A User-Centered Application Layer for the Internet of Things

Matthias Kovatsch
Institute for Pervasive Computing, ETH Zurich, Switzerland
kovatsch@inf.ethz.ch

Abstract

The recent years of research in the field of wireless sensor networks (WSNs) have led to the standardization of Internet technology for constrained embedded devices. While consensus was found up to the transport layer to interconnect these devices in an Internet of Things (IoT), an open application layer is yet to emerge. The thesis is devoted to design a Web-like application layer with a common programming model for IoT applications. The key idea is to separate the application logic from the device firmware, so application developers with different backgrounds can provide software for the numerous, diversified device types from many different vendors.

1 Introduction

The last decade saw the emergence of tiny embedded devices, endowed with sensing, actuation, computation, and wireless communication capabilities. They enable gathering information about everyday objects as well as their environment. This information can directly be fed into IT systems or immediately trigger events for actuation, thereby enabling the vision of the Internet of Things: Physical artifacts become interconnected and accessible through existing networks and IT systems, hence the real world becomes linked to the virtual world and vice versa. A large part of these networked devices will influence our personal daily lives, as they will be integrated in household appliances, public and commercial buildings, and even throughout city infrastructures. With information and control available in real-time, novel applications will emerge that increase quality of life, safety, and energy efficiency [21]. The key challenge of this vision is to define an application layer that is open and scalable, and provides a common programming model. The latter is also a key requirement for the adoption of the WSN technology by a broader audience.

SenSys'11, November 1–4, 2011, Seattle, WA, USA.

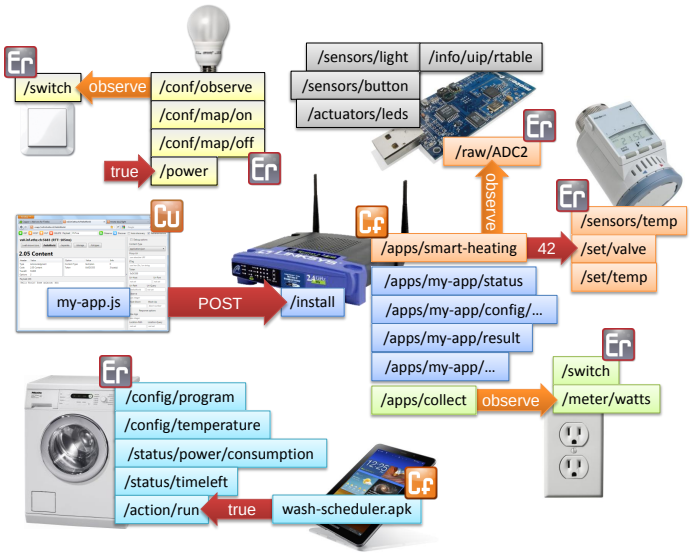

Figure 1. Devices export their hardware functionality through a RESTful, human-readable API. The application logic is separated and runs as 'app' in the Cloud, i.e., on any host with IP connectivity. We prototype this application layer with the Constrained Application Protocol (CoAP), where our Copper (Cu) browser, our Californium (Cf) Java framework, and our Erbium (Er) REST engine for Contiki are the main building blocks.

2 Position

We believe in a strictly layered architecture for the Internet of Things to reduce complexity and maximize interoperability. Energy concerns, for instance, can be handled by an application-independent radio duty cycling layer [10], which also enables multiple applications within the same low-power network. The cross-layer optimizations of previous systems introduced unmanageable complexity, which led to many unsuccessful deployments. Furthermore, each applications required development from scratch. Strict layering, however, might lower the performance and introduce certain application limitations.

We propose a programming model similar to Web applications that builds on standardized protocols. Devices provide their hardware functionality as well as their configuration through a resource-oriented API. Separated from the firmware, the application logic runs atop on any machine in the Cloud, a remote server by a service provider or a local host that is always available, such as a router or a digital video recorder (c.f., Fig. 1). Developers unspecialized in the

embedded domain can then create applications with common languages and tools. Tech-savvy end-users can customize applications, adapt them for their devices, or even create them by themselves when scripting support is provided. The architecture also allows reuse of deployed devices for different applications without changing the firmware. Working with readable URIs and being able to browse device resources, humans can assess the semantics and assure correct bindings and system behavior. We intentionally avoid device profiles and over-specified messages (e.g., ZigBee's Smart Energy Profile 2.0), as a globally defined standard will not scale for the Internet of Things—considering the number of vendors involved. Instead, the interfaces should be human-readable. This enables communities of end-user programmers and developers from independent companies to integrate systems (e.g., for small and medium-sized businesses), and interoperability can reach the same grade as the Web.

3 Related Work

There are two approaches towards the Internet of Things: bottom-up from the device level and top-down pushing Web technology towards devices.

3.1 Wireless Sensor Networks

Within WSNs, it was shown that RFC-compliant IP stacks are feasible for mote-class devices [3, 7], which eventually led to IETF standardizations [13, 23]. With these premises, large-scale low-power wireless networks like the ACme project were deployed successfully and performed well in terms of reliability and throughput.

