile Communications (ICWMC), pp. 399–403 (2008)
34. Rathnayaka, A.J.D., Potdar, V.M., Sharif, S., Sarencheh, A., Kuruppu, S.: Wireless sensor network transport protocol—a state of the art. In: Broadband, Wireless Computing, Communication and Applications (BWCCA), pp. 812–817 (2010).
35. Bandyopadhyay, S., Chandrakasan, A.P.: Platform architecture for solar, thermal and vibration energy combining with MPPT and single inductor. In: VLSI Circuits, VLSIC, pp. 238–239 (2011)
36. Seah, W.K.-G., Zhi, A.E., Tan, H.: Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP)—Survey and challenges. In: Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (VITAE), pp. 1–5 (2009)
37. Sendra, S., Lloret, J., Garcia, M., Toledo J.: Power saving and energy optimization techniques for wireless sensor neworks (invited paper). J. Commun. **6**(6) (2011)
38. Polastre, J., Hill, J., Culler D.: Versatile low power media access for wireless sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys '04, pp. 95–107. ACM Press, New York, USA, (2004)
39. Cano, C., Bellalta, B., Sfairopoulou, A., Barcelo, J.: A low power listening MAC with scheduled wake up after transmissions for WSNs. Commun. Lett. **13**(4), 221–223 (2009)
40. May, T.D., Dunning, S.H., Hallstrom, J.O:An RPC design for wireless sensor networks. In: Mobile Adhoc and Sensor Systems Conference, p. 138 (2005)
41. Cohen, M., Ponte, T., Rossetto, S., Rodriguez, N.: Using coroutines for RPC in sensor networks. In: Parallel and Distributed Processing Symposium, pp. 1–8 (2007)
42. Whitehouse, K., Tolle, G., Taneja, J., Sharp, C., Kim, S., Jeong, J., Hui, J., Dutta, P., Culler, D.: Marionette: using RPC for interactive development and debugging of wireless embedded networks. In: Information Processing in Sensor Networks, pp. 416–423 (2006)

43. Yuan, F., Song, W.-Z., Peterson, N., Peng, Y., Wang, L., Shirazit, B., LaHusen, R.: A lightweight sensor network management system design. In: Pervasive Computing and Communications, pp. 288–293 (2008)
44. Potdar, V., Sharif, Chang, E.: Wireless sensor networks: a survey. In: Advanced Information Networking and Applications Workshops (WAINA), pp. 636–641 (2009)
45. Boan, J.J.: Radio Experiments With Fire. IEEE Antennas Wirel. Propag. Lett. **6**, 411–414 (2007)
46. Figueiredo, C.M.S., Nakamura, A.D. Ribas, T.R.B. de Souza, Barreto, R.S.: Assessing the communication performance of wireless sensor networks in rainforests. In: Wireless Days, pp. 1–6 (2009)

A High Performance ROIC for a Standalone Monitoring System in IOT Environments

R. Aragones, J. Oliver and C. Ferrer

Abstract This chapter presents a low power consumption Read Out Integrated Circuit (ROIC) for conditioning and acquiring capacitive sensor signals that interfaces with a supervising system using frequency processing based architecture with high efficient frequency-to-digital converters with programmable resolution. The supervising system communicates with the world using an Ethernet bus interface. The precision module of the system is constituted by the ROIC that is designed using a temperature compensated voltage reference circuit (bandgap) coupled to a low power capacitance to frequency converter (CtoF). A controlled compensation of the channel length modulation effect and the suppression of mobility dependence reduce the consumption of the ROIC down to 26 μA at half power supply. Furthermore, the temperature dependent coefficient achieved is only 16 ppm/°C.

1 Introduction

This work presents an alternative to the design of front-end systems for capacitance sensor acquisition boards using a ROIC that improves the actual frequency conversion and acquisition techniques for sensor arrays.The use of frequency acquisition techniques is motivated because these structures present better performances in sensor signals acquisitions and present easier interface to the internet.

Different data acquisition techniques and processing methods has been used recently to measure environmental and physical parameters, like in the recent

R. Aragones (✉) · J. Oliver · C. Ferrer
Universitat Autònoma de Barcelona, Barcelona, Spain
e-mail: raul.aragones@uab.cat

C. Ferrer
Institut de Microeletrònica de Barcelona (CNM-CSIC), Bellaterra (Barcelona), Spain

R. Aragones
IEEE Member, Barcelona, Spain

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_8,

application based in a multi fingered robot gripper builtusing tactile proximity sensors [1]. The use of frequency acquisition techniques, mainly when applied in noisy environments, have demonstrated to be a perfect structure to acquire a complete range of sensor signals ensuring the information integrity. Moreover, they offer better electrical featuresin real time and better efficiency compared with the traditional analog-to-digital-based converters [2]. These techniques are also ideal in noisy environments due to the fact that they present wider input spectrum range ($\pm V_{DD}$) providing easy multiplexing using FDM or TDM [3], and also implying low costs with excellent accuracies [4].

There exist in the literature similar platforms [5, 6] as the one presented in this chapter. However, the requirements introduced during the design phase of the specific acquisition circuitry of the ROIC, are responsible of the good results that presents the integrated platform.

2 Motivation

The main goal of this work is to present an easier and more precise measurement method for capacitive sensing platforms that are used, beyond other scopes, in control of supervising systems in internet structures. To obtain these objectives this chapter uses an analog ROIC in conjunction with digital processing structure based on and embedded 32 bit processor on FPGA.

The ROIC presents a lower temperature dependence (due to its second order temperature compensation), better V_{DD} variation immunity avoiding the use of an on-chip reference capacitor to perform the calibration, lower ASIC active area, higher input capacitance range operation, lower power consumption (when compared to [6]), better figure of merit. It easily can interface to systems for the Internet of Things (IOT) applications.

In a recent report of 2008, the US National Intelligence Council predicted that by year 2025 Internet nodes may be present in the quotidian things like furniture, food packages, basic home apparels, etc. Furthermore they include the IOT in the list of the most disruptive civil technology by 2025 [7, 8].

Summarizing, the main goals introduced in this chapter are:

- To propose a front-end system to monitor signal acquisition data for raw sensor platforms.
- To present the new V_{DD} compensated CtoF circuitry based on anultra-low power and high temperature compensated Bandgap Reference Circuit (BGR).
- To integrate the design in an internet accessible platform in order to allow the easy readable information of the sensing platform.

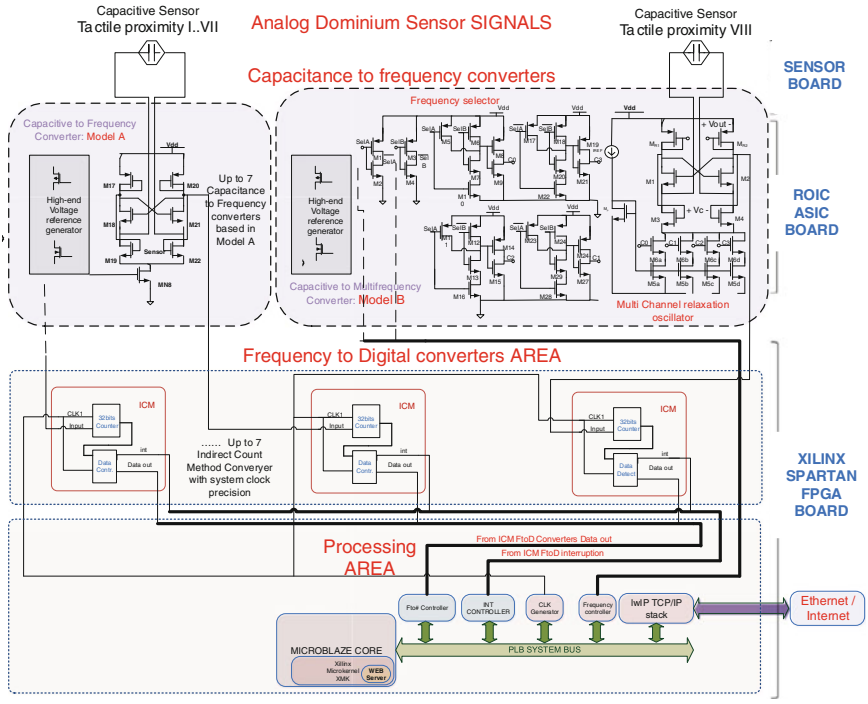


Fig. 1 ASIC ROIC 2.0 used for acquiring capacitance signals coupled to a frequency to binary code converter. The figure shows the integration of the front-end with the processing area based, in a 32 bit embedded processor

3 Read Out Circuit

Sensor acquisition structures based in frequency acquisition readout circuits allow the integration of raw sensors beside the readout circuitry, as it is in the case of multi-chip modules (MCM) where it is obtained better results in highly restrictive noisy environments. Papers as Refs. [8, 9] present the development of different frequency to digital converters (FDC) techniques that allow to evaluate the performances of the digital conversion system.

Figure 1 presents the acquisition hierarchy of the ROIC containing several interface channels for capacitive sensors signals and the novel processing unit that includes a 32 bit embedded processor (not presented previously in Ref. [13]).

The signal acquisition architecture consists of:

A. *The analog interface and frequency conversion board*

It corresponds to the CtoF converters used to interface with an array of capacitive sensors, as presented in Ref. [1]. It contains eight improved CMOS relaxation oscillators and four novel compensated bandgap reference (BGR).

B. *The digital conversion and processing unit*

It contains up to eight frequency to digital converters (FDC), being controlled by the embedded 32 bit processor. It performs the data acquisition, sensor output linearization, and internet services using the TCP/IP stack. The whole system is implemented in a Xilinx FPGA.

4 CMOS Relaxation Oscillator. State of the art

A. *Main characteristics*

A high frequency stable CtoF converter presents the properties of symmetrical output, high PSRR as well as excellent behavior at middle range frequencies. In this way, a CMOS relaxation oscillator [8] is one of the best options to develop a good CtoF converter is to design it due to the simplicity of its design. In this way, the CtoF designs [3] presents a good linearity, but with low power consumption (in the order of 760 μA), bad PSRR (only −21 dB) and without temperature compensation. In addition, the temperature coefficient degrades due to the second order effects of channel length modulation and body effect. The experimental temperature coefficient achieved in this CtoF is of 600 ppm/°C.

Figure 2 presents a CtoF circuit that solves the unwished effects, introducing the suppression of the temperature dependence due to the carrier mobility and with compensation of the thermal voltage.

B. *Response*

The block that helps to improve these parameters is the bandgap reference circuit (BGR). The CMOS relaxation oscillator output frequencyis given by Ref. (1):

$$F_{CtoF} = \sqrt{\frac{\mu C_{ox} \frac{W}{L_{M17}} \frac{W}{L_{M8}} I_{ref}}{2 \frac{W}{L_{M18}} C_{sensor^2}} \frac{1 + \lambda V_{DS8}}{1 + \lambda V_{DS17}}} \tag{1}$$

where:

- W/L_X is the MOS aspect ratio of transistor X.
- $V_{DS}x$ is the drain-source voltage of transistor X.

Due to the fact that the source terminals in M_{18} and M_{21} are fully differential, this relaxation oscillator has to be analyzed as a two-stage ring model with capacitive degeneration. So, its transfer function becomes (2):

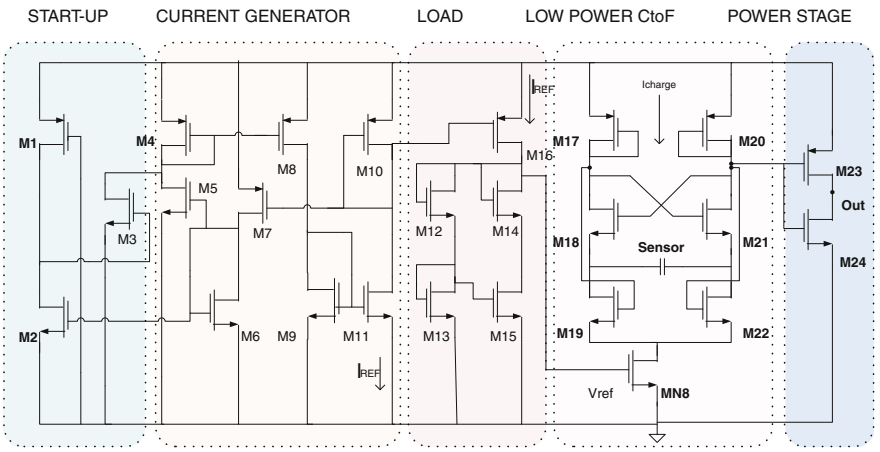


Fig. 2 Complete high-end CtoF converter based on a bandgap reference circuit and a relaxation oscillator

$$H(s) = \left[\frac{-g_m R_{MR17} C_A S}{(1 + R_{MR17} C_D S)\left(\frac{C_{sensor}}{2} s + g_m\right)} \right]^2 \tag{2}$$

where:

- R_{MR17} is the equivalent resistance value of M_{17}, biased in the linear region, like an active resistor.

Analyzing the thermal noise in each drain and the I_{noise} in drain-source, it is easy to obtain the equation that models its relative phase noise (3):

$$Relative\ phase\ noise \approx \frac{1}{I_{DD}} \left(\frac{\omega_0}{V_{cswing} \Delta\omega} \right) \tag{3}$$

where:

- V_{cswing} is the internal swing voltage range.
- I_{DD} is the total power consumption.
- ω_0 is the fundamental central frequency.
- $\Delta\omega$ is the bandwidth.

As shown in Fig. 3, the relaxation oscillator based on Ref. [9] (designed using Cadence layout XL and the microphotography of this oscillator with a final size of only $120.2\,\mu m * 56.2\,\mu m$, and integrated in AMS 0.35 technology) occupies a reduced area of the integrated circuit.

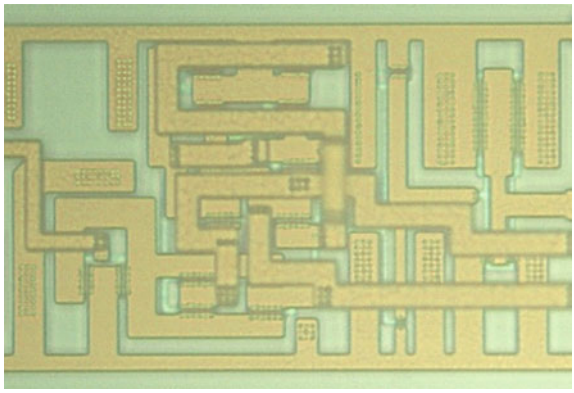


Fig. 3 Low power oscillator microphotography (AMS 0.35 technology)

5 Improved Low Power Temperature Compensated BGR

A. *Main characteristics*

The use of this new relaxation band gap reference circuits helps to improve the power supply rejection ratio and to obtain a circuit independent of the power supply variation [11] and the temperature. To do so, particular improvements in power supply circuitry were required in the design. Figure 2 presents the new band gap reference generator (BGR) integrated beside the low power relaxation oscillator. This BGR includesa start-up, a current generator and a load subcircuits.

This design avoids the use of any external resistor in the start-up subcircuit and decreases significantly the power consumption reducing the output frequency range, proportionally to the charging sensor current.

Moreover, power supply is dramatically reduced down to 954 mV, thus minimizing the temperature dependence of the whole CtoF circuit to 16 ppm/°C.

The relaxation oscillator frequency output response is linearized, making it independent from the V_{DD}. A PSRR lower than −49 dB (at 100 Hz) is obtained.

As Fig. 2 shows, the acquisition circuit is divided into two main blocks:

- The ultra-low power relaxation oscillator circuit, consisting on the relaxation oscillator and the power stage sub-circuits.
- The BGR circuit, which is divided into the start-up, the current generator and the load sub-circuits.

The CMOS relaxation oscillator there was not initially any V_{DD} compensation. Thus, a change in V_{DD} implied directly a variation in CtoF output frequency as shown in (1).

This circuit needs of a constant charging/discharging current (I_0) introduced by the BGR using a capacitor and considering a bi-stable effect. This constant current imposes a constant charging slope time in the capacitor (generating a constant F_{out}). This consideration is applicable whenever a constant power supply is used.

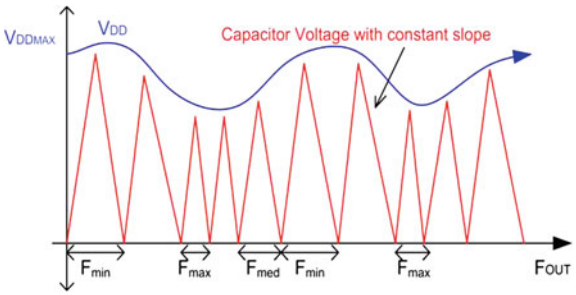


Fig. 4 Traditional CtoF response applying a V_{DD} variation, with standard BGR

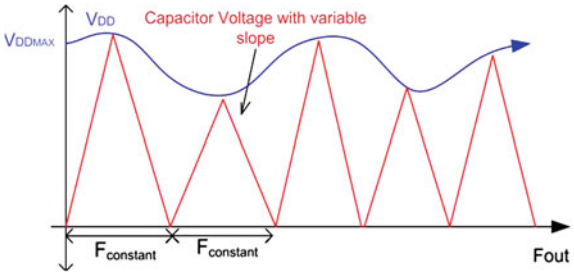


Fig. 5 Novel CtoF design applying a V_{DD} variation, with compensated BGR

Thus, the output frequency is proportional to (4):

$$F_{out} \approx \frac{I_c}{2CV_{DD}} \tag{4}$$

where:

- I_c, or I_{charge} in Fig. 2, a proportional current of I_{ref} (BGR).
- C is the capacitive value of the tactile proximity sensor.
- F_{out}, is output frequency.

As the relaxation oscillatoris powered by $V_{DD,}$ and any change in $V_{DD,}$ produces a variation in V_{DS17} in (1), an inverse dependence in F_{out} (due to the constant slope time characteristics in (4)), results in different output frequencies (F_{max}, F_{min}) as in Fig. 4.

The BGR introduced in Fig. 5 solves this undesired effect. A redesign of the current generator sub-circuit that makes the I_{REF} output current dependent on V_{DD}, compensates the V_{DD} dependences of the relaxation oscillator module (CtoF).

The goal is to generate a charging current inversely proportional to V_{DD}, thus obtaining from (4) an output frequency that only depends on the sensor capacitance. With this approach, the BGR has an output voltage not constant as usual in traditional BGRs.

B. *Bandgap reference circuit analysis*

This circuit will generate the reference voltage that compensates the V_{DD} in the CtoF converter (see Fig. 2).

