Design and Deployment of an IoT Application–Oriented Testbed

Laura Belli, Simone Cirani, Luca Davoli, Andrea Gorrieri, Mirko Mancin, Marco Picone, and Gianluigi Ferrari, University of Parma

The global reach and extreme heterogeneity of the Internet of Things present major application development challenges. Using the same Web–based approach underlying the Internet's evolution into the IoT, the Web of Things Testbed provides a stable, open, dynamic, and secure infrastructure to simplify application design and testing.

Due to recent development and innovation in both hardware and software, the global network of networks, or Internet of Things (IoT), is finally becoming a reality. The IoT's diverse billions of communicating devices, or smart objects (SOs), enable a new paradigm of interactivity among all manner of things and people. One of the IoT's biggest hurdles is the monolithic nature and fragmentation of existing vertical closed systems, architectures, and application areas. To overcome this, researchers are defining and standardizing interoperability in communication protocols and device mechanisms to allow for more efficient interaction among all IoT components,

and Internet Protocol version 6 (IPv6) is emerging as the base network protocol for all IoT applications.[1]

To foster IoT development and diffusion, applications are increasingly built around the well-known Web model, bringing about the so-called Web of Things (WoT). The Web-based approach helped to greatly expand the Internet, and will likely have the same effect on the IoT.[2] WoT applications rely on specific Web-oriented application-layer protocols similar to HTTP, such as the Constrained Application Protocol (CoAP),[3] and, more generally, protocols complying with the Representational State Transfer (REST) architectural style.

Whereas simulation tools typically focus on evaluating lower-layer communication protocols, in recent years several IoT testbeds have been deployed to evaluate IoT solutions in realistic smart environments under real-world conditions. The IoT-Lab[4] is an example of this kind of testbed environment: it provides a very large-scale infrastructure with more than 2,700 wireless sensor nodes spread across six different sites in France, and is used to test protocols at the link and network layers and to collect performance results such as energy consumption or packet delivery ratio. Although these lower-layer protocols have been widely investigated, additional efforts are needed to create new innovative services, promote long-term evolution of systems, and ensure the robustness of applications against changes that might occur over time—thereby furthering WoT development. To easily build applications for this environment, developers need the ability to work at a high level of abstraction and without worrying about low-level details.

Another relevant large-scale IoT testbed is SmartSantander, consisting of approximately 20,000 nodes deployed in different cities across Europe.[5] With its orientation toward smart-city services and applications, however, this testbed focuses on environmental data collection and is thus not set up to allow experimentation on a fully addressable and resource-oriented WoT. In particular, SmartSantander does not consider direct and bi-directional interactions between humans and objects.

WEB OF THINGS TESTBED
The Web of Things Testbed (WoTT) is a heterogeneous and innovative WoT-based testbed that enables developers to easily design and evaluate new services and applications in a real IoT environment and to effectively test human–object interaction mechanisms, which will play a fundamental role in broadening IoT use. WoTT is particularly suited for this purpose because its architecture is completely based on standard protocols and network interfaces, without custom or proprietary solutions that would jeopardize interoperability among nodes.

WoTT's main goals are to

› hide low-level implementation details,
› enhance network self-configuration by minimizing human intervention,
› transparently and simultaneously manage multiple protocols and platforms, and
› provide a platform for the design and testing of human–object interaction patterns.

To effectively test new WoT-related applications, WoTT consists of several types of nodes that differ in terms of both computational capabilities and radio-access interfaces. Nonetheless, these nodes can be grouped into two main classes: *constrained IoT* (CIoT) nodes and *single-board computer* (SBC) nodes. CIoT nodes are mainly based on the open source Contiki OS and correspond to Class 1 devices—those that cannot easily talk to other Internet nodes that have a full protocol stack, but that can use a protocol stack designed for constrained nodes—as defined in the Internet Engineering Task Force (IETF) memo "Terminology for Constrained Node Networks."[6] SBC nodes are more powerful, typically running a Linux OS and having multiple network interfaces; these correspond to Class 2 devices—those that use the same protocols as notebooks or servers.[6] Regardless of what the actual nodes are, the standard communication protocols and mechanisms used in WoTT enable the testbed to manage node diversity seamlessly, making it possible to treat each node simply as an IP-addressable host. Table 1 shows the CIoT and SBC nodes in detail.

