Firm Firmware and Apps for the Internet of Things

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

ABSTRACT

Among the challenges for the Internet of Things, two stand out: a scalable application layer with wide interoperability and a common, reliable programming model. We propose to strip all application logic from the firmware and only provide a RESTful interface to the hardware functionality. Critical parts such as the network stack remain in the immutable firmware that is maintained by experts. Applications are developed atop the resource abstraction and run in the cloud. Leaving the embedded domain, application development is eased while sharing and customizing applications helps to cope with the vast number of diversified device types.

Categories and Subject Descriptors

C.2.4 [**Computer-Communications Networks**]: Distributed Systems; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Design, Human Factors

Keywords

Internet of Things, WSNs, Programming model, CoAP

1. INTRODUCTION

The Internet as we know it today, with more than a million of nodes in its core (servers, core routers, etc.) and about a billion of nodes on its fringe (PCs, smart phones, etc.), will soon be extended with a trillion of nodes at its edge. Providing sensing and actuation capabilities, these tiny nodes will be embedded into everyday objects such as household appliances. This will facilitate the emergence of novel wireless sensor network (WSN) applications: Ad hoc and personal solutions will leverage sensors and actuators which are already deployed such as the light sensor of a television set, the temperature sensor of a dryer, or air quality and traffic sensors throughout a city infrastructure.

Two of the biggest challenges for this Internet of Things will be a scalable application layer for interoperability and a common programming model for application developers with very different backgrounds. Both are hard to achieve because the nodes will be heterogeneous and part of the resource-constrained, low-level embedded domain. Traditionally, WSN applications are optimized for this domain and mostly introduce custom protocols. Application code is directly nested into the embedded operating system and thus requires deep knowledge of the latter. This approach promotes isolated systems and error-prone software due to the lack of abstractions. As an alternative, a new initiative, the 'Web of Things', tries to leverage HTTP and a resource-orientated architecture for interoperability, and to adopt common Web principles. However, the synchronous, one-to-one communication model of HTTP requires heavy-weight workarounds to satisfy typical WSN applications.

In this paper, we argue for a combination of the two approaches to address the stated challenges. The ongoing standardization of WSN research results allows for efficient protocols as well as interoperable interfaces that can be reused across different applications. The firmware can provide RESTful resources to abstract and export hardware functionalities, with the Constrained Application Protocol (CoAP) playing a key role. Applications can then be developed outside the embedded domain and run in the cloud. For instance, a smart heating application could be developed in any language, a CoAP library provided, and would simply subscribe to sensor resources and control actuators in a RESTful manner (e.g., `coap://sensor1/temperature` and `coap://heater1/power`, resp.). Ideally, the application is deployed (e.g., via POST to `coap://appserver/install`) and configured in the same way (e.g., `coap://appserver/apps/smart-heating/config/comfortTemperature`).

2. STATE OF THE ART AND ISSUES

The field of WSNs is taking a bottom-up approach to the Internet of Things. It was shown that RFC-compliant IP stacks are feasible for 8-bit microcontrollers [7]. Since 2007, IPv6 solutions for tiny embedded devices have been developed, which led to the IETF standardization of 6LoWPAN. Currently, many work groups are drafting complementary Request for Comments (RFCs) which build upon research results from the WSN community. With this basis, large-scale WSNs are deployed that perform well in terms of reliability and throughput, e.g. ACme [2]. However, such deployments typically use custom protocols based on UDP that require matching application-level gateways. On top, powerful middleware such as SenseWeb or GSN then allows for easy access. Lately, the convergence of systems gains focus. sMAP [6] introduces a common REST layer for 6LoWPAN resources, building automation systems, and cloud services. A preset resource hierarchy and JSON objects for interaction are, however, too restrictive for a generic application layer for the Internet of Things. To cope with this problem, an IETF working group is currently standardizing the Constrained Application Protocol (CoAP) [5]. It enables building an open, Web-oriented application layer with native push notification and multicast support. Still, developing a distributed WSN application remains very complex. A way to address this challenge is to program the entire network behavior instead of single nodes, the so-called macroprogramming [4]. Often, this forces programmers into design patterns that are uncommon, however, intuitive for domain experts who utilize dedicated WSN deployments.