(1) *Current generator sub-circuit*

The kernel of this sub-circuit is attached to transistors M_5–M_6–M_9–M_{11} determining the value of I_{ref}.

To obtain an output reference voltage independent from the temperature coefficient, it is necessary to build a bias current where the temperature coefficient should be controlled by the V_{TH} difference of two transistors. It is the generation mechanism which obtains the output reference voltage of this BGR.

Thus, it is necessary to combine transistors of two different technologies (which it is not possible using AMS 0.35 μm technology) or to integrate transistors designed to be used with different power supplies (3.3 and 5 V). In this particular circuit, M_5 and M_9 are integrated using 5V CMOS powered technology with a 0.7 V threshold voltage (NMOS), and the remaining MOS transistors are powered at a 3.3 V with a threshold voltage of 0.45 V (NMOS). Additionally, with the purpose of obtaining the current I_0 (5), the following assertions must be considered:

- The V_{GS} values of M_5 and M_6, M_9 and M_{11} are the same.
- M_5 and M_9 are biased in the sub threshold region, while M_6 and M_{11} are maintained in the saturation region.
- No body effect implies $V_{th5} = V_{th9}$ and $V_{th6} = V_{th11}$.

$$I_0 = \frac{\mu C_{ox} W_{11}}{2L_{11}(N-1)^2} m^2 V_T^2 \ln^2\left(\frac{W_9 L_5}{W_5 L_9}\right) \tag{5}$$

where:

- m is defined as the subthreshold slope parameter.
- N is defined as the MOST form factor relation in (6):

$$N = \sqrt{\frac{W_{11} L_6}{L_{11} W_6}} \tag{6}$$

It is important to remark that in (2) the current generated by this stage depends on the temperature coefficient due to the mobility μ and the V_T voltage. So it is necessary to build a load subcircuit that supresses or minimizes the temperature dependence.

(2) *Start-up sub-circuit*

Since the reference voltage generator has two stable states, corresponding to I_0 and to the zero current, to ensure that the stable state is achieved, the sub circuit formed by M_1–M_3 is introduced in order to force an initial current.

(3) *Load sub-circuit*

To perform the output voltage equation (7), it must be assumed that:

- All transistors are working in the saturation region in the load sub-circuit.
- M_{14} and M_{15} are biased mirroring I_{REF}.
- M_{12} and M_{13} are an active voltage divider.

$$V_{\mathrm{Ref}} = \left(1 + \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)\left(\sqrt{\frac{2Io}{k_{15}}} + V_{TH}\right) + \left(1 - \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)V_{TH} \quad (7)$$

Observing (7) it is easy to identify that k_{15} (forward transconductance) depends directly with the temperature coefficient via μ and it is easy suppressible with the μ of I_0 (5) since they are in the same fraction. Furthermore I_0 has a quadratic dependence with the thermal voltage that should be suppressible with V_{th} of V_{ref} due the square root of I_0.

Equating correctly all the terms is possible to suppress or minimize V_{ref} thermal dependence.

(4) *Temperature compensation*

Considering that the threshold voltage decreases linearly with the temperature, the next expression for the NMOS transistor is written:

$$V_{TH}(T) = V_{TH}(T_0) - K_{t1}(T - T_0) \quad (8)$$

where:

- K_{T1} is a constant VSIM3.3 value.

Taking into account (7) and (8) and deriving it in reference to the temperature, expression (9) is obtained:

$$\frac{\partial V_{Ref}}{\partial T} = \left(1 + \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)\left(\frac{mK_B \ln\left(\frac{W_9L_5}{W_5L_9}\right)}{q(N-1)}\sqrt{\frac{k_{11}}{k_{15}}} + K_{t1}\right) + \left(1 - \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)\frac{K_B}{q} \quad (9)$$

Now, equating (9) to zero, (10) is obtained which gives us the relationship that should fulfill this load sub-circuit transistors form factors to cancel all the dependence with temperature.

$$\left(1 - \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)\frac{K_B}{q} = \left(1 + \sqrt{\frac{(W/L)_{13}}{(W/L)_{12}}}\right)\left(\frac{mK_B \ln\left(\frac{W_9L_5}{W_5L_9}\right)}{q\,(N-1)}\sqrt{\frac{k_{11}}{k_{15}}} + K_{t1}\right) \quad (10)$$

(5) *Body effect*

The previous model does not considerers the equivalent temperature coefficient due to the body effect or the channel length modulation effect. The body effect in the current generator sub-circuit has no effect because all the transistors have their $V_{BS} = 0$. In particular M_5, M_6, M_9 and M_{11} have their source grounded and M_4, M_7, M_8 and M_{10} have their source to V_{DD}. The same explanation is applicable to the start-up circuit.

In order to consider the body effect of M_{12} and M_{14}, we can write the bulk dependency of the threshold voltage as:

$$V_{TH} = V_{TH0} + K_1 \left(\sqrt{2\phi_s - V_{BS}} - \sqrt{2\phi_s} \right) \tag{11}$$

where:

- V_{TH0} is V_{TH} for $V_{BS} = 0$.
- K_1 is a process technology parameter.

By using (11) it is possible to rewrite (7) considering the body effect as (12):

$$V_{REF} = V_{REF0} + (V_{th12} - V_{th0}) - (V_{th14} - V_{th0}) \tag{12}$$

Finally taking into account (11) and (12) the temperature coefficient should be rewritten as (13):

$$\frac{\partial V_{Ref}}{\partial T} = K_1 \frac{\partial 2\phi_s}{\partial T} \left(\frac{1}{\sqrt{2\phi_s - V_{BS12}}} - \frac{1}{\sqrt{2\phi_s - V_{BS14}}} \right) \tag{13}$$

So forcing $VBS_{12} = VBS_{14}$, it will suppress its temperature dependence due to body effect.

(6) *Channel length modulation effect*

In (7) the channel length modulation effect was neglected. Taking into account this second order effect, it is possible to rewrite (5) as (14):

$$I_0 = \left(\frac{1}{N - \frac{1}{\sqrt{1+\lambda V_{DS11}}}} \right)^2 \ln^2 \left(\frac{W_9 L_5}{W_5 L_9} \right) \frac{m^2 V_T^2 K_{11}}{2} \tag{14}$$

Since transistors M_5 and M_9 work in the subthreshold region, and the channel length modulation effect can be neglected when the drain-source voltage is four times its thermal voltage, we can obtain the final I_0 that will suppress the temperature coefficient, as (15):

$$I_0 \cong \left(1 - \frac{\lambda V_{DS11}}{N-1} \right) \left(\frac{1}{N-1} \right)^2 \ln^2 \left(\frac{W_9 L_5}{W_5 L_9} \right) \frac{m^2 V_T^2 k_{11}}{2} \tag{15}$$

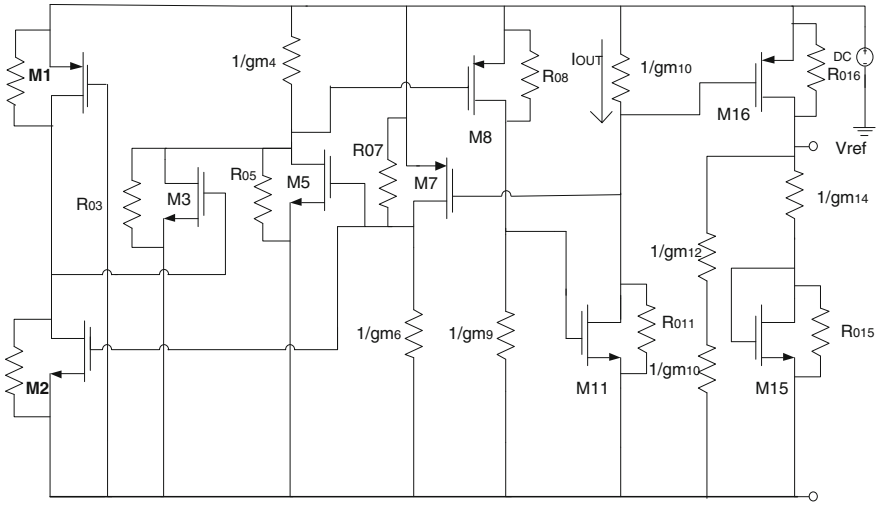


Fig. 6 Voltage reference generator PSRR equivalent circuit using DC analysis

Taking into account (7) and (15) and deriving it respect to the temperature, Eq. (16) is obtained:

$$\frac{\partial V_{Ref}}{\partial T} \cong V_{REF0} + (2V_{TH} - V_{REF0})\frac{\lambda}{2}K_{T1} \tag{16}$$

Forcing $2V_{TH} = V_{REF0} \approx 0,9\,\mathrm{V}$ a correct suppression of the temperature dependence will be obtained.

(7) *PSRR*

PSRR becomes critical as a result of the ultra low power consumption of the BGR. In fact, for low drain currents the transcoductance g_m becomes very small degrading the PSRR.

To improve the PSRR, r_0 must take a long value, obtaining a high gain of $g_m \cdot r_0$.

In particular, in subthreshold region, considering the exponential response of V_{DS}, with a few I_D current, a high value of r_0 is obtained. Furthermore, in saturation region, and with a large channel transistor, $r_o = 1/\lambda I_D$, and also with a low I_D, a high value of r_0 is obtained. Figure 6 shows the equivalent circuit for calculating the PSRR at DC. It is found in (17):

$$\frac{i_{OUT}}{v_{DD}} \cong \frac{1}{r_{07}}\left(\frac{g_{m10} - g_{m11}g_{m09}g_{m7}}{g_{m11}g_{m09}g_{m10}}\right)^{-1} \tag{17}$$

In order to achieve small changes in I_0 due to V_{DD}, M_{16} must be designed with a sufficient long channel in order to obtain a high value of r_{016} to minimize the term in parenthesis, obtaining (18):

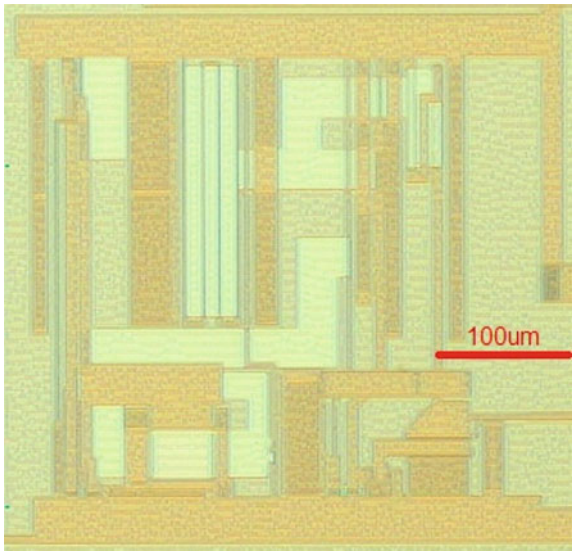


Fig. 7 BGR microphotography

$$\frac{v_{REF}}{v_{DD}} \cong \left(\frac{1}{r_{016}} + \frac{g_{m16}}{g_{m10}}\frac{i_{OUT}}{v_{DD}}\right)\left(\frac{g_{m12}+g_{m10}}{g_{m15}g_{m12}}\right)\left(1 - \frac{g_{m15}+g_{m14}}{g_{m15}g_{m14}}\frac{g_{m10}}{g_{m15}g_{m10}}\right) \tag{18}$$

Moreover, it is remarkable that PSRR depends on the relationship of g_{ms}, not on their absolute values, so it is easy to obtain optimal PSRR values for low I_{out} current.

According to specifications, it is necessary to force the output oscillation signal to $\pm V_{DD}$ (FSK) using the power stage sub-circuit.

Figure 7 shows the layout of the low power compensated bandgap reference circuit and its microphotography with a final area of only $0,013\,mm^2$.

6 Processing Unit

The digital conversion and processing area implements up to eight channels of frequency to code converters (FDC) using a frequency conversion based on the indirect counting method [4]. This method called, *FDC ICM conversion with system frequency precision*, transforms the output frequency obtained from the Ctof to a digital data (32 bit precision per each channel) with low relative quantification error.

The FDC control and processing task is controlled by an embedded 32 bit processor. Its implementation in a Xilinx FPGA development board is shown in Fig. 8. It presents several advantages compared to previous work [13]:

- The precision is programmable. It depends on the performance required, being use typically the 32 bit embedded processor model.

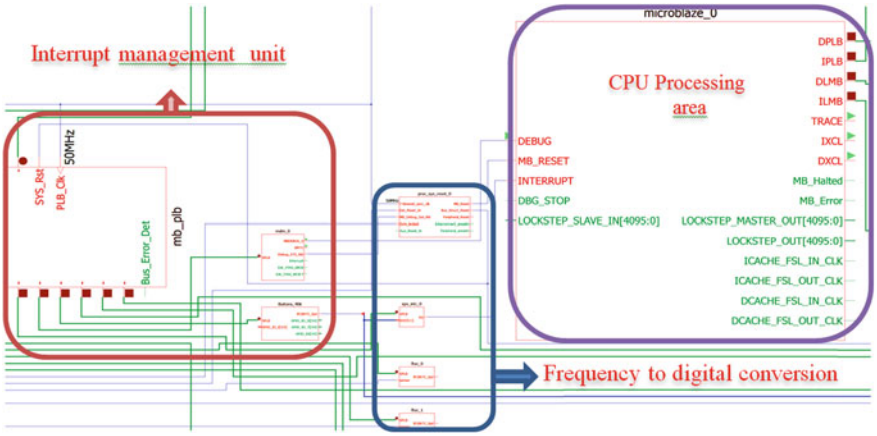


Fig. 8 Processing area with the FDC, interrupt controller and the embedded 32 bit processor

- It can be programmed using a high level C language.
- It uses interrupt services routines.
- It presents higher performance and scalability.
- It connects to different peripherals, like TCP/IP stack, USB, I2C and SPI UART.

This embedded CPU performs the FDC data acquisition using interruption routines. The interruption service routines are responsible of the synchronization of the acquisition data that arrives to the system from the ROIC. The output of the frequency to digital circuit presents a hyperbolic response (1). The processor based design allows the CPU to be responsible of performing the data linearization following this procedure. In order to generalize the procedure, the following steps can be followed:

- The relationship between the frequency and the capacitance is captured experimentally making different measurements using the capacitance range (from 1 pF to 100 μF) admitted by the acquisition system.
- Thus, a 1/x hyperbola frequency output response is obtained, as shown in Fig. 9a.
- Mathematical curve fitting algorithms using nonlinear minor quadratic equation model allows calculating the fitting curve. It is achieved the a and b values that fits in the following expression:

$$f = \frac{a}{c + b} \tag{19}$$

- It is obtained that a is 51.5681 and b is 0.0107 in expression (19) with a relative error of only 0.11 % (Fig. 9b). In the expression (20) the frequency f is expressed in KHz and the capacitance C in nF.

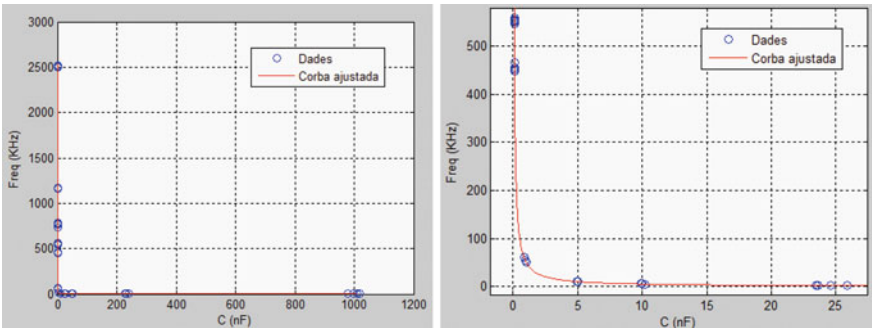


Fig. 9 **a** Capacitance sweep response versus output frequency. **b** A zoom in the center of the curve

$$C(\text{nF}) = \frac{51.5681}{f(\text{KHz})} - 0.0107 \tag{20}$$

The pre-calculation of the fitting curve allows to calculate in real time (by the embedded processor) the linearization data (20), thus obtaining the real capacitance value measured by the sensor.

Finally the processor performs the data transmission and monitoring using a customized web server [14]. To perform this process is necessary to implement [15]:

- lwIP TCP/IP stack, a small independent implementation of the TCP/IP protocol suitable for use in embedded systems.
- Xilinx Micro Kernel (XMK) operating System.
- XilMFS memory file system library.

Data is sent to a web server that hosts the web site which monitors information of different channels of the supervisor system [16, 17]. Figure 10 shows a windows capture of the implemented system monitoring information of three different FDC channels from the ASIC ROIC. The information monitored is:

- Information of each FDC channel with a precision up to 32 bit. The frame shows the sensor parameters Number (of the sensor), M and N (parameters related to the frequency to code counter), FDC Out (frequency of the sensor), and the real equivalent capacitance value obtained in real time from (20). In this particular period of time, three measured capacitive values of 179 pF, 666 pF and 5.56 nF are obtained in the screenshot.
- Finally, the screenshot presents up to four different graphs, in this case plotting the information obtained by the acquisition system and configured to acquire information of the three sensors humidity (Humirel HS 1100), pressure and barometer sensors. Different scales in time are possible.

Fig. 10 Web site that allows to monitor the result of three different FDC channels and plot the results hourly, weekly or monthly

7 Results

A. *Band Gap Reference*

Figure 11 shows the BGR output response obtained experimentally from the application of a V_{DD} ramp ranging from 0 to 3.3 V. In the figure, the oscillating (green response) shows excellent frequency stability for a V_{DD} value above 2.7 V (in yellow) being completely independent of the power supply. It can be also noted that the BGR starts regulating at 1.39 V of V_{DD}, while the oscillator is working al 1.13 V. The transistors M_{11} and M_{16} of the BGR are responsible of the generation of the constant voltage ramp (red graph) that compensates the charge time (4) of the CtoF converter.

Figure 12 shows the behavior obtained in PSRR. The figure shows that a value of −53 dB is obtained at 100 Hz and −49 dB at 10 MHz, when considering a power supply value of only $V_{DD}/2$. It is achieved an excellent PSRR that allows the circuitry to be immune to V_{DD} variations.

Analyzing the temperature coefficient (Fig. 13) for 2 and 4 V of V_{DD}, 16 and 21 ppm/°C have been respectively obtained. These results show again an excellent temperature coefficient compared with literature.