AN IP-BASED INFRASTRUCTURE FOR SMART OBJECTS
CIoT nodes are connected at the physical layer by IEEE 802.15.4 wireless links, whereas IPv6 is used at the network layer in combination with 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network)[7] and RPL (the routing protocol for low-power and lossy networks, or LLNs). Sensor-equipped CIoT nodes can act as CoAP servers or clients running Erbium, a lightweight CoAP implementation.

As with CIoT nodes, SBC nodes can act as CoAP clients or servers, with fewer implementation constraints. For example, on Arduino Yún nodes, a JavaScript application initializes a CoAP server through an instance of Node.js. Other SBC nodes, such as the Intel Galileo boards and Raspberry Pi, can support different types of languages ranging from Python to Java.

As Figure 1 shows, WoTT is heterogeneous by design to enable seamless communication among SOs and between the WoTT and external Internet elements such as the cloud, ISPs, and consumers. IP protocol adoption—in particular, IPv6 or IPv6+6LoWPAN—is universally considered a key communication enabler for the future IoT. Thus, WoTT adopts IP as a common network substrate, thereby allowing

TABLE 1. Constrained Internet of Things (CIoT) and single-board computer (SBC) nodes in the Web of Things Testbed.

CIoT nodes

No.	Node	Hardware	OS	Network interface
6	TelosB	MCU: TI MSP430F1611	Contiki	IEEE 802.15.4
		RAM: 10 Kbytes		
		ROM: 48 Kbytes		
20	Zolertia Z1	MCU: TI MSP430F2617	Contiki	IEEE 802.15.4
		RAM: 8 Kbytes		
		ROM: 92 Kbytes		
10	OpenMote	MCU: ARM Cortex-M3	Contiki	IEEE 802.15.4
		RAM: 32 Kbytes		
		ROM: 512 Kbytes		

SBC nodes

| No.	Node	Hardware	OS	Network interface(s)
20	Intel Galileo	CPU: SoC X Intel Quark X1000	[Linux] Debian	IEEE 802.3
		RAM: 256 Mbytes		
		Memory (SD): 8 Gbytes		
5	Raspberry Pi B	CPU: Broadcom BCM2835 ARM11	[Linux] Raspbian	IEEE 802.3/802.11
		RAM: 512 Mbytes		
		Memory (SD): 8 Gbytes		
5	Arduino Yún	Linux environment	[Linux] OpenWRT	IEEE 802.3/802.11
		CPU: Atheros AR9331		
		RAM: 64 Mbytes		
		ROM: 16 Mbytes		
		Arduino environment	Arduino	
		MCU: ATmega32u4		
		RAM: 2.5 Kbytes		
		ROM: 32 Kbytes		
4	UDOO	Linux environment	[Linux] UDOObuntu	IEEE 802.3/802.11
		CPU: Freescale i.MX 6 ARM Cortex-A9		
		RAM: 1 Gbyte		
		Memory (SD): 8 Gbytes		
		Arduino-like environment	Arduino	
		MCU: Atmel SAM3X8E ARM Cortex-M3		
		RAM: 100 Kbytes		
		ROM: 512 Kbytes		