Most applications, however, still use a custom application layer based on UDP and thus form technological islands, just like traditional WSN deployments. They require application-level gateways for protocol translation and powerful middleware like GSN [1] or sMAP [19] for wide-scale connectivity and easy access to the information. To address this problem, an IETF working group is currently standardizing the Constrained Application Protocol (CoAP) [18]. Being Web-oriented, it leverages the REST architectural style [5], but also provides native push notifications ("observing") and multicast messaging due to UDP. A mapping scheme to HTTP facilitates Web integration through transparent proxies. At the moment, several groups examine the feasibility of a CoAP/6LoWPAN protocol stack [2, 10, 12].

Application development is usually tightly coupled with the operating system, which requires specialists and still is error-prone because of the lack of abstractions. Macroprogramming is an alternate approach that uses frameworks like to program the entire network behavior instead of single nodes [15]. It is well-suited for domain experts who utilize specialized deployments, but does not foster interoperability.

3.2 Web of Things

The 'Web of Things' initiative [6, 22] advocates full Web integration over HTTP to create a user-friendly Internet of Things. Patterns commonly used on the Web, such as REST, linking, bookmarking, and crawling, are adopted to manage the vast number of devices of the Internet of Things. Developers can rely on well-known programming environments, as HTTP libraries are available for many languages and platforms. End-users can create physical mashups in the style of Web 2.0 applications with familiar editors and scripting languages. Due to a large overhead, Web of Things systems mostly depend on application-level gateways as well, but Web servers can also be embedded on resource-constrained devices. While also XML-based Web services were ported through DPWS and EXI [14], it was shown that RESTful services introduce less overhead.

A fundamental problem with this approach is the dependency on HTTP over TCP. Monitoring and automation applications will be central for the Internet of Things and rely on eventing and group communication. In the Web world, many heavy-weight workarounds had to be introduced to provide these features, such as long poll or PubSubHubBub[1] for push notifications. A better solution are WebSockets [4], which are introduced with the new HTML 5 standard. While HTTP provides full integration, the synchronous, one-to-one interaction model of HTTP is still a problem for constrained environments. Numerous embedded devices of the Internet of Things will only have about ten kilobytes of RAM, which limits buffer sizes and the number of open TCP connections.

4 Progress to Date

Our concept was assessed with several publications. The impact of the Internet of Things on home automation scenarios was presented to the automation community [11]. Using a Web API to program a system of devices was demonstrated in [17] and further evaluated with real-world prototypes [16]. Finally, the fundamental idea of our programming model was published for discussion in [9].

The thesis focuses on a 6LoWPAN implementation of the system using CoAP as central application protocol. In a first step, we created Copper [8], a CoAP-based browser, to allow user-interaction with networked embedded devices. For the device level, we use our low-power CoAP implementation for Contiki [10]. It was also used to create first smart appliance prototypes: digital thermostats, smart power outlets, and a networked alarm clock. Finally, we have developed a CoAP framework in Java, Californium,[2] which allows the creation of clients, servers, and proxies that run outside the constrained environment. We are currently integrating scripting support into Californium to provide a user-friendly application server.

5 Thesis Outline

While the qualitative benefits of our application layer and programming model are hard to assess objectively, we want to evaluate the performance in comparison to traditional systems and classify resulting limitations. For this, we are working towards two case studies, while the thesis focuses on the infrastructure, not the applications themselves.

5.1 Smart Appliances

Private homes provide a plethora of heterogeneous device types such as household appliances, multimedia equipment, sport and simple healthcare products, toys, and home automation sensors and controllers. When being part of the Internet of Things, we refer to all these personal devices as

[1] http://code.google.com/p/pubsubhubbub/

[2] https://github.com/mkovatsc/Californium/

'smart appliances'. They will be able to communicate, export their UI, and provide an open API to all their information and functionalities, such as the remaining run time of the washing machine, the light sensor reading of the television set, or fine-grained control of the heating system. We are in the process of building prototype systems as shown in Fig. 1. Appliances run our Erbium (Er) CoAP implementation for Contiki and the application server is implemented with Californium (Cf). We will create different applications:

- App server: We plan to implement several application kernels using JavaScript to evaluate the feasibility of the system for different application classes (data collection, actuation, etc.).
- Distributed electricity metering: Smart power outlets will collect detailed energy consumption data to be evaluated at the application server. This will be a more complex data collection application that will also involve remote control functionality.
- Smart heating control that leverages opportunistic sensing: Digital thermostats and various sensors will be accessible by a Java application that optimizes the energy consumption through occupancy estimation. This is a more detailed study of an actuator and control loop application.

At this point, we cannot answer if we can implement the applications following the strictly layered architecture. The answer will not be binary, but will depend on the system performance.