Regarding to the phase noise test [12], it presents an excellent result obtaining −21.2 dBC/Hz near the carrier and a maximum value of −112.3 dBC/Hz at 612 KHz from the carrier, obtaining better results compared to previous works.

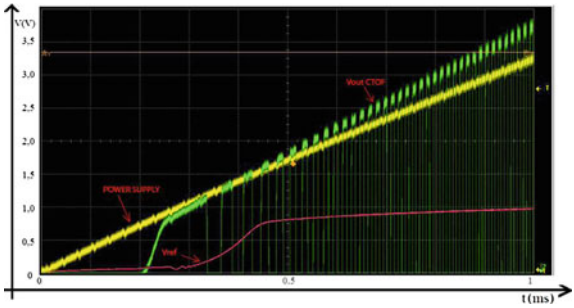


Fig. 11 VRG and CtoF output signals for a V_{DD} ramp from 0 to 3.3 V

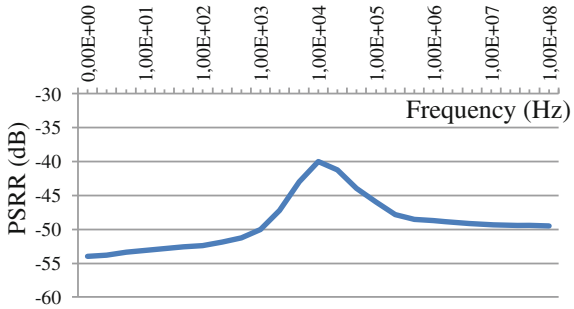


Fig. 12 PSRR test

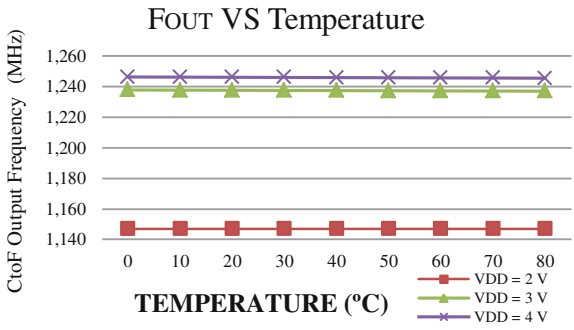


Fig. 13 F_{OUT} versus temperature and V_{DD} sweep

B. *Model comparison*

Table 1 presents a comparison of the results of the converter (BGR & CtoF) presented in this work with respect to results presented in recent bibliography. It can be observed that it presents better results in all the tests introduced in the table. Specifically, the table shows a nine-times reduction in area compared to Ref. [5]. The introduction of the high frequency FDC with highresolution (32 bit), presents significant improvements in the circuit in respect to Ref. [3]:

- The input capacitance range is spread up to 100 µF.
- The power consumption is 10 times better.

Table 1 Comparisons to previous CTOF

Model	[6]	[3]	This work
Input cap range	0.8–82 pF	2 pF–7.8 nF	5 pF – 100 μF
Chip area	0.51 mm^2	0.173 mm^2	0.142 mm^2
F.o.M (pJ/step)	49	217	41
P. Consumption (μA)	64	759 (267 at 1.8 V)	26 (at 1.8 V)
Max P noise (dBC/Hz)	N.A.	−101.9	−112.3
V_{DD} swing compen (ppm/V)	121	1270	98
Temp. Dep.	Not. Comp.	600 ppm/°C	16 ppm/°C
PSRR	N.A.	−21 dB	−53 dB
Resolution (bits)	15	12	32

- It also presents an improved stability in the CtoF in front of the V_{DD} swings.
- Also presents:
  - The best temperature coefficient ($16ppm/°C$).
  - The best figure of merit FOM(41pJ/step).

In the end, the ultra-low power CtoF with the 32 bit FDC ICM-based converter, helps to optimize the acquisition equivalent sampling rate up to 100 Ksps.

C. *Processing board results test*

In order to check the adequacy of the CtoF-bandgap circuitry, it was inserted in a sensor-based signal acquisition platform. The output of the platform is presented using a web site hosted in the Xilinx board web server (Fig. 8). Figure 14 shows this setup with the integrated ROIC (that includes the CtoF and bandgap), the circuit to test in this application.

The figure of merit test(FoM) (21) is performed to evaluate the converter effectiveness in energy consumption to resolution. 41 pJ/step is obtained being an excellent result for such kind of converter.

$$FoM = \frac{T_{measurement} * TotalPower}{2^{Resolution}} \tag{21}$$

Figure 15 present a die microphotography of the ROIC. It is designed using the AMS 0.35 μm technology with fourmetals and two polys using the Europractice hitkit 3.7. The total dimension of the ASIC ROIC core is 1200.2 μm ∗ 760 μm.

8 Conclusion

In the era of internet of things, the real time monitoring system has become an area of special interest since it allows to be instantly informed of the sensor status,especially in systems supervisors [15, 16].

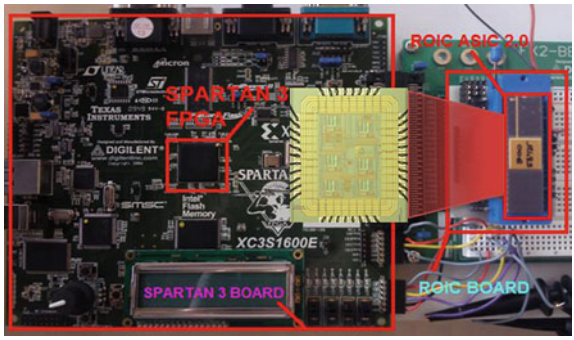


Fig. 14 Acquisition platform, with processing board (4 channel FDC and 32 bit embedded processor) and ASIC ROIC

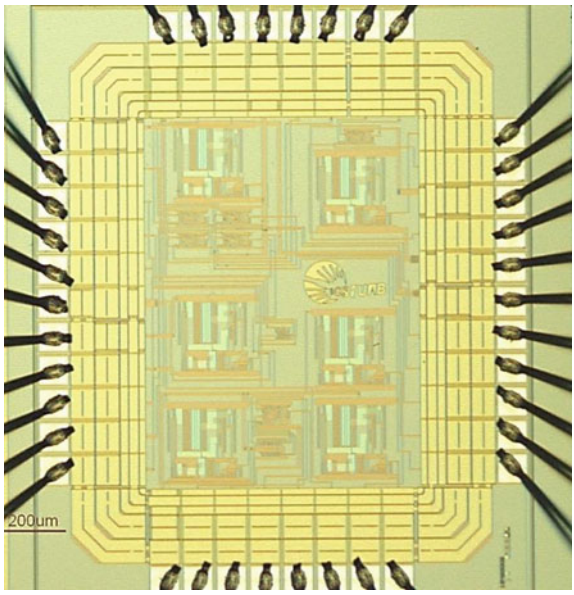


Fig. 15 ASIC ROIC microphotography

This work has presented a complete signal acquisition architecture from the front-endtoback-end of the system based in capacitance sensor measurement. The present architecture has different components that make up the low front end and shows how every thing is integrated into a system based on FPGA. In particular, FPGA-based systems of fera unique customization system minimizing the requirement to incorporate additional circuitry.

Within the acquisition system, the front end is especially important in that regard to the measurement accuracy. Thus, this word presents a novel compensated voltage reference circuit that has been integrated in an AMS 0.35 μm ROIC (readout integrated circuit).

This ASIC performs precise capacitance sensor measurements, being immune to V_{DD} swing and temperature dependences.

In order to test the performances of the circuit, a new ROIC was integrated with eight capacitive sensor structures to monitor sensor based systems. It resulted in an improved circuitry for capacitance to frequency converters.

The results obtained reveal excellent performance of the global ROIC compared to other previous converters [2, 5, 6].

To improve the conversion mechanism, as well as the whole system performance, a novel frequency to code converter and a novel processing platform based in the embedded 32 bit processor were implemented. The frequency acquisition system improved the performance and results previously presented [13].

Main conversion results are monitored using a web site hosted in a Xilinx Spartan 3E FPGA using an embedded web server using a lwIP TCP/IP stack [18].

References

1. Goeger, D., Blankertz, M., Woern, H.: A tactile proximity sensor, in Proceedings of the IEEE Sensors 2010 Conference, pp. 589–594 (2010)
2. Aragonés, R., Oliver, J., Ferrer, C.: A multichannel voltage to frequency to code converter for sensors applications, in Proceedings of the XXIII Conference on Design of Circuits and Integrated Systems (DCIS), November 2008
3. Aragonés, R., Álvarez, P., Oliver, J., Ferrer, C.: Comparison of readout circuitry techniques for data acquisition in raw sensor systems, in Proceedings of the IECON 2010 conference, Arizona, November 2010, pp. 1252–1257
4. Yurish, S.Y.: Data Acquisition for Smart Sensors and Transducers. Wiley Sensors Portal book, pp. 52–53, 58–60 (2001)
5. Shih, I.-C., Shen, T., Otis, B.P.: A 2.3 W wireless intraocular pressure/temperature monitor. IEEE J. Solid-State Circuits **46**(11), 2592–2601 (2011)
6. Tan, Z., Shalmany, S.H., Meijer, G.C.M., Pertijs, M.A.P.: An energy-efficient 15-bit capacitive-sensor interface based on period modulation. IEEE J. Solid-State Circuits **47**(7), 1703–1711 (2012)
7. Council, N.I.: Disruptive civil technologies: six technologies with potential impacts on us interests out to 2025. Conference Report CR 2008–2007, April 2008
8. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Int. J. Comput. Telecommun. Netw. **54**(15) (2010)
9. Song, B., Kim, B.: A 50 % power reduction scheme for CMOS relaxation circuit, AP-ASIC'99 Conference, pp. 154–157 (1999)
10. Aragonés, R., Oliver, J., Ferrer, C.: A generic signal processor for frequency sensor data acquisition, in Proceedings of the XXII Conference on Design of Circuits and Integrated Systems (DCIS), November 2007, pp. 345–350
11. De Vita, G., Iannaccone, G., Andreani, P.: A 300 nW, 12 ppm/C voltage reference in a digital 0.35 μm CMOS process. in Symposium VLSI Circuits Digest of Technical Papers, Honolulu, HI, 2006, pp. 81–82
12. Razavi, B.: A study of phase noise in CMOS oscillation. IEEE J. Solid-State circuits **31**(3), 331–343 (1996)
13. Aragonés, R., Oliver, J., Ferrer, C.: A 16 ppm/°C ROIC for capacitive-sensor signal-acquisition applications, in Proceedings of the IEEE Sensors 2012, Taipei
14. Hylby, Z.: Embedded web services. IEEE Wirel. Commun. **17**(6):52–57 (2010)
15. Xilinx: ZedBoard v 14.4: Zynq-7000 AP SoC concepts, tools, and techniques a hands-on guide to effective embedded system design (March 2013)

16. Garcia-Macias, J.A., Alvarez-Lozano, J., Estrada-Martinez, P., Aviles-Lopez, E.: Browsingthe internet of things with sentient visors. IEEE Comput. Soc. **44**(5), 46–52 (2011)
17. Wang, W., Tse, P.W., Lee, J.: Remote machine maintenance system through internet and mobile communication. Int. J. Adv. Manuf. Technol. **31**(7–8), 783–789 (2007)
18. Armstrong, R., Muggli, M., Ouellette, M., Thammanu, S.: XAPP433—Embedded system example: web server design using microBlaze soft processor. Xilinx Application Note (2006)

Ambient Assisted Living Environment Towards Internet of Things Using Multifarious Sensors Integrated with XBee Platform

N. K. Suryadevara, S. Kelly and S. C. Mukhopadhyay

Abstract In this study, we reported the design and development of an integrated platform for monitoring and controlling of household appliances using internetworking technologies associated with factors of ZigBee wireless sensor network. The intelligent internetworking architectural mastery plus the reliable measurements associated with household sensors variables are comprehended. The developed system is a combination of distributed smart sensing systems and a data system for aggregation and exploration of fused data. Benefits associated with the developed system are in effective realization of household appliances monitoring variables through Internet of Things. The robustness of the system in executing multiple tasks for long durations provides the longitudinal assessment behavior of the inhabitant. The prototype has been tested in the actual home environment and the results are viewed through real-time graphical data analysis representation.

1 Introduction

The applications of Wireless Sensor Networks (WSN) have wide usage in areas such as smart energy, smart logistics, health care and home automation [1–3]. The most applicable protocol used or followed in WSN is the IEEE 802.15.4 (ZigBee) for effective and reliable communications [4]. ZigBee was designed for use in local networks such as home automation environments. Moreover, it does not directly communicate with servers on the internet without proper integrating mechanisms [5]. Remote management and controlling of ZigBee based devices over the internet can be mechanized by following certain architectural design strategies.

N. K. Suryadevara · S. Kelly · S. C. Mukhopadhyay (✉)
School of Engineering and Advanced Technology, Massey University,
Palmerston North, New Zealand
e-mail: S.C.Mukhopadhyay@massey.ac.nz

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors,
Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_9,

The data transmission of smart sensing devices using ZigBee over the internet can be done by integrating an internet gateway with a ZigBee network. In a ZigBee network, end devices collect data and send data to the gateway, which then translates the data from ZigBee protocol format to Internet Protocol version 6(IPv6) format, and vice versa. This allows ZigBee devices to communicate with servers on the Internet. In this research task, a novel architectural design strategy is implemented for remote management and controlling of household appliances via the internet.

This chapter demonstrates an operational low-cost and flexible solution for smart home automation, remote management and control of smart sensing devices related to household appliances. The system is realized using a ZigBee based integrated platform system to monitor a home environment through the internet. Also, it discusses about internetworking mechanisms which are practicable to integrate with sensing modules like Intelligent home monitoring systems such as [6–11].

1.1 Related Works

In order to have low-power consumption, the ZigBee protocol follows the physical and data link layer stack of IEEE 802.15.4. On the other hand, it has limitations on network and application layer functionalities such as addressing, routing and interoperability with the internet. Alternatively, adapting to IPv6 Low Power Personal Area Network (6LoWPAN) protocol, help us to have better end-end communication with the sensing devices. However, translation mechanisms such as SOAP/REST, GRIP [12] will increase the complexity of the network system.

The concurrence impact of wireless sensor network on the IEEE 802.15.4 devices was assessed in [13]. Studies in [14, 15] have been proven theoretically that WSN performance is more perceptible to reduce when interfered with other radio networks and likelihood of faults in 802.15.4 network is high. In [16], the authors have studied the coexistence of IEEE 802.15.4 and other radio networks, based on outage probability, packet loss rate and changes in RSSI value.

Research for internetworking 802.15.4 with IP networks has been conducted. 6LowPAN [17] provides well-defined method for transferring IPv6 packets over 802.15.4 network. However, complexity to deploy in 802.15.4 network nodes is very difficult [18]. IPv6 over 6LoWPAN is proposed by the Internet Engineering Task Force (IETF) working group to accomplish the concept of IP-based WSN. A new layer is incorporated between IPv6 network layer and 802.15.4 MAC layer, which is entitled adaptation layer. It was observed that the adaptation layer, in particular the fragmentation process may increase the energy consumption of a sensor node by 5–10 % [18]. As mentioned above there are many issues related to integrating IPv6 with the WSN.

In this chapter, the above issues have been resolved and effective internetworking mechanisms for transmission of ZigBee based smart sensing information over internet is presented.Additionally, the integrated platform is flexible to use with existing

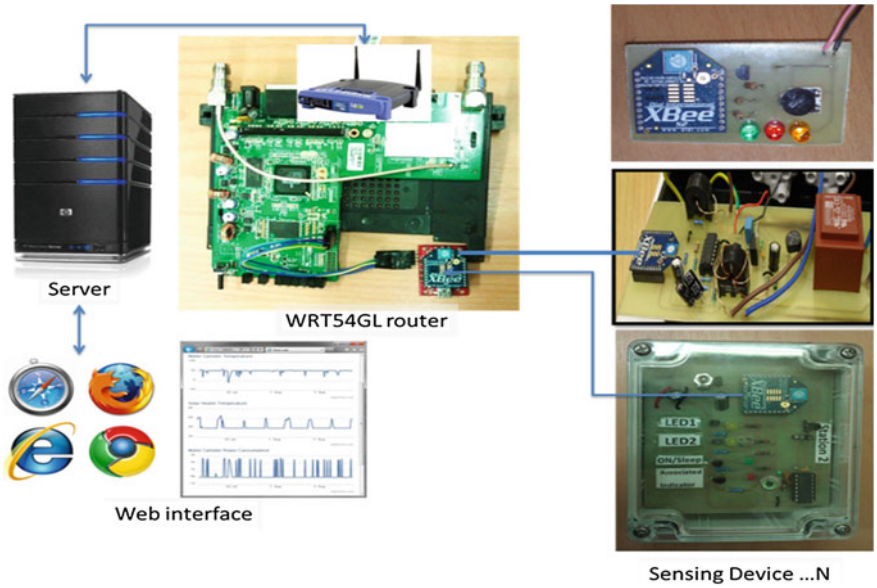


Fig. 1 System architecture—integrated ZigBee sensing devices with IPv6 [19]

intelligent home monitoring systems like [6–11]. Reliability, throughput and jitter in terms of packet transmission in relation to Internet of Things are discussed in the results.

2 System Description

In this section, we describe the implementation of our work. Figure 1 shows the layout depicting key elements of the ZigBee integrated platform. It consists of (i) Smart sensing devices [6, 10], (ii) Wireless Router and (iii) Server.

The ZigBee WSN comprises of XBee-S2 modules built by Digi are configured as end devices (sensor nodes) are connected to a coordinator in the form of mesh topology. In the present setup, heterogeneous sensing devices designed and developed indigenously for Intelligent home monitoring systems like [6–11] are used to integrate with IPv6 networks. The following sections describe about these entities:

A. Sensing Units

We have used four different types of sensing units for effective data management on the integrated networks. This implies that data transmission is of variable length data formats.

Type #1 sensing unit measures the attributes of a hot water system in home. This unit comprises of a hot water cylinder that heats the water from mains electricity and a solar heater that uses sunlight to heat the water.