MCU: memory control unit
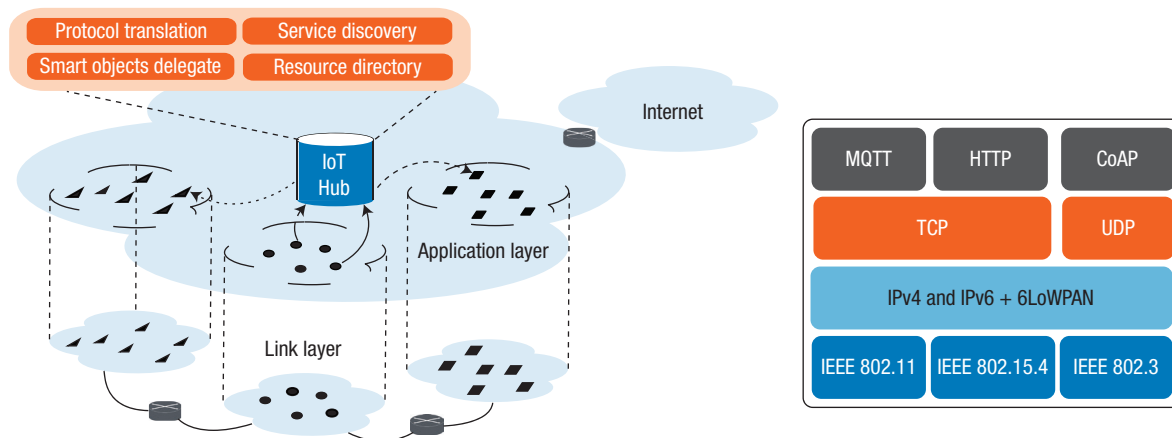
FIGURE 1. Web of Things Testbed (WoTT) architecture and protocol stack. The central component is the Internet of Things (IoT) Hub, which interacts with the various layers and manages the testbed's heterogeneous network. 6LoWPAN: IPv6 over Low-Power Wireless Personal Area Network; CoAP: Constrained Application Protocol; MQTT: Message Queuing Telemetry Transport protocol; UDP: User Datagram Protocol.

simple integration into the existing Internet. Note that all WoTT nodes use standard protocols at all layers of the protocol stack; this includes several physical (PHY) and media access control (MAC) standards—for example, IEEE 802.11, IEEE 802.15.4, and IEEE 802.3—as well as application-layer protocols. The architecture also contains an innovative network element, the IoT Hub,[8,9] which operates at different layers of the protocol stack to further enhance interoperability among communicating devices by integrating several networks into a single IP-based substrate and implementing important functions at the application layer. Due to greater capabilities in computational power and networking, SBC nodes can effectively implement all IoT Hub functions.

WoTT's Wi-Fi networking infrastructure is based on Cisco Connected Mobile Experiences (CMX) access points and can be used to track devices for indoor localization purposes. In particular, the CMX platform provides Mobility Services Engine RESTful APIs, which allow developers to integrate service customization with location information into mobile applications, such as location-aware equipment tracking, guest access, and device-based services. This feature is currently used to build user location–aware IoT applications that can continuously monitor users, enabling specific and augmented interaction with the surrounding environment.

Focusing on the application layer, WoTT currently supports CoAP, the Message Queuing Telemetry Transfer protocol (MQTT), and HTTP. CoAP is a binary and lightweight Web transfer protocol built on top of the User Datagram Protocol (UDP) and follows a request/response paradigm.[3] It has been explicitly designed to work with devices operating in LLNs. MQTT is also a lightweight publish/subscribe protocol that is suitable for constrained SOs running on top of TCP, such as sensors or actuators. Finally, HTTP is mainly used for communication between WoTT and external Internet actors or consumers, such as cloud storage services or IoT-unaware clients. Among WoTT components, the IoT Hub is the key IoT enabler because it manages the different access technologies and supports full IP connectivity among all objects. The implementation of several functionalities at the application layer enables all the protocols listed in Figure 1 to coexist in the same environment.

SO software has been developed with different programming languages,

reinforcing the idea that—because of various features provided at the application layer, together with strict compliance to IoT standards—developers can create new IoT applications easily and without additional constraints.