5.2 Smart Cities

For large-scale applications, we have to consider caching, access control, and Web integration. Thus, we will integrate CoAP proxies and the transparent CoAP-HTTP mapping into our system to address the following issues:

- Scalability: We will run the experiments conducted by Trifa [20] with our system. They examined the system behavior over a varying number of events, publishers, and subscribers as well as different payload sizes.
- Sharing: Proxies allow access to resources while shielding the low-power network. We plan to examine how to implement multi-user access control, especially for actuators, with the current IETF drafts.
- Security: We will only consider the required security architecture based on the corresponding IETF drafts. Details such as how to implement DTLS on mote-class devices is beyond our scope.

This case study contributes policies for optimal caching and allows a performance comparison with the HTTP-based Internet of Things. The latter two points will provide guidelines on how to integrate large-scale IoT applications.

6 Conclusion

By now, we gathered a lot of expertise in technologies for the Internet of Things. We also contributed three comprehensive open-source implementations for the CoAP community. In the doctoral colloquium, we would like to discuss interesting experiments enabled by our CoAP-based infrastructure and suitable metrics to compare the system with previous approaches. Recommendations for the academic twist of our project are highly appreciated.

7 References

[1] K. Aberer, M. Hauswirth, and A. Salehi. A Middleware for Fast and Flexible Sensor Network Deployment. In *Proc. VLDB*, Seoul, Korea, 2006.

[2] A. Castellani, M. Gheda, N. Bui, M. Rossi, and M. Zorzi. Web Services for the Internet of Things through CoAP and EXI. In *Proc. ICC*, Kyoto, Japan, 2011.

[3] A. Dunkels. Full TCP/IP for 8-bit Architectures. In *Proc. MobiSys*, San Francisco, CA, USA, 2003.

[4] I. Fette and A. Melnikov. The WebSocket Protocol. draft-ietf-hybi-thewebsocketprotocol-14, 2011.

[5] R. T. Fielding and R. N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[6] D. Guinard. *A Web of Things Application Architecture - Integrating the Real-World into the Web*. PhD th., ETH Zurich.

[7] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proc. SenSys*, Raleigh, NC, USA, 2008.

[8] M. Kovatsch. Demo Abstract: Human–CoAP Interaction with Copper. In *Proc. DCOSS*, Barcelona, Spain, 2011.

[9] M. Kovatsch. Firm Firmware and Apps for the Internet of Things. In *Proc. SESENA*, Honolulu, HI, USA, 2011.

[10] M. Kovatsch, S. Duquennoy, and A. Dunkels. A Low-Power CoAP for Contiki. In *Proc. MASS*, Valencia, Spain, 2011.

[11] M. Kovatsch, M. Weiss, and D. Guinard. Embedding Internet Technology for Home Automation. In *Proc. ETFA*, Bilbao, Spain, 2010.

[12] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg. Implementation of CoAP and its Application in Transport Logistics. In *Proc.IP+SN*, Chicago, IL, USA, 2011.

[13] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC4944, 2007.

[14] G. Moritz, E. Zeeb, S. Pruter, F. Golatowski, D. Timmermann, and R. Stoll. Devices Profile for Web Services in Wireless Sensor Networks: Adaptations and Enhancements. In *Proc. ETFA*, Mallorca, Spain, 2009.

[15] L. Mottola and G. Picco. Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art. *ACM Computing Surveys*, 43(4), 2011.

[16] B. Ostermaier, M. Kovatsch, and S. Santini. Connecting Things to the Web using Programmable Low-power WiFi Modules. In *Proc. WoT 2011*, San Francisco, CA, USA, 2011.

[17] B. Ostermaier, F. Schlup, and M. Kovatsch. Leveraging the Web of Things for Rapid Prototyping of UbiComp Applications. In *Adj. Proc. UbiComp*, Copenhagen, Denmark, 2010.

[18] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). draft-ietf-core-coap-07, 2011.

[19] D. C. Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz. sMAP: A Simple Measurement and Actuation Profile for Physical Information. In *Proc. SenSys*, Zurich, Switzerland, 2010.

[20] M. V. Trifa. *Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks*. PhD thesis, ETH Zurich, 2011.

[21] J.-P. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[22] E. Wilde. Putting Things to REST. Technical Report 2007-015, School of Information, UC Berkeley, Berkeley, CA, USA, 2007.

[23] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. P. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. draft-ietf-roll-rpl-19, 2011.

Biographical Sketch

Matthias Kovatsch has a background in electrical engineering and computer science through an interdisciplinary study course within the Elite Network of Bavaria. He received the academic degree 'Diplomingenieur' (equiv. to M.Sc.) from the University of Erlangen-Nuremberg, Germany in November 2008. During his studies, he was working with Siemens Pte Ltd Singapore and the Fraunhofer Institute for Integrated Circuits (IIS). At the latter, he also conducted his diploma thesis "Services for Wireless Sensor Networks – A Design for Health Monitoring and Motion Analysis."

Since March 2009, he is a Ph.D. student at ETH Zurich, Switzerland, where he joined the Distributed Systems Group at the Institute for Pervasive Computing. From March to June 2011, he was also a visiting researcher at the Swedish Institute of Computer Science (SICS). Since May 2011, Matthias Kovatsch is an accredited Ph.D. candidate under the supervision of Prof. Dr. Friedemann Mattern. The expected date of dissertation is summer 2013.