Fig. 2 Hot water system monitoring—light intensity and temperature monitoring [19]

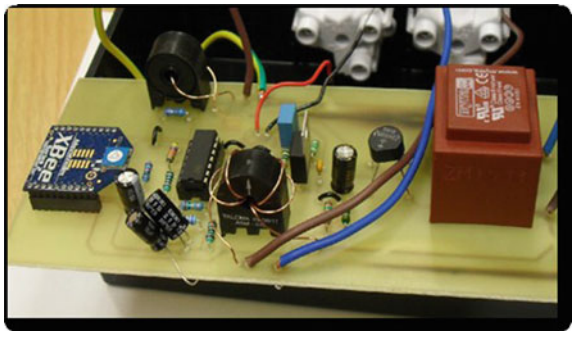


Fig. 3 Household electrical appliance monitoring [19]

The Type #1 sensing unit measure various parameters related to the hot water system:

- A sensor that measures water temperature in a hot water cylinder and solar heater.
- A sensor that measures the ambient temperature around the hot water cylinder and the approximate current consumption of the hot water cylinder.

Type #2 sensing unit measures the humidity, light intensity and temperature outside the home. Figure 2 shows the sensing unit used to measure the light intensity and hot water system temperature monitoring.

Type #3 sensing unit measures the power consumed for a particular electrical appliance connected. In the present case we have connected a water supply pump. Figure 3 shows the sensing unit used to monitor the usage of a household electrical appliance.

Type #4 sensing unit consist of three temperature sensors measuring ambient temperature of a room. The purpose of incorporating three temperature sensors on the sensing unit is to analyze the variations and degradation of the sensors.

The Fig. 4 shows the fabricated temperature sensing unit used for monitoring ambient room temperatures.

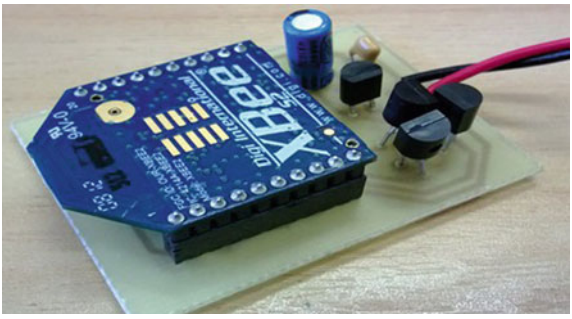


Fig. 4 Temperature sensors fabricated on single board

The coordinator is connected to a router with a wired serial connection. The router runs an open source embedded Linux (OpenWRT) [20]. This essentially provides internet access to the Xbee-S2. The router acts as an application gateway and interconnects the IPv6 and ZigBee network.

An internet connection is established with an Ethernet connection from the router to the server. A private IPv6 network is used for connecting to the local network. The server collects sensor data forwarded by the application gateway and stored in a database for further processing and then to be viewed via a website that can display data in terms of previous day, week, and month time periods graphically.

B. Application Gateway

Developed program for transformation of sensing information between the ZigBee and IPv6 network is executed by the application gateway, as the ZigBee network does not have the architecture to communicate with internet protocols. The application gateway provides this architecture by transforming ZigBee addresses and encapsulating data payloads in an internet protocol.

The XBee-S2 modules produce sample packets which are converted by the application gateway to IPv6 User Datagram Protocol (UDP) packets and sent to a server. Command packets to control the XBee-S2 modules are encapsulated in a UDP packet by the server, and converted by the application gateway to ZigBee packets.

C. Display on Web Interface

A windows based server collects sample data by receiving the UDP packets containing sample data from the application gateway and store in a database. These samples can be accessed from the database from a website hosted on the server.

The raw sample data, sample source (channel and sensor node) and time of arrival are stored in the database. This enables the samples to be ordered by date and organized by their source.

Sample data is displayed on the website in time series graphs; each graph represents the corresponding sensing input information specifically depicting each parameter values. The website performs conversion of raw sample data to engineering units for display in these graphs. Data can be viewed in terms of different time periods like previous day, week and month.

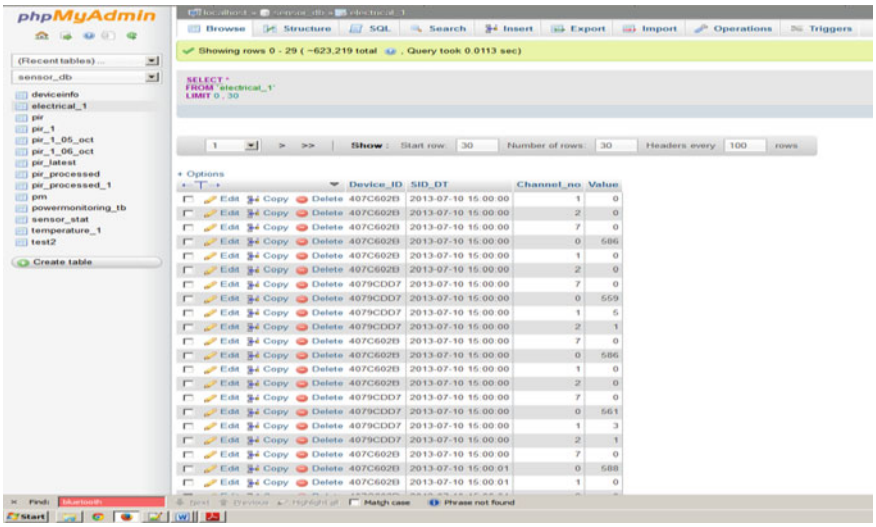


Fig. 5 Screen shot of the database server showing the electrical sensor data

3 Implementation Details

The address transformation is performed by the application gateway for determining the source or destination address of a packet that encapsulates a ZigBee packets' payload. The corresponding application gateway performs the address transformation mechanism for ZigBee to address non ZigBee nodes. The following paragraph provide the details

ZigBee is based upon the 802.14.5 protocol which uses a 64 bit address for each node on a PAN, and 16 bits to identify the PAN ID [802.15.4-2003]. IPv6 uses 128 bits to address a node on the network, of which 48 bits represent the network, 16 bits represent the local network (PAN ID), and 64 bits represent the host id (sensor node). Therefore, the node address for 802.15.4 can placed in an IPv6 address, and the PAN ID can be used to identify the ZigBee network in an IPv6 address. Figure 5 shows the address transformation of ZigBee and IPv6 packet [19].

The packets from the XBee-S2 network are sent to a server using a tunnelling technique, where the addressing information is removed and placed in the encapsulating protocol. Packets destined for the XBee-S2 network use a stateful translation where the source address is stored on the gateway. This enables to reply packets from the XBee-S2 network to be sent to the correct address [19].

A serial interface is used to transmit Application Programming Interface (API) packets from/to the coordinator and router. The WRT54GL router has two serial ports—one of which is used to connect to the XBee-S2 coordinator. The router performs the conversion of the XBee- S2 API packet to an IPv6 packet [19].

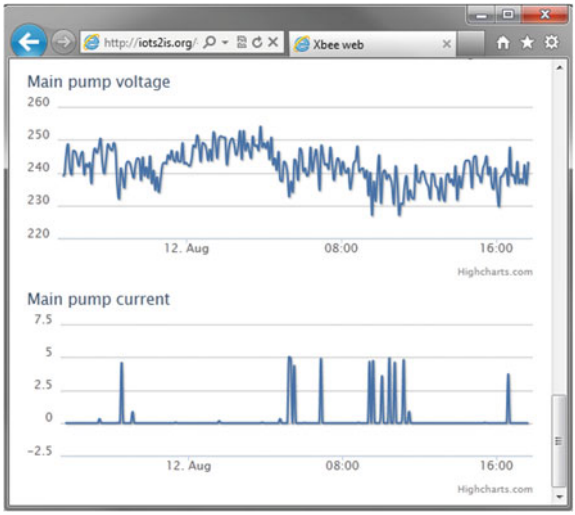


Fig. 6 Measurements of the voltage and current at the water supply pump

The Xbee-S2 modules provide architecture to sample an analogue value and transmit a packet containing this sample to a coordinator. The coordinator outputs an API packet on its serial interface each time a sample packet is received. The API packet contains; the type of API packet, the address of the node that took the sample, the PAN ID of the network the node is on and the sample data. To enable transmission over the internet the API packet is translated into an IPv6 UDP packet [19].

The address is translated using the method discussed in Sect. 4 and the sample data is encapsulated as the UDP payload. The source and destination port of the UDP packet is arbitrarily set in order to identify the UDP packet as containing sample data. Figure 6 shows the corresponding schematic of ZigBee packet transformation to IPv6 format.

The transformation from IPv6 to an API packet requires a stateful conversion. This implies that the source address of the packet must be stored on the router as there is no method of including it in the API packet.

The Linux-Open WRT provides the networking architecture to participate in many types of networks. These networks are abstracted into devices, which generalizes management and configuration. This abstraction requires device drivers which operate in the kernel space, making development difficult. A tun/tap device driver acts as a virtual network device with its output is directed to a user space program instead of a physical device. This simplifies the development of a network device as a user space program is easier to develop.

IPv6 can be used with a tun/tap driver provided the kernel has IPv6 support. This means a virtual IPv6 network can be created, where packets destined for this network are routed to a user space program. IPv6 packets can also be created and sent via the tun/tap driver and will appear to originate from the virtual network.

This virtual network can be mapped to the ZigBee pan using the address transformation and packets are transformed. Access to the tun/tap driver is provided through the standard input output routines. These are the same routines used for the serial interface implying multiplexing between the serial interface and tun/tap interface can be done by the Linux kernel. This eliminates the need for complex threading [19].

D. Storage of data
UDP packets encapsulate sample data to be sent to windows based server. An application on the server receives UDP packets on an arbitrary port, and stores the relevant information in the background of MySQL database.

The database table has four columns; source address, time, source channel and sample data. Rows are added to this table for each UDP packet received. This allows samples to be sorted by time, sensor node and sensor channel. Figure 5 shows the screen shot of the database server.

4 Experimental Results

The developed system is tested by installing the Smart sensing units and setting up a ZigBee based WSN at an elderly house. Interconnecting ZigBee network with IPv6 network is performed by connecting and configuring the modified router as discussed in Sect. 4. Integrated system was continuously used and generated real-time graphical representation of the sensing information on request.

In the present system, programs for address, packet transformations and data transmission are written using 'C' programming language, programs for packet reception and data storage are written using 'C#' and Web interface is developed using PHP Script and Java Scripts.

The graphical representation of different types sensing unit's information in real-time on the website are discussed as follows. Measurements related to sensing unit type #1 and type #2 hot water and solar heating systems and the type #3 sensing unit-water supply pumps are considered. The temperature measurements taken at the hot water cylinder and the solar heater, and the power consumption of the hot water cylinder are depicted. From the graphs it is inferred to determine when hot water was being used and the reduction in power consumption when the solar heater is active.

The Fig. 6 shows measurements of the voltage and current at the water supply pump. The voltage measurement has large amount of noise, and is due to several factors such as ungrounded inputs, insufficient filtering and power supply fluctuations.

The Fig. 7 shows measurement of three same type of temperature sensor fabricated on a single board to study the variation and degradation of the temperature sensors. It was clearly observed that the temperature sensors are not shown any variation from the mean in the collected readings.

The Fig. 8 depicts the correlation of typical household usage scenario of an inhabitant. The parameters measured are: the temperatures of the hot water cylinder and

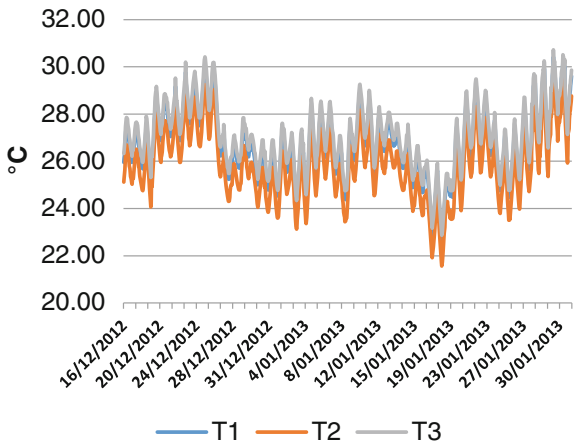


Fig. 7 Measurements of three same types of temperature sensor readings

solar heater, the power consumption of the hot water cylinder and the light intensity outside the home. These parameters allow for analysis of the water heating system.

The times when hot water was used can be clearly identified when the water cylinder temperature decreases, subsequent increases in power consumption by the water cylinder can be seen. These times are highlighted in blue. The light intensity measurements give an indication of when there is an adequate amount of sunlight to heat the water in the solar heater. The first yellow section shows that light intensity was too low to heat the water, as this was an overcast day. The second yellow section shows that the light intensity was significantly higher and therefore the solar heater heated the water. The temperature of water in the solar heater does not exceed the water cylinder temperature due to a circulation pump providing colder water.

The Fig. 9 shows the ambient temperature of a room over a continuous period of 45 days. It is observed that the day light temperatures are higher and night time temperatures are lower when compared with the average temperature of a day. This is a real-time graphical depiction of ambient readings of a room in a smart home.

Real time data storage compression: The real-time data storage for the temperature sensor recordings is done by considering the variations on the successive temperature readings. If there is a change in the consecutive temperature readings then the new reading is stored thereby reducing the amount of sensor recordings to be stored. This technique of variable sensor data recordings has improved the performance for plotting the real-time graphs on the web significantly. This is mainly because the client browser is receiving fewer points of data to be plotted. Table 1 shows the amount of compression achieved for each sensor devices, which was an average of 78 % less stored data.

Quality of Service (QoS) of the integrated ZigBee and IPv6 System: The data from two sensors of hot water system for the duration of a month was analysed to determine the reliability, throughput and jitter of the system. The Xbee-S2 devices were configured to send samples every 10 s. The arrival time of these samples on the

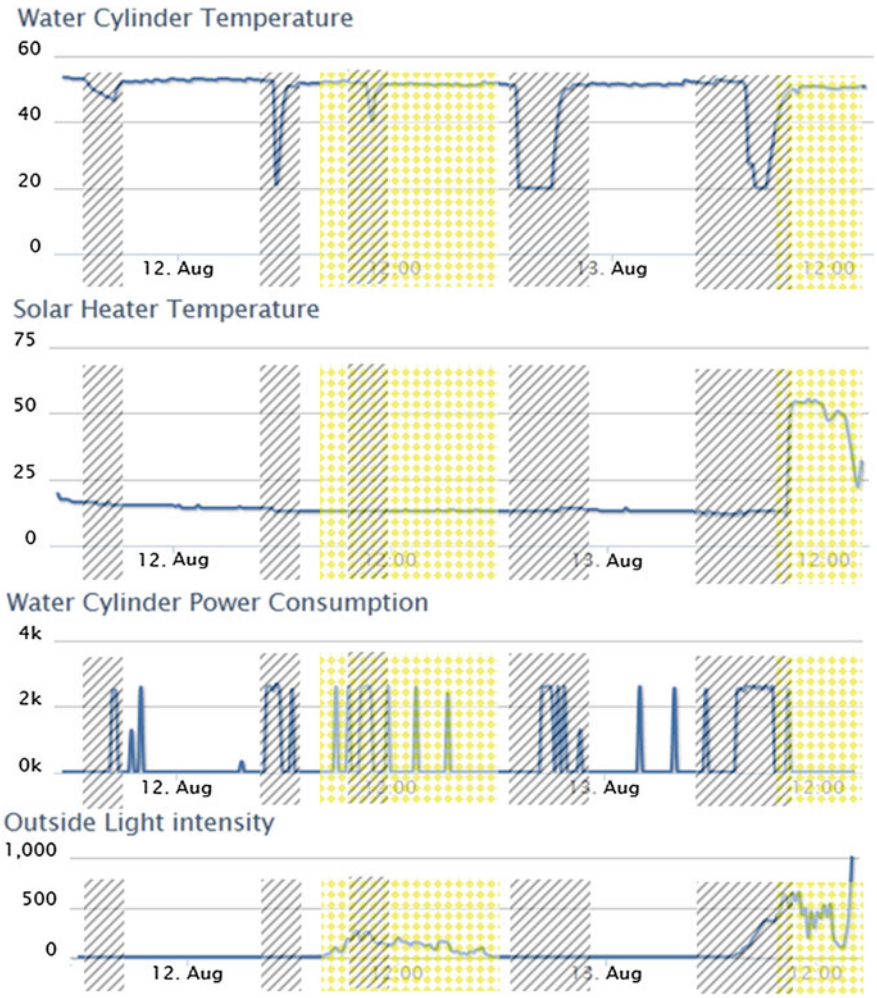


Fig. 8 Graphical representation of the various sensor data on the website

server are recorded in the database accordingly. The time between these recorded times is the interval at which the Xbee device is sending sample information. The total amount of sample information received by the server was calculated by dividing running time of the system with the sample interval.

Reliability: The reliability of the system was determined by comparing the calculated value with the amount of sensor information received correctly. The difference between arrival times of successive sensor information gives the interval value. If the time interval is greater or less than 10 s then there was an error. When the interval is less than 10 s then the sample information received was incorrect or duplicated and therefore it is erroneous. When the interval is greater than 10 s sample information

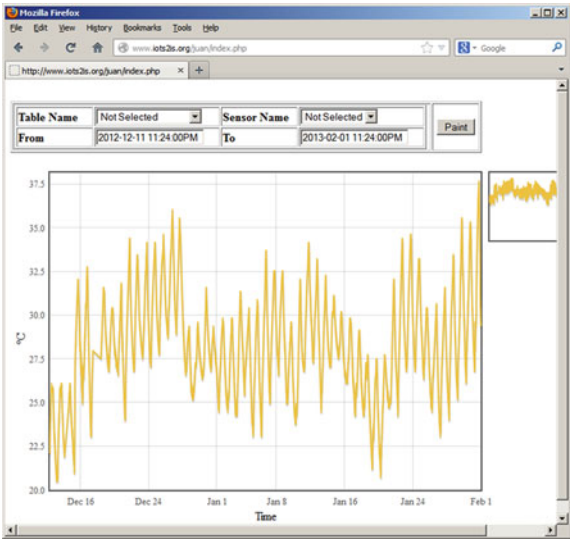


Fig. 9 Ambient temperature readings of a room displayed on the web

Table 1 Total and variable sensor temperature recordings

Sensing unit	Channel #	Sample packets received	Samples stored	Compression (%)
Module 89	T1	74836	11277	84.93
	T2	74836	10576	85.87
	T3	74836	8555	88.57
Module 50	T1	74866	19262	74.27
	T2	74866	16377	78.12
	T3	74866	15552	79.23
Module 94	T1	74932	21166	71.75
	T2	74932	22032	70.6
	T3	74932	20568	72.55
Average				78.43

has been lost, the amount of information lost can be determined by dividing the greater interval by the expected interval.