IoT HUB–ENABLED SMART OBJECT INTERACTIONS

WoTT does not simply enable communications between IoT actors; it constitutes a "uniform" super-entity able to provide enhanced functionalities that go beyond the mere union of its components' features.

To achieve this super-entity status, WoTT uses the various communication technologies in the IoT Hub to bridge and merge together several networks into a single IP network. The IoT Hub also implements several functions at the application layer: it manages the services and resources available in the overall infrastructure, thereby playing a key role at the application layer.

IoT Hub use is expedient for several reasons. The extreme heterogeneity of IoT devices requires mechanisms to support the management of and seamless interactions among SOs as well as humans. Moreover, because SOs' limited data-collection capabilities could preclude them from handling large numbers of concurrent requests,

limiting direct access to SOs is preferable. In other cases, extremely limited devices could act as clients to implement data-collection behavior.

Standardization efforts for IoT Hub design are in progress. A relevant example standard is HyperCat,[10] which introduces standard specifications to allow servers to expose JavaScript Object Notation (JSON)-based hypermedia catalogues as collections of URLs. Thus, IoT clients can discover data available on servers using RESTful methods on HTTPS and JSON formats. Unlike the IoT Hub, however, HyperCat works at a higher level of abstraction because it is intended to allow IoT concept–based reasoning and service composition, and it does not take into account direct interactions with constrained devices in which specific protocols (such as CoAP) should be adopted to minimize energy and memory consumption.

From a networking standpoint, the IoT Hub is a fog node[11] placed at the edge of multiple physical networks with the goal of creating an IP-based IoT network. The IoT Hub plays a fundamental role by implementing the following functions at the link and application layers of the protocol stack:

❭ *Border Router*—the IoT Hub bridges one or more networks (such as several IEEE 802.15.4 networks);

❭ *Service and Resource Discovery*— the IoT Hub is able to discover which SOs are available in the network and their hosted resources;

❭ *Resource Directory (RD)*—the IoT Hub complies with the specifications provided in the IETF technical report "CoRE Resource Directory"[12] and maintains a list

of all resources available in the bridged networks, thereby creating a centralized entry point for applications that need to perform resource lookup;

❭ *Origin Server (OS)*—the IoT Hub provides a CoAP server that hosts the SOs' resources;

❭ *CoAP-to-CoAP (C2C) proxy*— the IoT Hub provides proxying capabilities for CoAP requests coming from external clients targeting constrained nodes;

❭ *HTTP-to-CoAP (H2C) proxy*—the IoT Hub provides HTTP-to-CoAP cross-proxying (protocol translation) to allow HTTP clients to access CoAP resources; and

❭ *Cache*—the IoT Hub keeps a cache with the representation of most recently accessed resources to act as an SO delegate, thus minimizing latencies and unloading constrained devices.

Because the IoT Hub relies on standard interaction mechanisms and communication protocols, SOs do not depend on it for their operation. Instead, the IoT Hub mitigates the presence of nonstandard components that are interoperable with standard-compliant devices. The IoT Hub is not required for interaction and interoperability, but its presence extends the IoT network and increases its capabilities by simplifying and hiding complex and important tasks such as service discovery and routing.

INTEGRATION CHALLENGES

The main challenges encountered in WoTT deployment have been design related: how to define different elements and their functionalities, represent different resources and their relationships through suitable

hypermedia, and maintain compatibility with standards.

The efforts devoted to WoTT design, the IoT Hub, and the use of standards have simplified the deployment process, making the integration of all different elements straightforward, notwithstanding their heterogeneity. In particular, WoTT hides critical implementation issues encountered at lower layers. For example, in a constrained network, such as IEEE 802.15.4, the resource advertisement feature is a critical point, requiring strict assumptions about energy and memory consumption on each constrained device. It is possible to tackle this issue by adopting an efficient multicast-based solution that can reduce the energy consumption.[13] WoTT's application-oriented end users are not concerned with these implementation details.