A top-down approach to a more user-friendly Internet of Things is taken by 'Web of Things' projects [1]. They advocate full Web integration over HTTP and enable users to create physical mashups in the style of Web 2.0 applications. Patterns commonly used on the Web, such as bookmarking, linking, and crawling, allow users to cope with the vast number of devices. While also XML-based Web services were ported to resource-constrained devices through DPWS [3], RESTful services were shown to be more efficient [8]. A fundamental problem, however, is the one-to-one interaction model of HTTP over TCP. Monitoring and automation applications will be central for the Internet of Things and rely on push notifications and group communication. In the Web world, many workarounds had to be introduced, e.g., long poll for eventing or PubSubHubBub for publish/subscribe. These are, however, inefficient and thus unfavorable for inexpensive embedded devices, i.e. 'things'.

3. POSITION AND WORK IN PROGRESS

Once deployed, the hardware of wireless sensor nodes does not change and applications always come down to sampling input ports or switching output ports. Hence, the firmware can remain immutable (i.e., 'firm') for every kind of application as long as all hardware functionality is accessible. The stable firmware is maintained by system experts and only hosts the device drivers, network stack, and a server process providing an all-purpose interface: RESTful resources abstract whole functional units with all their parameters (e.g., `coap://dev/sensors/temperature` and `coap://dev/conf/temperature/samplingInterval`, resp.) or, for instance for rapid prototyping, offer raw hardware features (e.g., `coap://dev/hw/uart0`, `coap://dev/conf/uart0/baudrate`, etc.).

The application code composes these resources and runs in the cloud, keeping devices simple and low-cost. In-network processing should also be avoided on sensor and actuator nodes for simplicity reasons. More powerful local devices, however, could also run applications or modules such as data collection. This is similar to the concept of cluster heads, thereby enabling a form of in-network processing.

Extracting the application code from the firmware significantly eases development. Programmers are not required to be familiar with embedded design, can focus on the functional requirements, and can resort to the programming environment of their choice. Application modules or 'apps', such as the mentioned data collection or the automation of a specific device, can be reused for many deployments. More complex and specific objectives can be achieved by combining and tweaking the modules. With scripting support, even tech-savvy end-users can be empowered to customize applications or port them to their devices that offer comparable functionality. This tackles the problem of device heterogeneity with a crowdsourcing approach. Thus, the community would benefit from the concept of an app store or market place known from smart phones where 'IoT apps' can be shared or offered by service providers and vendors.

To evaluate the strength and applicability of our programming model, we are working on a case study of smart appliances. The scenario is complex, providing a plethora of device types, performance requirements for user interaction, as well as security and privacy aspects.

4. CONCLUSIONS

Devices of the Internet of Things must be interoperable to gain a real advantage over the current situation where networked embedded systems are isolated islands. We cannot, however, specify every 'thing' in a globally supported standard due to the vast number of slightly different devices provided by many different manufacturers. Thus, to cope with the heterogeneity, we need an open architecture that allows developers with very different skills—including tech-savvy end-users—to create and customize applications. Separating the application code from the firmware moves the application development out of the embedded domain and thus significantly eases the process. The required interface can be provided by CoAP, which combines the efficiency of WSN protocols and the openness of the Web. Becoming a long-lived IETF standard, the protocol would allow for interoperability and would facilitate a common programming model for the Internet of Things.

5. REFERENCES

[1] D. Guinard, V. Trifa, and E. Wilde. A Resource Oriented Architecture for the Web of Things. In *Proc. IoT*, Tokyo, Japan, 2010.

[2] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and Implementation of a High-Fidelity AC Metering Network. In *Proc. IPSN*, Washington, DC, USA, 2009.

[3] 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.

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

[5] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). draft-ietf-core-coap-04, 2011.

[6] 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.

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

[8] D. Yazar and A. Dunkels. Efficient Application Integration in IP-Based Sensor Networks. In *Proc. BuildSys*, Berkeley, CA, USA, 2009.