Throughput: The throughput of the sensing module is the amount of data sent from the sensing module to the server in a given time period [21]. The amount of data in each sample packet was 16 bytes, which is sent every 10 seconds. Therefore the throughput of the sensing module was 1.6 bytes per second. To measure the throughput of the sensing module, the number of packets received in a 5 min time span was considered. The throughput was obtained by dividing the time interval of 5 min. Figure 10 shows the throughput of a sensing unit for the period of 1 month.

The average throughput is 1.55 bytes/s since the reliability was 97 %. Investigation into the sporadic significant change in throughput is in process. Causes are likely to be the interferences from other networks such as Wifi.

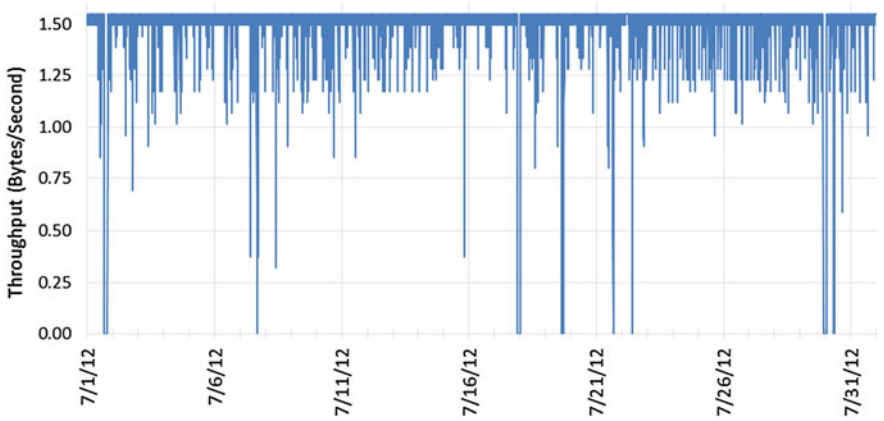
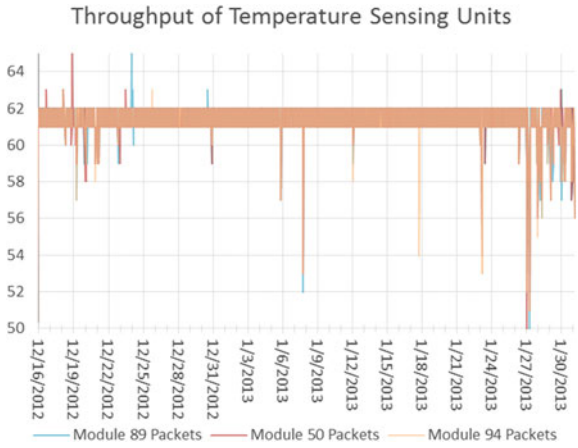


Fig. 10 Throughput of an electrical sensing unit for a period of one month

Fig. 11 Throughput of temperature sensor units



The throughput for **temperature sensor units** was measured by considering the number of packets received in an hour. Figure 11 shows the throughput of the temperature sensors for a period of 45 days. The average throughput was 61.3 packets per hour this is because the XBee module timers are inaccurate, as they were set to 1 min interval. The reason for throughput above average is either due to: XBee module resetting and resending or the application gateway duplicating packets. The reason for throughput below the average is either due to: XBee module failing to send packet or application gateway failing to send packet.

Jitter: The jitter was measured as the delay between two consecutive packets following the technique as mentioned in [22]. Figure 12 shows the jitter of sensing data for a period of one month. The instances of large jitter relate to the correlated low throughput. The other QoS parameters such as delay, energy consumption measurements are in the trial stage.

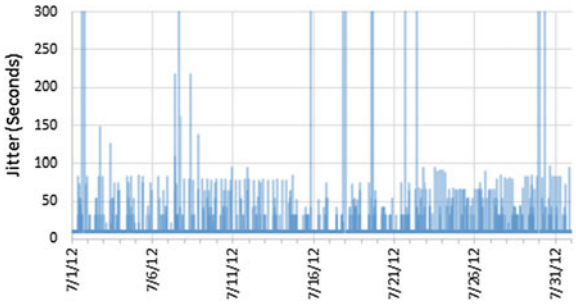


Fig. 12 Jitter for a sensing unit for a period of 1 month

5 Discussion and Future Work

With the advancements in technology, it is expected that the availability of internet is everywhere and online at all time. Low-cost smart sensor node development enabled things to be connected easily and corresponding information can be accessible globally. With the features of scalability, fault tolerance and effective power consumption of nodes and transceiver "Internet of Things" have facilitated ubiquity computational ability to internetwork heterogeneous smart devices easily and facilitate availability of data anywhere. In this chapter, we propose an efficient method for internetworking of 802.15.4 with IP network. The key idea of proposed method is to provide low-cost solution and flexible connection mechanisms for integrating Internet of things with home monitoring systems. The advantages of the developed system are to have greater control over routing of packets (security and customization) and ability to adapt to other wireless sensor networks.

Issues like availability of IPv6 connectivity may be major concern in implementing the methods as discussed in this chapter. As most of the internet domains still operate using IPv4 and IPv6 adoption is low. Better compression techniques can be implemented for minimizing storage requirements and effective retrieval of data. In the next step, sensing units will be interconnected using 6LowPan network setup for better reliability and effective data transmission. Also, Security Issues and study related to comparison of ZigBee based data transmission and 6LowPan network for Smart Home environment with heterogeneous smart sensors will be performed.

It was observed that collecting multiple sensor data over long periods of time generates a huge amount of data. In order to reduce the amount of data stored without losing important information we have implemented the variable sensor data recording technique. This has resulted in significant performance improvement for the real-time data display. The next step will be investigating an appropriate model for continuous heterogeneous sensor data storage mechanism for the research tasks as mentioned in [23–25].

References

1. Surie, D., Laguionie, O., Pederson, T.: Wireless sensor networking of everyday objects in a smart home environment. In: Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing- ISSNIP- 2008, pp. 189–194 (2008)
2. Vision and challenges for realizing the Internet of Things, European Union 2010, ISBN 9789279150883
3. Internet 3.0: The Internet of Things. © Analysys Mason Limited 2010
4. Baronti, P., Pillai, P., Chook, V.W.C., Chessa, S., Gotta, A., Hu, Y.F.: Wireless sensor networks: a survey on the state of the art and the 802.15.4 and ZigBee standards. Comput. Commun. **30**(7), 1655–1695 (2007)
5. Lu, C-W., Li, S-C., Wu, Q.: Interconnecting ZigBee and 6LoWPAN wireless sensor networks for smart grid applications. In: Proceedings of the 5th International Conference on Sensing Technology (ICST)- 2011, pp. 267–272
6. Suryadevara, N.K., Mukhopadhyay, S.C.: Wireless sensor network based home monitoring system for wellness determination of elderly. IEEE Sens. J. **12**(6), pp. 1965–1972 (2012)
7. Suryadevara, N.K., Gaddam, A., Rayudu, R.K., Mukhopadhyay, S.C.: Wireless sensors network based safe home to care elderly people: behaviour detection. Elsevier Sens. Actuat. A Phys. **25**, 96–99 (2012). http://dx.doi.org/10.1016/j.sna.2012.03.020
8. Ranhotigamage, C., Mukhopadhyay, S.C.: Field trials and performance monitoring of distributed solar panels using a low cost wireless sensors network for domestic applications. IEEE Sens. J. **11**(10), 2583–2590 (2011)
9. Kaur, K., Mukhopadhyay, S.C., Schnepper, J., Haefke, M., Ewald, H.: A ZigBee based wearable physiological parameters monitoring system. IEEE Sens. J. **12**(3), 423–430 (2012)
10. Alabri, H.M., Mukhopadhyay, S.C., Punchihewa, G.A., Suryadevara, N.K., Huang, Y.M.: Comparison of applying sleep mode function to the smart wireless environmental sensing stations for extending the life time. In: Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC)-2012, pp. 2634–2639 (2012)
11. Mendez, G.M., Yunus, M.A.M., Mukhopadhyay, S.C.: A WiFi based smart wireless sensor network for monitoring an agricultural environment. In: Proceedings of IEEE I2MTC 2012 conference, IEEE Catalog number CFP12MT-CDR, pp. 2640–2645. Graz, Austria, 13-16 May 2012. ISBN 978-1-4577-1771-0
12. Howitt, I., Gutierrez, J.A.: IEEE 802.15.4 low rate -wireless personal area network coexistence issues. Proc. IEEE Wirel. Commun. Netw. WCNC **3**(20), 1481–1486 (2003)
13. Angrisani, L., Bertocco, M., Foortin, D., Sona, A.: Assessing coexistence problems of IEEE 802.11b and IEEE 802.15.4 wireless networks through cross-layer measuremennts. In: Instrumentation and Measurement Technology Conference Proceedings, 2007, pp. 1–6. IMTC 2007-IEEE, 1–3 May 2007
14. Khaleel, H., Pastrone, C., Penna, F., Spirito, M.A., Garello, R.: Impact of Wi-Fi traffic on the IEEE 8022.15.4 channels occupation in indoor environments. In: International Conferrence on Electromagneticcs in Advanced Applications, 2009 ICEAA '09, pp. 1042–1045, 14–18 Sept 2009
15. Khoshdelniat, R., Sinniah, G.R., Bakar, K.A., Shaharil, M.H.M., Suryady, Z., Sarwar, U.: Performance evaluation of IEEE802.15.4 6LoWPAN gateway. In: Proceedings of the 17th Asia-Pacific Conference on Communications (APCC)- 2011, pp. 253–258 (2011)
16. Kushalnagar, N., Montenegro, G., Hui, J., Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (2007)
17. Lee, K.H., Won, J.H., Bae, C.S.: IGN2IP: Internetworking intelligent gadget network based on 802.15.4 with IP network. Proceedings of the International Conference on Consumer Electronics, ICCE 2008, Digest of Technical Papers, pp. 1–2 (2008)
18. Mesrinejad, F., Hashim F, Noordin, N.K., Rasid, M.F.A., Abdullah, R.S.A.R.: The effect of fragmentation and header compression on IP-based sensor networks (6LoWPAN). In: Proceedings of the 17th Asia-Pacific Conference on Communications (APCC)- 2011, pp. 253–258 (2011)

19. Kelly, S.DT., Suryadevara, N.K., Mukhopadhyay, S.C.: Towards the implementation of IoT for environmental condition monitoring in homes. IEEE Sens. J. **13**(10), 3846–3853 (2013). doi:10.1109/JSEN.2013.2263379
20. OpenWRT website. http://openwrt.org/
21. Wang, Y.: Probabilistic QoS analysis in wireless sensor networks. Ph.D dissertation, Department of Computer Science and Engineering, University of Nebraska - Lincoln, Nebraska (2012)
22. Zakaria, A.: Quality of service in wireless sensor networks. http://cs.uwindsor.ca/richard/cs510/survey_zakaria.pdf. Accessed 06 Oct 2012
23. Suryadevara, N.K., Mukhopadhyay, S.C., Wang, R., Rayudu, R.K.: Forecasting the behavior of an elderly using wireless sensors data in a smart home. Eng. Appl. Artif. Intell. **26**(10), pp. 264–2652 (2013). ISSN 0952–1976, http://dx.doi.org/10.1016/j.engappai.2013.08.004
24. Suryadevara, N.K., Mukhopadhyay, S.C., Wang, R., Rayudu, R.K., Huang, Y.M.: Reliable measurement of wireless sensor network data for forecasting wellness of elderly at smart home. In: Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE, International, pp. 16–21, 6–9 May 2013. doi:10.1109/I2MTC.2013.6555372
25. Survadevara, N.K., Mukhopadhyay, S.C., Rayudu, R.K.: Applying SARIMA time series to forecast sleeping activity for wellness model of elderly monitoring in smart home. In: Sixth International Conference on Sensing Technology (ICST), 2012, pp. 157–162, 18–21 Dec 2012. doi:10.1109/ICSensT.2012.6461661

Towards Autonomous Wireless Sensors: RFID and Energy Harvesting Solutions

Y. Duroc and G. Andia Vera

Abstract In this chapter, an extend vision on the sensing techniques and powering on smart and autonomous RFID tags is presented. In a society leaded by the information and networking, a link is defined between our real world and a virtual scenery for all the everyday objects. The linker is a smart wireless sensor. No better definition can be done by exploiting the real meaning of the words: This is the "Internet of the things era". The most suitable technology for the development of smart sensors is the passive RFID technology because its battery-less operation, network interoperability and wireless capability. Several challenges are faced on the design of these smart and autonomous RFID sensors: sensing techniques, structure considerations and wireless powering are the main challenges discussed in this chapter. The power autonomy is presented under harvesting techniques with special interest on the electromagnetic energy harvesting. Design criteria of electromagnetic energy harvesters are also discussed. Some examples of applications on the most common sensed parameters including physical, biomedical, automatic product tamper detection and noninvasive monitoring are described.

1 Introduction

In the literature, the Internet of Things (IoT) concept is defined in different ways although enough similar. The definition given by the CASAGRAS consortium is maybe the most general because it refers to a Things oriented vision and an Internet oriented vision [1]: "A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as

Y. Duroc (✉) · G. Andia Vera
University of Grenoble Alpes, LCIS, Valence 26900, France
e-mail: yvan.duroc@esisar.grenoble-inp.fr

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7_10,

the basis for the development of independent cooperative services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability." The US National Intelligence Council provides a definition currently in use, and maybe simpler to apprehend [2]: "The IoT is the general idea of things, especially everyday objects, which are readable, recognizable, locatable, addressable, and controllable via the Internet—whether via RFID, wireless LAN, wide-area network, or other means." Therefore the IoT include Internet-connected devices, smart connected devices, Wireless Sensor Networks (WSN), machines and devices communicating wirelessly, ubiquitous computing, ambiance intelligence, and smart matter. In this context, the association of WSN and RFID (Radio Frequency Identification) systems would enable a lot of new applications.

The RFID technology appears as a relevant technology with the convergence of sensing and identification features and presents multiple capabilities in terms of communication, identification, localization, tracking, sensing, etc.; and also it has intrinsic advantages in terms of low power and low cost solutions. The evolution of the RFID with this objective will lead to RFID sensor networks very adequate in the IoT context. In particular two interesting options are possible: passive RFID systems using tags which collect the energy from the signal transmitted by a reader for powering the chips; and chipless RFID systems which present fully passive tags.

Section 2 presents the RFID technology and its variants with the objective to develop sensing RFID systems. Section 3 introduces the energy harvesting approaches with a focus on electromagnetic power harvesting devices. Section 4 gives a conclusion and future research hints.

2 Evolution of the RFID as IoT Technology

2.1 General Presentation of the RFID

RFID technology allows data storage in small electronic transponder circuits that are particularly portable [3, 4]. Communication between an RFID tag and an interrogator is realized by radio frequency (RF) waves. In consequence, the data can be read and written without contact and often through obstructions. A typical RFID system is illustrated by Fig. 1. Three fundamental components constitute RFID systems:

- Transponder (or RFID tag), which contains the identification code, and are permanently or temporally attached to objects.
- Interrogator (or reader), which sends interrogation signals to an RFID tag in order to identify it. Its complexity and configuration depend on the functions to be fulfilled, which can vary quite significantly from one application to another. However, the main reader's function is to provide the communication way to the RFID tags nd to facilitate the data transfer by accomplishing a specific protocol.

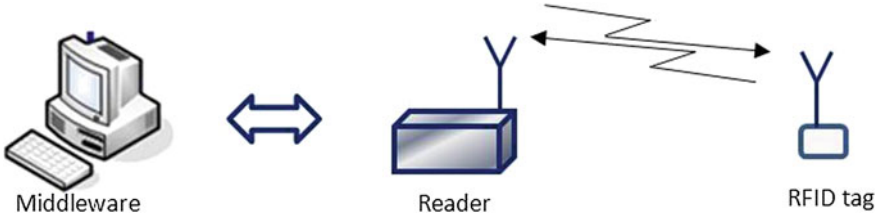


Fig. 1 Architecture of a classic RFID system

- Middleware is the software installed in a host computer that manages the interrogator or reader tasks, it is the responsible to converting tag data into meaningful information, and to ensure the integrity of the data obtained. The middleware often offers additional data processing and internet connection for a global connectivity

In a typical RFID application, RFID tags are attached or embedded into objects locking for identification or tracking. In the most frequent application such as supply chain management, the RFID tags simply serve the purpose of UPC (Universal Product Code) bar codes. However compared to bar codes, solutions based on RFID tags have much better capabilities as: non-line-of-sight communication, writing capabilities, multiple reading, durability, security; however their cost is still much higher. Moreover when combined with sensors, RFID can also help managing objects that are environmentally sensitive (e.g. food, medicine, blood) or non-sensitive (durable products and machines).

Although RFID technology is always in evolution and development, several techniques and standards exist. Table 1 summarizes the properties of the conventional RFID technologies. For each technology, used frequency range, distance range, coupling type, existing standards (mainly defined by the International Standard Organization, ISO), and traditional RFID market segments are presented.

According to the powering method, an alternative and complementary classification can be done: active, passive and semi-passive tags. Active tags embed an internal battery which continuously powers the tag. Contrary, passive tags have no internal power supply. They get their energy from the RF carrier wave continuously provided by the reader during the communication. The communication consists of a query sent by the reader, that one that the tag responses by backscattering a signal, i.e. part of RF power received from the carrier is transmitted back to the reader with an appropriative modulation and coding performed by the logical part of the RFID chip. Passive tags are the most commonly used tags. They have a shorter read range compared to active tags but they have a smaller size, are cheaper and do not require any maintenance. Semi-passive tags communicate with the readers like passive tags (using scattering principle) but they are equipped with an internal battery constantly powering their internal circuitry. Finally and more recently, a new RFID solution called chipless uses purely passive tags, which potentially means an infinite lifetime. The chipless tags do not rely in any powering to operate and present a fixed electromagnetic signature that allows its remote identification. However the development of chipless

Table 1 Summary of properties RFID techniques

	Frequency range	Distance range	Coupling	Existing standards	Applications
LF	125 kHz	∼0.1 m	Magnetic	11784/85, 14223	Smart card, ticketing, access, animal tagging, laundry
HF	13.56 MHz	∼1 m	Magnetic	18000-3.1, 15693, 14443A, B, C	Small item management, supply chain, anti-theft, library
UHF	900 MHz	∼2–7 m	Electromagnetic	EPC C0, C1, C1G2, 18000-6	Transportation vehicle ID, access/security, supply chain, large item management
Microwave	2.4 GHz	∼10 m	Electromagnetic	18000-4	Transportation vehicle ID (road toll), access/security, supply chain, large item management

solutions requires new functioning principles and adequate standard definitions to be commonly exploited [5].