Another implementation challenge hidden from end users is configuring network elements such as routers and access points to create a unique IP-addressable network. Original firmware provided with common network elements typically does not allow such unusual configurations. To overcome this limitation, WoTT network components have been flashed with other more customizable and advanced firmware, like Tomato.

SECURITY, AUTHENTICATION, AND AUTHORIZATION

With our goal of having the WoTT eventually become a publicly available platform, we designed the testbed to comply with security policies and mechanisms and to adopt strong defensive measures to counteract security threats coming from external environments as well as internal malicious IoT entities. Web-derived security technology
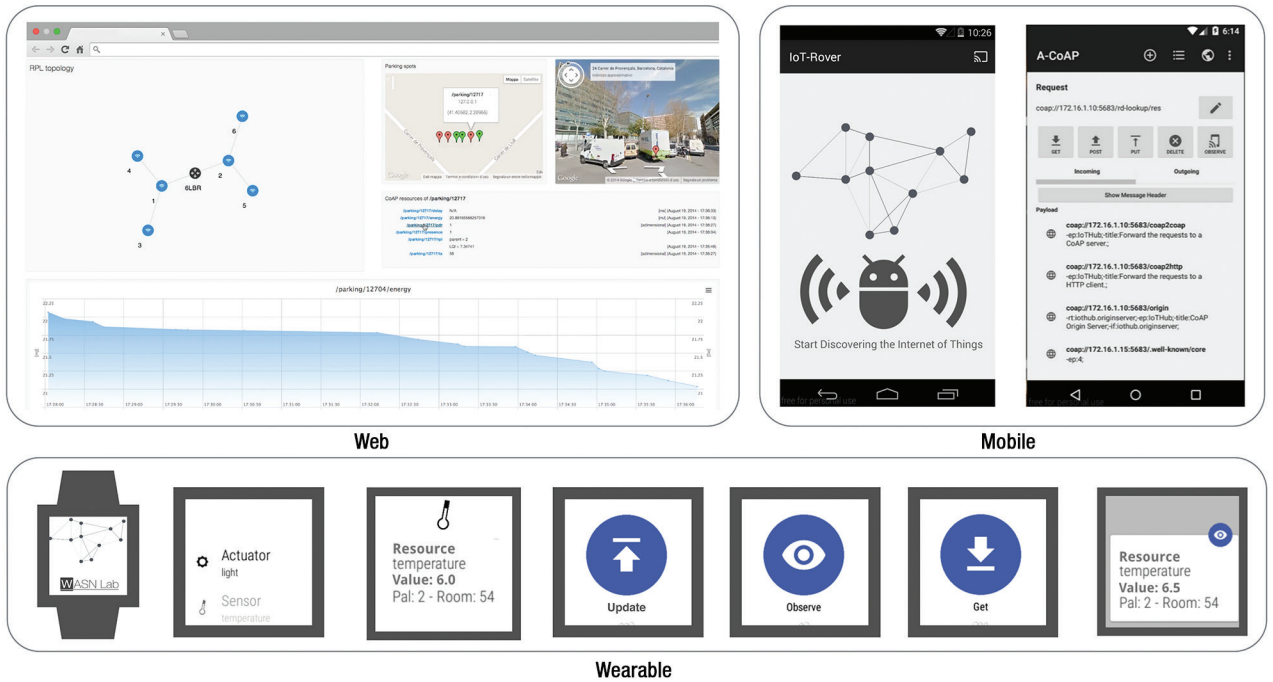
FIGURE 2. A WoTT-based application for mobile and wearable devices. Resources and interactions are revealed gradually, according to the Representational State Transfer (REST) paradigm, so that the application can adapt itself dynamically.

and mechanisms are useful—they can be taken as a reference and applied to enhance the protection and reliability of IoT environments.

The WoTT can manage and authenticate external request issuers, while simultaneously defining processing policies and rules. To comply with these specifications, the testbed allows external connections through virtual private network (VPN) and Secure Socket Shell (SSH) tunnels using credentials issued by WoTT administrators.

Several security mechanisms for SO interaction are implemented within the testbed. Transport Layer Security (TLS; in conjunction with HTTP) and Datagram TLS (DTLS; in conjunction with CoAP) are implemented on a set of devices hosting resources that need to be accessed through secured application protocols. The IoT Hub implements both protocols to provide secure resource access through proxying. Of course, end-to-end security through proxying cannot be ensured as no TLS-to-DTLS mapping is defined.

Although DTLS and TLS are implemented to enforce confidentiality

and authenticity of communication between endpoints, real-world IoT systems must be able to manage several users accessing resources deployed in a smart environment. This is when authorization comes into play: mechanisms must be defined to ensure that only authorized entities can interact with objects. For example, building offices in an IoT environment should be secure and only able to be unlocked by individuals who have been granted access. Although authorization is obviously a critical issue, research has yet to focus on defining mechanisms to manage IoT access policies and grant authorizations. WoTT implements IoT-OAS,[14] an authorization architecture based on the OAuth protocol, but this architecture implements a lightweight delegation approach to authorization.

BUILDING WoT APPLICATIONS

To validate WoTT's benefits and demonstrate the ease of integrating a newly deployed application within the testbed, we developed an application for wearable and mobile-oriented applications.

Thanks to their portability, wearable and mobile devices are obvious solutions for tracking people's activities. To accomplish this, we implemented indoor localization features into WoTT using CMX access points to triangulate the locations of people and objects. The availability of localization APIs in the CMX system enables us to create applications that can follow users throughout an IoT environment and possibly even anticipate their movements. Together with access-point-based localization, the use of on-board inertial measurement units can further improve the tracking experience.

Mobile devices play an important role in WoTT's architecture. Aside from interacting with SOs, they can also act as SOs, providing data generated by their onboard sensors. Mobile devices can thus be considered as WoTT nodes, making WoTT highly dynamic.

As a uniform, application-oriented platform, WoTT can be used by developers to easily create and test real-world IoT applications in a short period of time, thus making it more

attractive than other currently available platforms. This is due to ready-to-use capabilities and the direct/active interactions that a deployed application can have with the resources available in WoTT. From an operational point of view, developers simply run their applications on testbed resources without needing to add virtualized

performed according to the function set specified by the selected SO—for example, a light bulb might provide an on/off switch, or a temperature sensor might provide a way to read its value. Resources and interactions are revealed gradually, according to the REST paradigm, so that the application can adapt itself dynamically.

› *Acting* is used by clients to set up the value of a specified resource, such as activating an actuator, depending on the function set provided by the resource.

[**OPEN EVALUATION PLATFORMS ARE NECESSARY TO EXPLORE AND INTEGRATE INNOVATIVE IoT APPLICATIONS AND TO BRIDGE THE GAP BETWEEN USERS AND THINGS.**]

The observing approach, which has not been defined in HTTP, is a lightweight CoAP-oriented interaction mechanism.[16] SO resource observation is achieved by performing a particular CoAP GET request, which contains an "Observe" option. This option instructs the target SO to add a new subscriber that will receive subsequent resource updates in push mode. Subscribers can also stop observing a resource at any time and unsubscribe from updates.

environments or services; thus the application becomes part of a WoT scenario in which consumers are not only "readers" but active participants.

Based on the WoTT's capabilities, we developed the application shown in Figure 2. We tested this application on the Android Wear platform using LG G Watches and Android 5.0.1 smartphones.[15] In the near future, as more SOs are deployed, vendor-provided apps are less likely to be the usual means by which we interact with things, and a more standard approach will be required to do so effectively.