Today the RFID technology is present in the industrial world with diverse applications from access control fields until logistics and automatics [6]. If the RFID is known for its current applications, it also offers promising applications for the future sensor networks and the IoT. Indeed the same mechanisms that allow the communication and identification between RFID reader and tags can also be applied to collect sensor data. The RFID-based sensor response is electromagnetically sensitive to the dielectric of its close medium and enables environmental sensing applications besides the identification, such as presence detection, gas detection, temperature detection, moisture detection or physical strain. This sensibility is analysed by processing the electromagnetic signature of the tag antenna. EPGGlobal (worldwide governing entity for RFID standards) has envisioned the architecture IoT in which devices with RFID tags dispersed through the Internet can communicate with each other, providing real-time information about their location, contents, destination and ambient conditions [7]. The potential attractiveness of RFID that enables such vision is largely based on the key advantages of RFID components relative to other technologies: small form-factor tags that afford a potentially long life, resulting from optimized low power operations and energy harvesting.

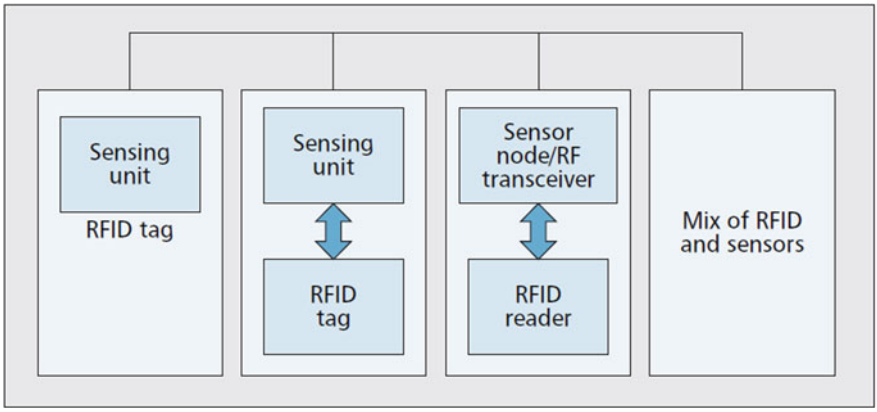


Fig. 2 Four type of integration [8]

2.2 RFID Sensors

RFID and WSN technologies

RFID and WSN technologies are two complementary technologies. Combining both technologies it is offered a great number of advantages. Indeed, RFID tags can replace some of the sensor nodes in WSNs and offer cheaper solutions. In addition, RFID technology provides the possibility of tracking objects. Alternatively sensors can provide various sensing capabilities to RFID tags, introducing logic into nodes to enable RFID readers and tags a certain level of intelligence, and afford the ability of operating in a multihop fashion extending potentially RFID applications. Several scenarios can be envisaged for combining RFID and WSNs (Fig. 2), such are: tag integration with sensors, tag integration with WSN nodes and wireless devices, reader integration with WSN nodes and wireless devices, and a mix of RFID and WSNs [8].

Thus, the integration of RFID and WSNs opens up a large number of applications in which it is fundamental to sense environmental conditions and to obtain additional information about the neighboring objects.

Sensing antenna

The identification information may be enhanced with physical information about local agents as well as changes in the tagged object itself. The majority of existing solutions rely on RFID UHF tags and a dedicated specialized sensor. Like RFID tags, wireless sensors can be divided into active (battery-powered devices), semi-passive sensors (battery-assisted) and fully passive sensors. Thus, RFID can be combined with a sensor to monitor different elements in their environment, including temperature, humidity, shock and vibration. However, sensor addition often increases both tag size and cost. An alternative solution for more compactness and cost-efficiency is the functional integration of the antenna and the sensor component in order to obtain

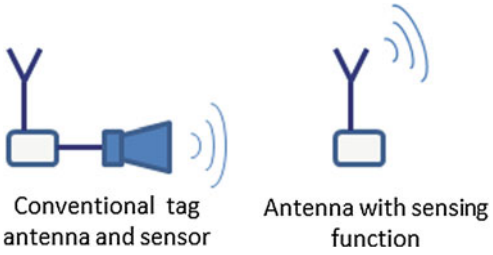


Fig. 3 Configuration of RFID tag sensing systems

a sensor tag. The challenge is then to use the RFID tag antenna directly as a sensor. The Fig. 3 illustrates these two different concepts: the conventional antenna attached to a sensor versus the antenna with sensing function.

There are three regions of interest around the antenna. The far-field region is the "simplest" case to consider since no significant coupling or perturbation will disturb the radiated field. On the contrary, the near-field region become very sensitive due to significant coupling that can exist between the antenna and any other element located in this region. Then this characteristic can be exploited in order to transform an antenna into a sensor. To do that the design should establish the relationship between the parameter to be sensed and one of the antenna parameters. Even if this seems to be challenging and difficult in term of analysis, many experimental results have demonstrated this capability. As an example, consider a RFID tag located close to a metallic surface. The proximity effect will modify the radiation pattern of the antenna and consequently the impedance of the antenna will change. This effect could be used to design a displacement sensor as demonstrated in Bhattacharya et al. [9]. The same effect, but with high permittivity dielectric, will lead to detune the tag antenna and consequently transform the tag into sensor too [10].

Examples

Several concepts for sensor tags have been proposed with different and various functionalities.

RFID-based sensors offer great promise to the perishables supply chain. The losses in perishable products are estimated around $35 billion annually [11] and it is necessary to highly increase the product visibility. One of the most relevant information is about the evolution of the temperature in the transit operations, and notably in the cold supply chain operations. In Bhattacharyya et al. [12], a temperature sensor is introduced (Fig. 4). The sensor takes into account temperature violations and changes on the tag antenna properties by using a shape memory polymer which is a temperature sensitive. The temperature actuated shape memory mechanism to preserve state changes even in the absence of transmitted power from the reader.

Other environmental information interesting to capture is the humidity. A printed UHF RFID sensor tag that indicates if the tagged object has been exposed to a certain degree of moisture (Fig. 5) in presented in Gao et al. [13]. A printed coupling loop with an embedded resistive moisture sensor is placed above the surface of a tag antenna, thus the changes on the sensor resistance cause variations on the

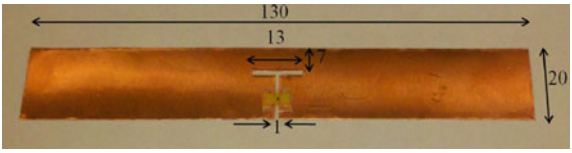


Fig. 4 Prototype of the temperature tag sensor [12]

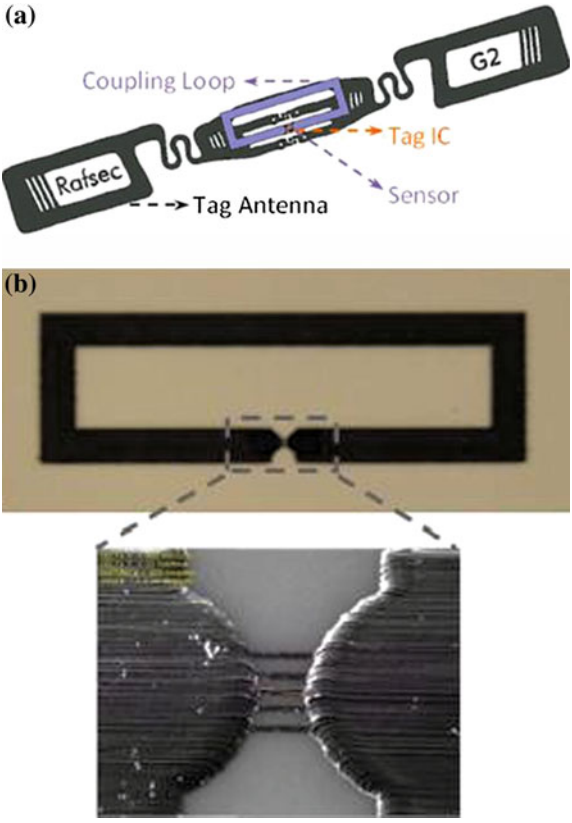


Fig. 5 **a** UHF RFID tag from Rafsec, extended with sensor functionality using electromagnetic coupling [13]. **b** Photograph of the (*upper*) printed coupling loop with the (*bottom*) write-one-read-many structure made of five parallel sensor lines enlarged [13]

tag antenna properties through electromagnetic coupling. It should be noted that the electromagnetically coupled sensor is easy to apply as a sticker or by similar manners, using commercial tags.

A patch antenna with a relative humidity sensing function is proposed in Chang et al. [14] using a modified polyimide. The proposed antenna is a passive device that physically and functionally combines an antenna with a relative humidity sensor (Fig. 6).

Another important application of wireless sensors is the remote monitoring of environments to detect the presence of hazardous gases within industrial sites (supply chain and food integrity) or do the quality control of work spaces. Otherwise, one of the most promising techniques to transform the antenna into a sensor is to

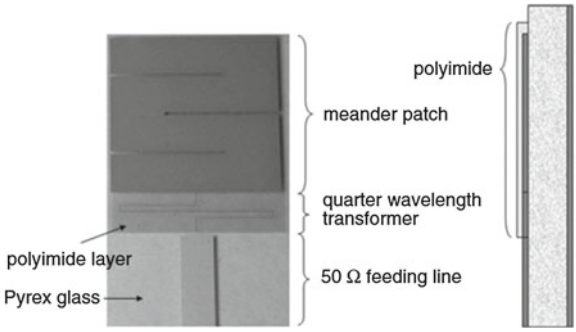


Fig. 6 Configuration of the antenna integrated with relative humidity sensor: *top view* optical photograph (*left*) and side view of structure (*left*) [14]

integrate nanomaterial to antenna. As the nanomaterials could exhibit very high sensitivity to environment parameters (such as temperature, moisture, gas, etc.) so their integration with the tag antenna will allow it to become sensitive to its environment. Some experimental works have already demonstrated this property. For example, a complete integration of carbon nanotube (CNT) film (buckypaper) into the RFID tag aiming to design a passive gas (ammonia gas, NH_3) radio sensor is presented in Occhiuzzi et al. [15]. CNT film is used as a localized and variable resistive load integrated into the tag antenna, which becomes able to transduce the presence of hazardous gas in the environment into changes of its electromagnetic features (Fig. 7).

The RFID also offers other applications capabilities. A "zero-power RFID-enabled threshold shock sensor" is proposed in Todd et al. [16]. This is a latching accelerometer which records an acceleration event above a specific threshold. The presented entire package is designed to provide low cost sensors for monitoring shock loads. The sensor is based on a mechanical bistable mechanism laser-cut from a single Delrin layer which switches between states when exposed to accelerations above a threshold (Fig. 8).

Another great field of application concerns real-time biomonitoring (temperature, blood pressure, heartbeat, glucose content, human behaviour) and location of people within hospitals or domestic environment. However the design of effective wearable tags is difficult due to the strong interaction of the antenna with the human body which is responsible of impedance detuning and efficiency degradation. Occhiuzzi et al. [17] proposes a family of wearable tags based on a particular folded geometry structure presenting a good independence to the proximity of human body. Finally in Lakafosis et al. [18], inkjet-printed flexible antennas, RF electronics and sensors fabricated on paper and other polymer substrates have been presented as system-level solutions for ultra-low-cost mass production of UHF RFID tags and RFID-enabled wireless sensor nodes or even wireless cognition applications (Fig. 9).

Otherwise, MEMS (microelectromechanical systems) technology has been also studied to be implemented in passive wireless sensors or tags. This type of wireless MEMS sensors enables very low manufacturing cost, long reading distance, high frequencies and compact size without requiring embedded electronics. One example

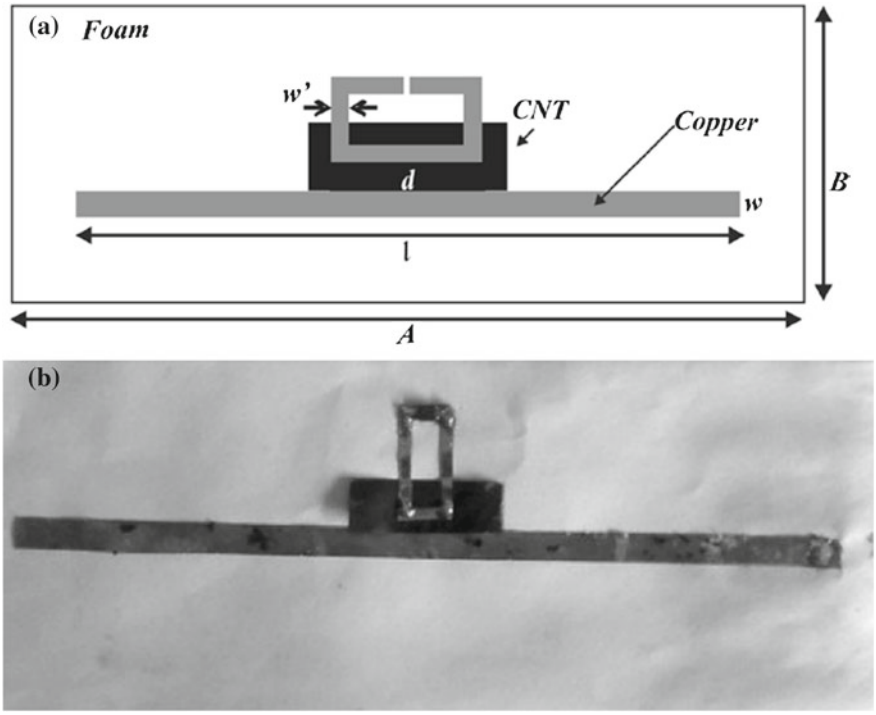


Fig. 7 **a** RFID tag inductively coupled loop with a rectangular CNT load (size in millimeters: $A = 180$, $B = 80$, $l = 160$, $w = 5$, $w' = 2$, $d = 2$) [15]. **b** Photograph of prototype [15]

of such MEMS tag sensors replying at an intermodulation frequency is presented in Viikarai and Seppa [19]. Another approach is to not include any additional sensor or sensitive material on the overall tag but rather relies on the variation that the environmental conditions have on the response of the tag. In Capdevila et al. [20], a passive multiprobe sensor using conventional UHF RFID shows the feasibility of impedance-based sensing with RFID probes. Indeed, any physical, chemical or physiological factors which are known to impact the tag antenna impedance (via the dielectric permittivity) can be used to detect changes in local neighborhood.

Finally the use of RFID technology for the automatic transmission of physical characteristics in WSN opens the way to a large class of attractive applications. However, although some RFID tags enable to transmit sensor-like information are already on the market, only a few sensors are embedded in the tag. Catarinucci et al. [21] presents a new multi-ID and cost effective RFID tag (called S-tag) for sensor data transmission. As shown in Fig. 10, the S-tag transmits the sensor value by standard RFID technology of the digital output of a generic sensor.

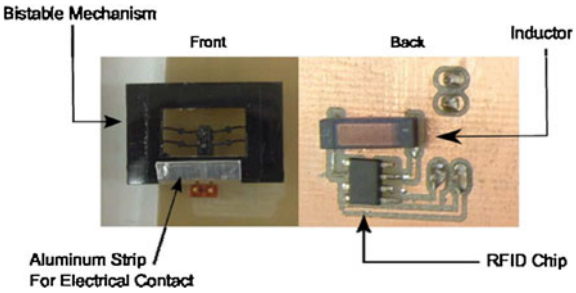


Fig. 8 Working shock sensor prototype, front and back view showing the inductor and RFID chip with the sensor [16]

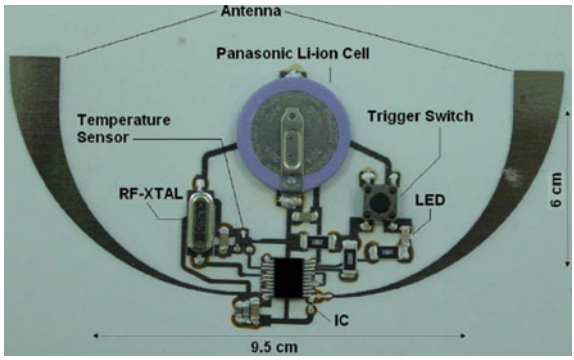


Fig. 9 Inkjet-printed wireless sensor transmitter prototype on paper substrate [18]

Alternative solution: chipless

The main cost of an RFID tag comes from the embedded chip; new solutions present new alternatives using chipless tags without silicon integrated circuit. Chipless tags encoding data can be achieved thanks to three main categories based on time domain reflectometry (TDR), spectral signature encoding and amplitude/phase backscatter modulation. Time domain reflectometry (TDR) based chipless tags are interrogated with a pulse signal and the information is given by the echoes of the pulse sent by the tag. SAW (Surface Acoustic Wave) tags, thin-film-transistor circuit, microstrip tags with discontinuities, are used. Spectral signature-based chipless tags encode data into the spectrum using (multi)resonant structures [22]. Each bit corresponds to a predetermined frequency. Amplitude/phase backscatter modulation-based chipless tags are realized by controlling the reactive loading of the tag's antenna. The Fig. 11 illustrates the layout of the proposed chipless wireless sensor node.

For example, [23] demonstrates the use of chipless tags based on periodic magneto inductive-wave delay lines in order to realize a low cost wireless sensor of temperature or humidity (Fig. 12).

However, chipless technology is in infancy recently and several challenges are facing its development in terms of technology process, coding capacity, miniaturization, sensing capabilities. The development of these chipless (and cheaper) solutions will also require new standards (the two first approaches need larger frequency bands), novel readers integrating pulse generator functions and processing signal capabilities.

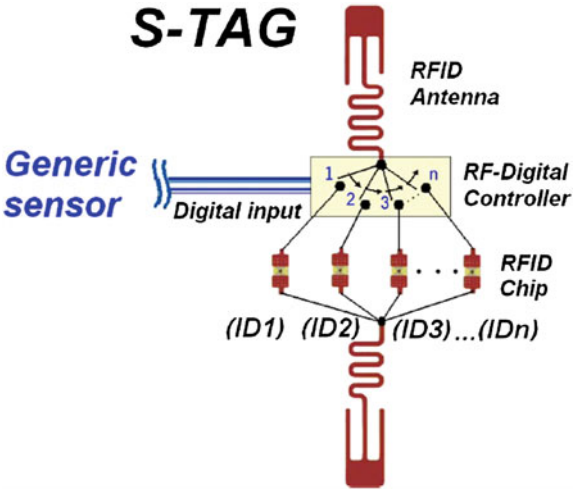


Fig. 10 Simplified scheme of the designed RFID S-tag [21]

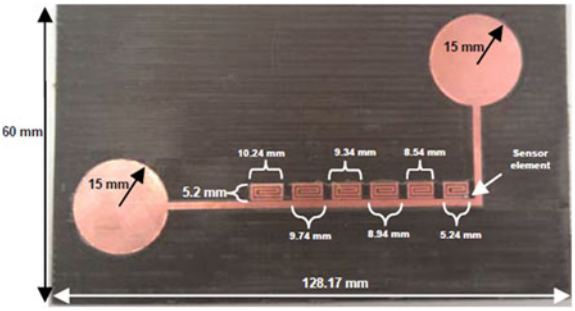


Fig. 11 Layout of chipless wireless sensor node [22]

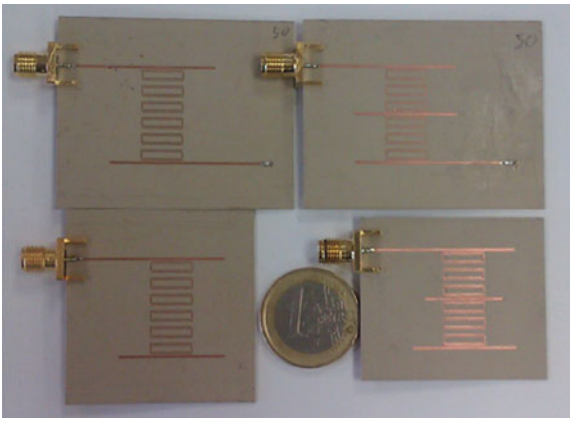


Fig. 12 Fabricated set of two-bit chipless tags [23]

2.3 Conclusion: RFID as Sensor Technology

The RFID tends to become an invisible and ubiquitous sensing technology. Using the automated monitoring capability of the RFID technology in addition to the sensing capabilities based on the near-field coupling dependence has increased the applications in which RFID is deployed. Major RFID sensing application fields include monitoring physical parameters, biomedical sensing and monitoring, automatic product tamper detection, robot direction sensing, and noninvasive monitoring. More particularly, UHF RFID is a promising approach for the development of the IoT. This technology is quite inexpensive, also naturally compatible with Internet and allows a reading range adequate for many applications. In this context, the techniques of scavenging power become one of the key points to resolve with the most possible efficient way.

Harvesting energy from environment is a very rapidly evolving topic. When combined to wireless advantages, harvesting energy could lead to all passive solutions with potentially infinite lifetime. This is why a technology like passive RFID is greatly investigated and considered in numerous domains and applications. As an example, an RFID chip needs only -18 dBm power to be activated. Many existing sources of ambient energy exhibit power in the order of the milliwatt. Antenna is a well-suited tool for electromagnetic power harvesting. Indeed, an antenna loaded by a charge pump circuit obtained by connecting a diode/rectifier and a capacitance, is equivalent to DC generator. In such configuration the optimization should consider the maximum converted power, which is obtained under perfect impedance matching between the antenna and the charge pump circuit. When harvesting and backscattering are considered with the same antenna, a trade-off between the two functions must be achieved during the design.

3 Energy Harvesting Solution

Actual social tendencies are evolving toward creating smart environments where a multitude of sensors and devices are interacting to deliver an abundance of useful information. Essential to the implementation of the IoT is the design of energy efficient systems aiming toward a low-carbon–emission society. Within this context, harvesting solutions appears as an alternative to provide these sensors and devices with self-sustained operation [24].

Table 2 provides an indicative list of attainable harvested values, corresponding to solar, kinetic, thermal and electromagnetic harvesters based on existing products or published results in the literature [25]. Even if the solar power appears as the largest and most commonly available source of ambient power, a challenge of increasing its power conversion efficiency while maintaining a low cost associated with the materials and fabrication conditions is for an extended use [26]. In the case of kinetic harvesters, the major challenge is the design of multiband or frequency

Table 2 Indicative harvested power from different transducers [25]

Energy sources	Harvested power	Condition/Available power
Light/Solar	60 mW	6.3 cm × 3.8 cm flexible solar cell AM1.5 Sunlight (100 mW·cm^{-2})
Kinetic/Mechanical	8.4 mW	Piezoelectric shoe mounted
Thermal	0.52 mW	TEG ($\Delta T = 5.6$ K)
Electromagnetic	0.0015 mW	Ambient power density 0.15 μW·cm^{-2}

tuneable harvesters. Research on the field of thermal energy harvesters consists of optimizing the related devices and circuit topologies in order to increase the conversion efficiency [27]. Electromagnetic harvesters provide the lowest harvested power. However they additionally allow a wireless power transmission (WPT): the capability to intentionally power a sensor device using a dedicate transmitter, which is the case of the passive RFID technology,

The utilization of electromagnetic energy harvesters is supported by the fact that sensors typically operate in a wireless environment, which means that they require the use of antennas and therefore the implementation of electromagnetics harvesters comes with a minimal cost.

3.1 Electromagnetic Harvesters

The concept of WPT is not new; rather it was demonstrated in 1899 by Tesla who carried out his experiments on power transmission by radio waves (Fig. 13). The main lines of historical research and development on wireless power transmission are listed below [28]:

- 1864: James C. Maxwell predicted the existence of radio waves;
- 1884: John H. Poynting defined the Poynting vector to quantify electromagnetic energy;
- 1888: Heinrich Hertz showed experimental evidence of radio waves;
- 1899: Marchese G. Marconi and Reginald Fessenden invented wireless communications via radio waves;
- 1856–1943: Nikola Tesla conducted the first wireless power transmission experiment;
- 1889: Wardenclyfee Tower was proposed by Tesla;
- World War II: Microwave Energy Converter was invented;
- 1940–50s: Photovoltaic Cell was built;
- 1958: US Solar Power Satellite (SPS) was proposed;
- 1964: William C. Brown started the first microwave power transmission;
- 1968: Peter Glaser proposed SPS System;

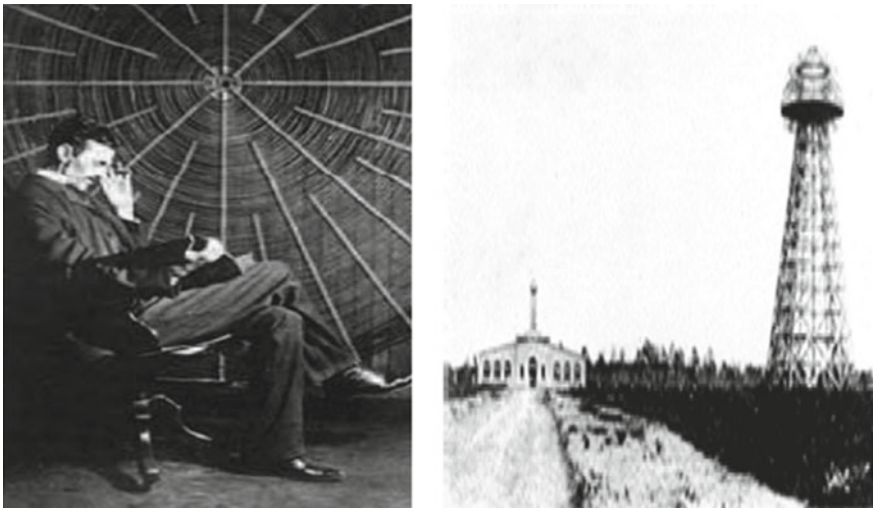


Fig. 13 Nikola Tesla proposed a gigantic coil connecting to a high mast of 200-ft with a 3 ft-diameter ball at its *top*. He fed 300 kW power to the Tesla coil resonated at 150 kHz and reached the RF potential of 100 MV at the top sphere [30]

- 1969: William C. Brown patented the device used to convert electromagnetic power to DC power, called rectenna [29] 1970s Oil Embargo turned out;
- 1978-1981: US Department of Energy Program was supported;
- 1980s: Japanese SPS System started;
- 1987: Canadian Project started;
- 1995: NASA's Fresh Look was conducted;
- 1999: NASA's SERT was supported;
- 1990s: French Grand Bassin–La Reunion was built;
- 2000: Japanese Project and 8 Joint Countries were reported;
- 2012: Chinese Project to be launched (2 national wide meetings were held and white papers were submitted);
- 2025: Low cost model demonstration will be expected.

Was in 1969 when the rectenna, device used to convert RF power to DC power was patented, then establishing the basis for the actual electromagnetic energy harvesting in the IoT era, where the device to be powered are presumed to be sensors integrating simple computing and communication capabilities into common objects of everyday use.

Figure 14 shows a simplified schematic of a WPT scenery. On one side there is a transmitting device that radiates a RF signal at a certain frequency and with certain power. On the other side there is a rectenna formed by an antenna and a rectifier circuit that collects the RF signal and convert it into DC power. One of the key parameters when designing WPT systems is the maximization of the RF to DC efficiency conversion. This will limit the performance of the final powered device.

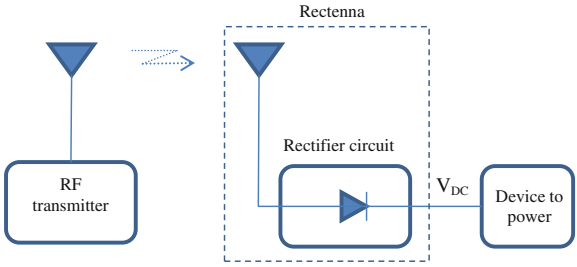


Fig. 14 Simplified schematic of a WPT [24]

Initial applications on WPT were required for a directive and high-power transmission. One of these applications for example is the proposed solar-power satellites in 1958, where the solar energy is captured and converted to electromagnetic signals that can be potentially re-radiated to the Earth as source of power [31]. Much later on, with the interest in autonomous devices, sensors led to the concept of energy harvesting in where rectennas are used to provide DC power by converting the available RF power from existing ambient low-power electromagnetic sources not specially transmitting power to the sensor, in fact powered by electromagnetic energy harvesting.

In general, WPT can be seen as a contactless manner of transferring power to a system in order to power it. There are several classifications that can be made on wireless energy transfer systems attending to the transfer mechanism and the distance between the transmitting source and the device that needs to be powered. One of these classifications divides the WPT mechanisms in two: near field and far field. Near field mechanisms include coupling and resonant inductive coupling. Far field mechanisms include radiation of RF/microwave electromagnetic signals [24]. The electromagnetic energy harvesting or dedicated WPT for WSN and passive RFID technology belong to the far field mechanism.

3.2 WPT in Passive RFID Technology

In recent times, RFID technology is a clear example of wireless power transmission where such a tag operates using the incident radio-frequency (RF) power emitted by the transmitter. A typical far-field passive RFID sensor network consists of one (or more) RFID reader and a number of RFID sensors (called tags), where these communicate data to the reader by modulating, possibly amplifying, and transmitting back a continuous wave (CW) emitted by the reader itself through a process called backscatter modulation [32, 33]. The RF field emitted by the reader is the only source of energy that allows passive tags to activate their circuitries, while more sophisticated classes of tags, referred as semi-active and active, rely on energy storage devices (simply called batteries) charged in a previous phase (see Sect. 2.2).

The architecture of a passive RFID tag is shown in Fig. 15. The radiating antenna is typically a dipole or a patch antenna. A rectifier circuit converts the input alternating

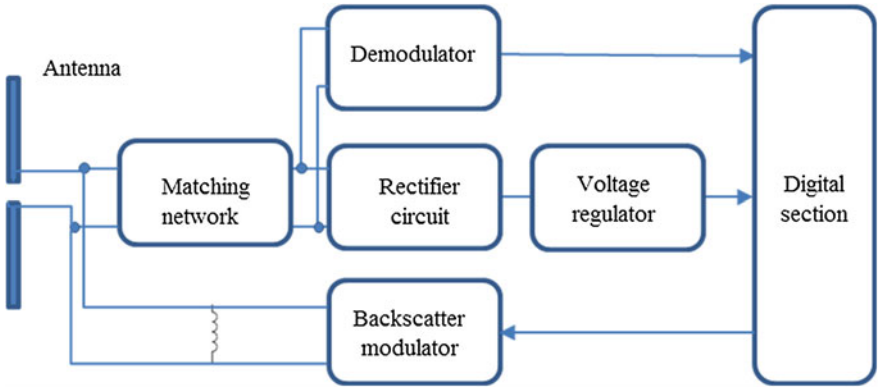


Fig. 15 Passive RFID tag architecture [35]

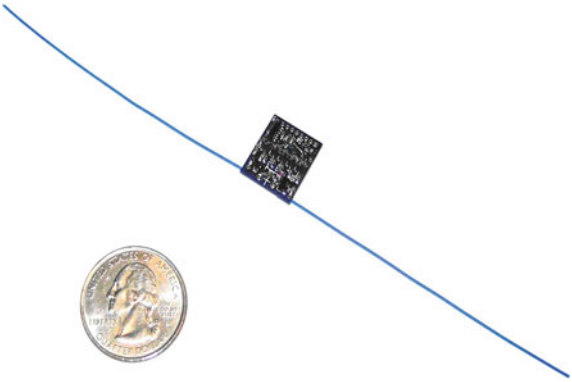
voltage (RF signal) into a DC voltage, which is used by a series voltage regulator to provide the regulated voltage required for the correct operation of the digital section. The rectifier is matched with the antenna in order to ensure the maximum power transfer from the tag antenna to the input of the rectifier. A backscatter modulator is used to modulate the impedance seen by the tag antenna, when transmitting [3]. The RF section is then connected to the digital section, which typically is a very simple microprocessor or a finite-state machine able to manage the communication protocol. In a general description, the conjunction of RF section (after the antenna) and digital section is known as RFID chip. To generalize, all this functions are pointed to a RFID tag but it should be understood that these RF and digital tasks are performed in fact by the RFID chip.

Smart RFID sensors

As RFID technology advances from simple passive tags to smart tags that are enhanced with sensors and computational ability, energy harvesting and power management become increasingly important. These tags may perform multiple functions, such as generating user-specific coded responses to inquiry, forming ad-hoc networks with other devices of the same class, or performing measurements of environmental parameters (e.g. temperature, pressure, or humidity). In order to facilitate these new sensing regimes, additional computing power is needed. The ability to harvest sufficient energy for long periods of operation is crucial to these applications [36]. The electromagnetic energy harvesting for smart RFID sensors in a WSN is then a critical design feature on the path to the grand vision of ubiquitous computing.

The literature defines this smart tag embodiment as the Wireless Identification and Sensing Platform (WISP) [36]. The WISP depicted in Fig. 16, is an augmented tag, powered and read by the commercial EPC Gen 2 RFID readers. Unlike conventional tags, WISPs include a fully programmable, low-power microcontroller unit (MCU) and sensors. The cornerstone of low-power operation rests on enhanced power harvesting and capacitive storage as well as envisaged future enhancements

Fig. 16 WISP platform [37]



via integrated circuit and optimization of the protocol stack. In the case of the WISP, the EPC Gen 2 protocol is implemented in software on the microcontroller instead of a dedicated hardware finite state machine; this enhanced programmability is expected to be a critical component enabling future RFID system optimization [37].

The analog architecture of WISP's and RFID sensors in general slightly differs in purpose from that conventional RFID tags. Due to the relatively high power consumption of WISP, the rectifier is designed to supply more current than ordinary tags. This is why the voltage rectification is the key point on the demanding need for next generation of RFID sensors in the IoT.

3.3 Voltage Rectification

Given the recent advances in energy efficiency for the circuit components of a sensor (i.e., diodes that require less forward voltage threshold), and the low-power operation modes supported by the device itself (i.e., sleep mode consuming only millivolt), there is a visible need for revisiting energy harvesting circuit design that can successfully operate a RFID sensor node. Several works are focused on increase this efficiency conversion. Then there is still a lot of open issues when designing WPT systems, and a lot of challenges still to be achieved, especially on the receiving side [25, 33, 34].

As mentioned before, the alternating voltage induced in the tag antenna is passed through an impedance matching network and fed into the power harvester for rectification (rectifier circuit on Fig. 15). There are two important considerations to take into account in the design of voltage rectification circuits. Depending on the consumption of DC power of the digital device (a simple RFID tag or a smart tag but both on the passive classification) two models can be established:

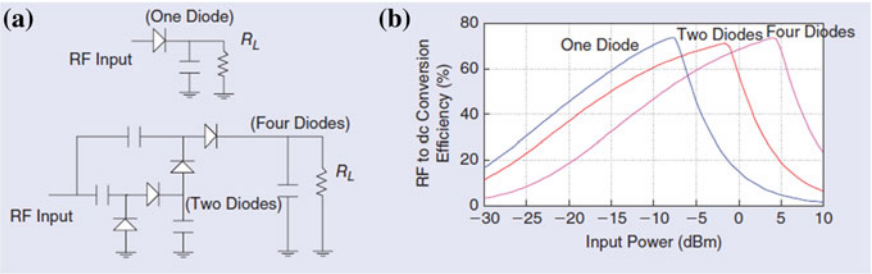


Fig. 17 RF-to-DC conversion efficiency dependency on the rectifier circuit topology. **a** Rectifier topologies. **b** RF-to-DC conversion efficiency versus RF input power [24]

- the low power model, usually used in traditional passive RFID tag without special sensing functions;
- and the duty cycle model where the RFID tag introduces some smart tasks (a microcontroller), i.e., a smart tag sensor.

Additionally and for both models and in general for all the passive RFID technology, there should be a special consideration on the non-linearity of the rectifier circuit in order to ensure the maximum transmission power from the antenna to the rectifier in the rectenna.

A low power consumption model

Figure 17 shows that for low-input power levels, topologies using a single diode lead to a better RF-to-DC conversion efficiency than topologies with more diodes. Traditionally, the optimization goal for this type of circuits with low power consumption, is to ensure that the minimum threshold voltage required by the digital logic is maintained at all times.

A schematic of different topologies of N-stage voltage multiplier conforming the rectifier are shown in Fig. 17. The power harvester is a half-wave rectifier in which current is passed to the next stage only during the positive phase of the RF signal. Traditionally, an N-stage voltage multiplier circuit (such as a Dickson multiplier) is used as rectifier, with N chosen to satisfy the minimum required voltage. Historically, Schottky diodes have been used in the multiplier circuit to utilize their very low turn on voltages, low series resistance, and low junction capacitance. The design parameters involve the number of stages of the voltage multiplier, the size of the diodes, and the coupling capacitors. The main constituent of the tag input impedance arises from the rectifier circuit (a parallel of a resistance and a capacitance [35]).