The application performs the following steps. First, the mobile device discovers nearby SOs proactively and reactively, by means of standard service discovery and resource directory mechanisms. Then, it forwards the collected information to its connected wearable device interface. Through wearable interfaces, a user can see and browse a list of all the resources that have been discovered and select one to interact with. Interactions are thereby

WoTT resources can be deployed on different platforms, such as different SOs, and by using heterogeneous protocols. However, this is completely transparent to a developer who is able to access all these resources; the only constraint is to use standard protocols. This is possible through the IoT Hub's abstraction ability—it is not provided by other existing testbeds, such as SmartSantander.

The SOs' own sensors and actuators, users' devices, and other SOs can each act as clients and interact with one another through the following approaches:

› *Polling* allows clients to retrieve the value associated with the queried resource by performing a CoAP GET request.
› *Observing* can be used by clients to receive asynchronous updates when the value of the specified resource changes, which is a more efficient mechanism because it avoids periodic data polling.

The use of standard communication protocols and network interfaces, well-known Web-based design approaches, and widely varied hardware platforms are changing the IoT and presenting new opportunities to developers, businesses, and users. In this dynamic and evolving scenario, the availability of real and accessible resource-oriented testbeds that allow active and direct interaction among users and devices with low-level nodes and services are a key enabler for widespread future IoT adoption—and, indeed, are driving the transition from the IoT to the WoT.

WoTT exemplifies a novel architectural and networking approach to important IoT challenges. Designing and implementing the testbed has confirmed the need for open evaluation platforms to explore and integrate innovative IoT applications and to bridge the gap between users and things. WoTT's hardware/software heterogeneity confirms that proper

use of standard protocols such as HTTP, CoAP, and MQTT and of interaction paradigms such as REST and service/resource discovery are fundamental to enabling transparent and dynamic interactions among multiple SOs and personal mobile and wearable devices. Moreover, this heterogeneity can be extended by improving the IoT Hub features—for example, by adding support to Bluetooth devices. This would open WoTT to an emerging category of IoT-enabled devices, namely those using Bluetooth low energy (BLE), and BLE service discovery mechanisms such as UriBeacon.

Nevertheless, important issues must be addressed to make the IoT part of daily life. Improved security, greater device and network interoperability, faster data processing, and easier development and deployment will be core focus areas for academic and industrial R&D during the next few years. Testbeds like WoTT are the perfect experimental playgrounds to boost and support IoT application development by creating a common space that designers, hardware manufacturers, and companies can exploit to make the IoT accessible and easy to use for everyone, just as the Internet is right now. ⬛

REFERENCES

1. J.P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*, Morgan Kaufmann, 2010.
2. R. Want, B.N. Schilit, and S. Jenson, "Enabling the Internet of Things," *Computer*, vol.48, no.1, 2015, pp. 28–35.
3. Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, IETF RFC 7252, June 2014; https://tools.ietf.org/html/rfc7252.
4. G.Z. Papadopoulos et al., "Adding Value to WSN Simulation Using the IoT-LAB Experimental Platform," *Proc. IEEE 9th Int'l Conf. Wireless and Mobile Computing, Networking and Communications* (WiMOB 13), 2013, pp. 485–490.

ABOUT THE AUTHORS

LAURA BELLI is a PhD student in the Department of Information Engineering at the University of Parma, Italy. Her research interests include the Internet of Things (IoT), cloud computing, and mobile computing. Belli received an MSc in computer engineering from the University of Parma. Contact her at laura.belli1@studenti.unipr.it.

SIMONE CIRANI is a research engineer at Guglielmo srl. He was a postdoctoral research associate in the Department of Information Engineering at the University of Parma when this work was done. His research interests include the IoT, peer-to-peer networks, pervasive computing, and mobile computing. Cirani received a PhD in information technologies from the University of Parma. He is a member of the IEEE Computer Society and the IEEE Communications Society. Contact him at simone.cirani@unipr.it.

LUCA DAVOLI is a PhD student in the Department of Information Engineering at the University of Parma. His research interests include the IoT, security, software-defined networking, and pervasive computing. He received an MSc in computer engineering from the University of Parma. He is a graduate student member of the IEEE Communications Society. Contact him at luca.davoli@studenti.unipr.it.