The RF-to-DC efficiency conversion depends on the topology selected for the rectifier circuit. Depending on the amount of input power at the rectifier, topologies with less number of rectifying devices may lead to better RF-to-DC conversion efficiency. This is explained because of the need of a minimum amount of input power in order to switch the rectifying devices [24].

A duty cycle model

However, in the case of RFID sensor applications when a microcontroller is used, it is not required a continuous supply of power. A duty cycling approach selected for this

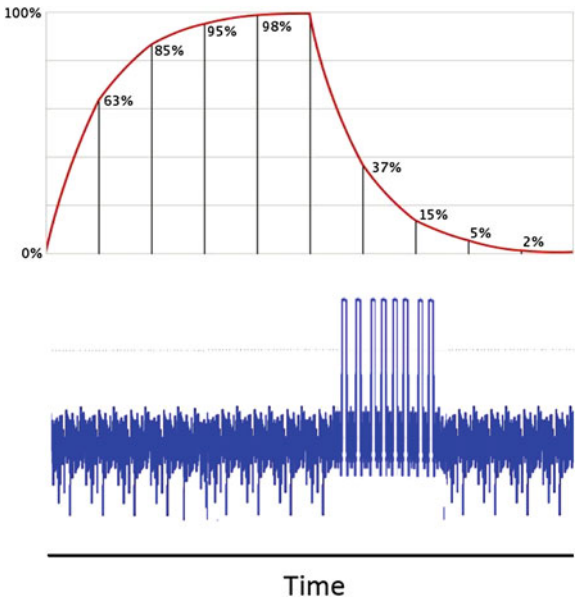


Fig. 18 WISP charge and discharge cycle of the at the *top*, microcontroller activity at the *bottom*. The duty cycle is the ratio of the time when the microcontroller is "on" to the total time of each period of charge and discharge of the WISP [37]

application allows a charge accumulation over several transmission cycles (Fig. 18). The duty cycling approach consists in reproduce a periodic sensing operation, where a duty cycle ratio is defined as the ratio of the time when the RFID sensor is "on" to the total time of each period. Once a sufficient amount of energy is harvested and the voltage threshold is met, the microcontroller can be switched "on". In order to maintain operation of the RFID sensor enough time to let the microcontroller to accomplish its tasks, it would have to be allocated for the harvester to recharge. It means the frequency and duration of tag activity will depend on the available harvested power.

This operation mode leads to new design constraints and different optimization criteria for the rectifier stage. The optimization goals can be shifted toward producing higher voltage by sacrificing the output current since the charge can be stored over multiple read cycles. These consumption requirements are modelled by calculating the RF-to-DC conversion efficiency for different load resistances given a topology in function of the powering needs [37].

Considerations on the non-linearities in passive RFID sensors

Other important parameter that improves RF-to-DC conversion efficiency is the non-linear treatment of the rectifier circuit. In order to achieve an optimum input matching, the antenna impedance should be equal to the complex conjugate of the input impedance of the RF circuit (i.e. rectifier) to ensure the maximal power transfer. This type of matching is easily done in linear circuits where the input impedance of the circuit does not change with the input signal amplitude. However since the rectifier diodes have a non-linear behaviour, the input impedance of rectifier circuit

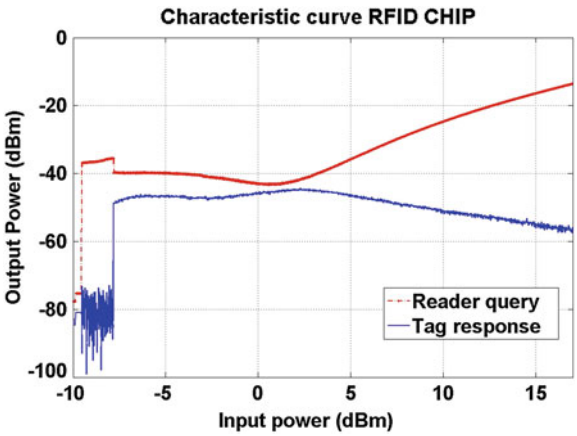


Fig. 19 Non-linear behavior of the UHF passive RFID tag. The tag response level changes in function of the input power due the impedance variations of the rectifier circuit

significantly changes with the input signal level and it is hard to get a conjugate impedance match for all values of input power. Then the rectifier circuits are typically designed for a specific value of input power level. For this reason, a behavioural model that accounts for these changes should be used on the design process. Theoretical models as the poly-harmonic distortion (PHD) [38] and simulation tools as harmonic balance (HB) [39] approximate the reactive behaviour of the circuit in order to get the proper antenna-rectifier matching.

The rectennas have similar rectifier architecture as passive RFID tags. The difference is that the harvesting rectennas achieve efficiencies of RF to DC conversion up to 77 % thanks to the treatment of the non-linearity effects with appropriate techniques of simulation as the HB, which considers the effect of the harmonics currents on the rectifier design encouraging also the effective matching with the antenna over a frequency range considering more than one impedance point minimizing the harmonic effect. In passive RFID tags contrary, the antenna design begins directly from the knowledge of one impedance value, which is the impedance of the RFID chip at the fundamental frequency described on manufacturer data sheets. Then, the design process tries to ensure the matching at the frequency of work, but there is no treatment of the harmonic currents received from the reader, which impairs the purity of the rectified DC signal, nor to those reflected in the antenna input as a result of the non-linearity of the diodes in the return link. This absence of non-linear treatment, besides that decreases the rectifier efficiency of the tag, allows the tag antenna to radiate the reflected harmonic currents generated by the rectifier that triggers in backscattered harmonics [40].

Figure 19 shows the non-linear behaviour of a UHF passive RFID tag. After the tag is activated around −8 dBm of input power (sent by the reader), its backscattered power level (output power) changes in function of the power sent by the reader. The tag response even decreases (until −60 dBm) when the input power is maximal (17 dBm). This shows the variations on the matching between antenna and chip due

to the non-linear behaviour of the rectifier. The figure demonstrates how the tag has an optimal behaviour only at certain level of input power (around 2 dBm).

3.4 Conclusion: The Ubiquity of Energy

Passive sensor tags are a new area, and it is expected to appear a number of new solutions for distributed networks and sensing systems in the near future, basically powered by harvesting and not with dedicated sources of energy. The major challenges here are the need for extremely low power consumption and the limited accuracy of very low power sensors. However it is expected that new fabrication technologies, novel device structures and new processing and communication techniques will increase the power output and efficiency, while reducing the size and weight of many energy harvesters, all resumed in reduce the power consumption.

Focusing on the electromagnetic energy harvesting, its future is to add a true meaning of mobility of electrical devices by eliminating the need of centralized power sources and depending on the RF Energy existing in the air to charge these electronic devices. In this case we are not talking only about sensors or smart RFID tags, but any electrical device. This scheme imposes not only a ubiquitous computing but also the ubiquity of energy.

4 Conclusion

This chapter showed the important place that the RFID technology with its identification, communication and sensing functionalities could occupy in the IoT context. Moreover, a focus on the energy harvesting methods and in particular, the electromagnetic energy harvesting is presented while highlighting its techniques and performance.

The evolution of the RFID integrating sensing capacities, low power sensors and efficient energy harvesters will actively participate to the fast development of the IoT. It is hoped that these developments will always adopt ecological considerations and also play an important role in the development of applications that help in delivering a greener world [41]. With this vision, emerging ways of thinking must produce new ideas. For example in a tag network, each tag could benefit from the power transmitted by all the others. Preliminary works are presented in Duroc and Andia Vera [42] with an analytical development that defines some guidelines in order to choose the optimal network configuration in terms of energy efficiency.

References

1. CASAGRAS, CASAGRAS EU project final report, http://www.grifs-project.eu/data/File/CASAGRAS%20FinalReport%20%282%29.pdf
2. National Intelligence Council, Disruptive technologies global trends 2025. Six technologies with potential impacts on US interests out to 2025 (2008)
3. Finkenzeller, K.: RFID Handbook: Fundamentals and Applications in Conctless Smart Cards, Radio-Frequency Identification and Near-Field Communication, 3rd edn. Wiley, New York (2010)
4. Weinstein, R.: RFID: a technical overview and its applications to the enterprise. In: IEEE Computer Society, pp. 27–33 (2005)
5. Preradovic S., Karmakar N.C.: Chipless RFID: bar code of the future. In: IEEE Microwave Magazine, pp. 87–97 (2010)
6. Ondemir, O., Ilgin, M.A., Gupta, S.M.: Optimal end-of-life management in closed-loop supply chains using RFID and sensors. IEEE Trans. Ind. Inform. **8**(3), 719–728 (2012)
7. Roy, S., Jandhyala, V., Smith, J.R., Wetherall, D.J., Otis, B.P., Chakraborty, R., Buettner, M., Yeager, D.J., Ko, Y.C., Sample, A.P.: RFID: From supply chains to sensors nets. Proc. IEEE **98**(9), 1583–1592 (2010)
8. Liu, H., Bolic, M., Nayak, A., Stojmenovic, I.: Taxonomy and challenges of the integration of RFID and wireless sensor networks. In: IEEE Network, pp. 26–32 (2008)
9. Bhattacharya, R., Florkemeier, C., Sarma, S.: Towards tag antenna based sensing-An RFID displacement sensor. In: Proceedings of 2009 International Conference on RFID, pp. 95–102 (2009)
10. Marrocco, G., Occhiuzzi, C., Amato, F.: Sensor-oriented passive RFID. In: Book Chapter The Internet of Things, Part 4, pp. 273–282 (2010)
11. Delen, D., Sharda, R., Hardgrave, B.C.: The promise of RFID-based sensors in the perishables supply chain. In: IEEE Wireless Communications, pp. 82–88 (2011)
12. Bhattacharyya, R., Floerkemeier, C., Sarma, S., Deavours, D.: RFID tag antenna based temperature sensing in the frequency domain. In: Proceedings of 2011 IEEE International Conference on RFID, pp. 70–77 (2011)
13. Gao, J., Siden, J., Nilsson, H.E.: Printed electromagnetic coupler with an embedded moisture sensor for ordinary passive RFID tags. IEEE Electron Device Lett. **32**(12), 1767–1769 (2011)
14. Chang, K., Kim, Y.H., Kim, Y.J., Yoon, Y.J.: Functional antenna integrated with relative humidity sensor using synthesized polyimide for passive RFID sensing. Electron. Lett. **47**(5), 7–8 (2007)
15. Occhiuzzi, C., Rida, A., Marrocco, G., Tentzeris, M.: RFID passive gas sensor integrating carbon nanotubes. IEEE Trans. Microwave Theory Tech. **59**(10), 2674–2684 (2011)
16. Todd, B., Phillips, M., Schultz, S.M., Hawkins, A.R., Jensen, B.D.: Low-cost RFID threshold shock sensors. IEEE Sens. J. **9**(4), 464–469 (2009)
17. Occhiuzzi, C., Cippitelli, S., Marrocco, G.: Modeling, design and experimentation of wearable RFID sensor tag. IEEE Trans. Antennas Propag. **58**(8), 2490–2498 (2010)
18. Lakafosis, V., Rida, A., Vyas, R., Yang, L., Nikolaou, S., Tentzeris, M.M.: Progress towards the first wireless sensor networks consisting of inkjet-printed, paper-based RFID-enabled sensor tags. Proc. IEEE **98**(9), 1601–1609 (2010)
19. Viikarai, V., Seppa, H.: RFID MEMs sensor concept based on intermodulation distortion. IEEE Sens. J. **9**(12), 1918–1923 (2009)
20. Capdevila, S., Jofre, L., Bolomey, J.C., Romeu, J.: RFID multiprobe impedance-based sensors. IEEE Trans. Instrum. Measur. **59**(12), 3093–3101 (2010)
21. Catarinucci, L., Colella, R., Tarricone, L.: A cost-effective UHF RFID tag for transmission of generic sensor data in wireless sensor networks. IEEE Trans. Microwave Theory Tech. **57**(5), 1291–1296 (2009)
22. Preradovic, S., Menicanin, A.: Chipless wireless sensor node. In: Proceedings of International Convention MIPRO, pp. 179–182 (2012)

23. Herraiz-Martinez, F.J., Paredes, F., Zamora Gonzalez, G., Martin, F., Bonache, J.: Printed magnetoinductive-wave (MIW) delay lines for chipless RFID applications. IEEE Trans. Antennas Propag. **60**(11), 5075–5082 (2012)
24. Boaventura A., Collado A., Borges Carvalho N., Georgiadis A.: Optimum behavior. IEEE Microwave Mag. **14**(2), 26–35 (2013)
25. Bui, N., Georgiadis, A., Miozzo, M., Rossi, M., Vilajosana, X.: SWAP Project: Beyond the state of the art in harvested energy-power wireless sensors platform design. In: Proceedings of 2011 IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, pp. 837–842 (2011)
26. Green, M.: Third Generation Photovoltaics, Advanced Solar Energy Conversion. Springer, Heidelberg (2006)
27. Bebby, S., White, N.: Energy Harvesting for Autonomus Systems. Artech House, London (2010)
28. Le-Wei Li, J.: Wireless power transmission: state-of-the-arts in technologies and potential applications (Invited Paper). In: Proceedings of the Asia-Pacific Microwave Conference 2011, pp. 86–89 (2011)
29. Brown, W.C., George, R.H., Heeman, N.I.: Microwave to dc converter, US Patent 3 434 678, March 1969
30. Brown, W.C.: The history of power transmission by radio waves. IEEE Trans. Microwave Theory Tech. **32**(9), 1230–1242 (1984)
31. Shinora, N.: Wireless Power Transmission for Solar Power Satellite (SPS), Internal Report. Kyoto University, Japan (2006)
32. Liu, L., Zhang, R., Chua, K.: Wireless information transfer with opportunistic energy harvesting. IEEE Trans. Microwave Theory Tech. **12**(1), 288–300 (2013)
33. Nintanavongsa, P., Muncuk, U., Richard Lewis D., Roy Chowdhury K.: Design optimization and implementation for rf energy harvesting circuits. IEEE J Emerg. Sel. Top. Circ. Syst. **2**(1), 24–33 (2012)
34. Iannello, F., Simeone, O., Spagnolini, U.: Energy Management Policies for Passive RFID Sensors with RF-Energy Harvesting. IEEE International Conference on Communications (ICC) **2010**, 1–6 (2010)
35. De Vita, G., Iannaccone, G.: Desgin criteria for the RF section of UHF and microwave RFID transponders. IEEE Trans. Microwave Theory Tech. **53**(9), 2978–2990 (2009)
36. Sample, A.P., Yeager, D.J., Smith, J.R., Powledge, P.S., Mamishev, A.V.: Energy harvesting in rfid systems. In: Proceedings of International Conference on Actual Problems of Electron Devices Engineering, pp. 445–449 (2006)
37. Sample, A.P., Yeager, D.J., Powledge, P.S., Mamishev, A.V., Smith, J.R.: Design of an RFID-Based Battery-Free programmable sensing platform. IEEE Trans. Instrum. Measur. **57**(11), 2608–2615 (2008)
38. Boaventura, A.S., Testera, A.R., Carvalho, N.B., Barciela, M.F.: Using X-parameters to model diode-based RF power probes. In: Proceedings of IEEE International Microwave Symposium, pp. 1–4 (2011)
39. Georgiadis, A., Andia Vera G., Collado A.: Rectenna design and optimization using reciprocity theory and harmonic balance analysis for electromagnetic energy harvesting. IEEE Antennas Wirel. Porpag. Lett. **9**, 444–446 (2010)
40. Andia Vera G., Duroc Y., Tedjini S.: Analysis of harmonics in UHF RFID signals. IEEE Trans. Microwave Theory Tech. **61**(6), 2481–2490 (2013)
41. Duroc, Y., Kaddour, D.: RFID potential impacts and future evolution for green projects. Energy Procedia J. Elsevier/Science Direct **18**, 91–98 (2012)
42. Duroc, Y., Andia Vera, G.: Considerations on the backscattered wireless communication links. Microwave Opt. Technol. Lett. **55**(3), 554–559 (2013)

Biography of the Editor

Dr. Subhas Chandra Mukhopadhyay (M'97, SM'02, F'11) graduated from the Department of Electrical Engineering, Jadavpur University, Calcutta, India with a **Gold medal** and received the Master of Electrical Engineering degree from Indian Institute of Science, Bangalore, India. He has PhD (Eng.) degree from Jadavpur University, India and Doctor of Engineering degree from Kanazawa University, Japan. Currently he is working as a Professor of Sensing Technology with the School of Engineering and Advanced Technology, Massey University, Palmerston North, New Zealand. He has over 24 years of teaching and research experiences.

His fields of interest include Sensors and Sensing Technology, Electromagnetics, control, electrical machines and numerical field calculation etc.

He has authored/co-authored two books and over **300** papers in different international journals, conferences and book chapter. He has edited **10** conference proceedings. He has also edited **10** special issues of international journals as lead guest editor and **17** books out of which **15** are with Springer-Verlag.

He was awarded numerous awards throughout his career and attracted over NZ $3.5 M on different research projects.

He is a **Fellow** of IEEE (USA), a **Fellow** of IET (UK). He is a Topical editor of IEEE Sensors journal, an Associate Editor of IEEE Transactions on Instrumentation and Measurements and a technical editor of IEEE Transactions on Mechatronics. He is in the editorial board of, Sensors and Transducers, Transactions on Systems, Signals and Devices (TSSD). He is the co-Editor-in-chief of the International Journal on Smart Sensing and Intelligent Systems (http://www.s2is.org). He was the Technical Programme Chair of ICARA 2004, ICARA 2006 and ICARA 2009. He was the General chair/co-chair of ICST 2005, ICST 2007, IEEE ROSE 2007, IEEE EPSA 2008, ICST 2008, IEEE Sensors 2008, IEEE Sensors 2009, ICST 2010, IEEE Sensors 2010, ICST 2011, ICST 2012 and ICST 2013. He is the Founding and Ex-**Chair** of the IEEE Instrumentation and Measurement Society New Zealand Chapter.

He currently chairs the Technical Committee TC 18 on Environmental Monitoring of IEEE Instrumentation and Measurements Society.

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7,

Index

S. C. Mukhopadhyay (ed.), *Internet of Things*, Smart Sensors, Measurement and Instrumentation 9, DOI: 10.1007/978-3-319-04223-7,