ANDREA GORRIERI is a PhD student in the Department of Information Engineering at the University of Parma. His research focuses on the IoT, efficient routing in mobile ad-hoc networks, and broadcast/multicast communications protocols. Gorrieri received an MSc in telecommunications engineering from the University of Parma. Contact him at andrea.gorrieri@studenti.unipr.it.

MIRKO MANCIN is a PhD candidate in the Department of Information Engineering at the University of Parma. His research interests include machine-to-machine networks, cloud computing, databases, UI design, IoT protocols, and pervasive computing. Mancin received an MSc in computer engineering from the University of Parma. Contact him at mirko.mancin@studenti.unipr.it.

MARCO PICONE is a research engineer at Guglielmo srl. He was a postdoctoral research associate in the Department of Information Engineering at the University of Parma when this work was done. His research interests include mobile and pervasive computing, location-based services, distributed systems, and the IoT. Picone received a PhD in information engineering from the University of Parma. Contact him at marco.picone@unipr.it.

GIANLUIGI FERRARI is an associate professor in the Department of Information Engineering at the University of Parma. His research interests include the IoT, wireless networking, and signal processing. Ferrari received a PhD in information technologies from the University of Parma. He is a Senior Member of the IEEE Communications Society. Contact him at gianluigi.ferrari@unipr.it.

5. L. Sanchez et al., "SmartSantander: IoT Experimentation over a Smart City Testbed," *Computer Networks*, vol. 61, 2014, pp. 217–238.

6. C. Bormann, M. Ersue, and A. Keranen, *Terminology for Constrained-Node Networks*, IETF RFC 7228, May 2014; https://tools.ietf .org/html/draft-ietf-lwig -terminology-03.

7. G. Mulligan, "The 6LoWPAN Architecture," *Proc. 4th Workshop Embedded Networked Sensors* (EmNets 07), 2007, pp. 78–82.

8. S. Cirani et al., "The IoT Hub: A Fog Node for Seamless Management of Heterogeneous Connected Smart Objects," presented at the Fog Networking for 5G and IoT Workshop, in conjunction with the 12th Ann. IEEE Int'l Conf. Sensing, Communication and Networking (SECON 15), 2015, pp. 464–470; http://secon2015 .ieee-secon.org/workshops/fog -networking-5g-and-iot-workshop /program.

9. S. Cirani et al., "A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things," *IEEE Internet of Things J.*, vol.1, no.5, 2014, pp. 508–521.

10. R. Lea, *HyperCat: An IoT Interoperability Specification*, tech. report, IoT Ecosystem Demonstrator Interoperability Working Group, April 2014; http://eprints.lancs.ac.uk/id /eprint/69124.

11. F. Bonomi et al., "Fog Computing and Its Role in the Internet of Things," *Proc. 1st MCC Workshop Mobile Cloud Computing* (MCC 12), 2012, pp. 13–16.

12. Z. Shelby and C. Bormann, *CoRE Resource Directory*, IETF Internet Draft, Jun. 2015; https://tools.ietf .org/html/draft-ietf-core-resource -directory-03.

13. M. Antonini et al., "Lightweight Multicast Forwarding for Service Discovery in Low-Power IoT Networks," *Proc. 22nd Int'l Conf. Software, Telecommunications and Computer Networks* (SoftCOM 14), 2014, pp. 133–138.

14. S. Cirani et al., "IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios," *IEEE J. Sensors*, vol. 15, no. 2, 2015, pp. 1224–1234.

15. S. Cirani and M. Picone, "Wearable Computing for the Internet of Things," to be published in *IT Professional*, vol. 17, no. 5, 2015.

16. K. Hartke, *Observing Resources in CoAP*, IETF Internet Draft, Dec. 2014; https://tools.ietf.org/html/draft -ietf-core-observe-16.