Simply RIOT: Teaching and Experimental Research in the Internet of Things

Oliver Hahm, Emmanuel Baccelli, Hauke Petersen, Matthias Wählisch, Thomas Schmidt

Simply RIOT: Teaching and Experimental Research in the Internet of Things

Oliver Hahm, Emmanuel Baccelli
INRIA, France
{first.last}@inria.fr

Hauke Petersen, Matthias Wählisch
Freie Universität Berlin, Germany
{first.last}@fu-berlin.de

Thomas C. Schmidt
HAW Hamburg, Germany
t.schmidt@ieee.org

***Abstract*—Manufacturers announce on a regular basis the availability of novel tiny devices, most of them featuring network interfaces: the Internet of Things (IoT) is already here – from the hardware perspective. On the software side however, embedded platforms available so far made it uneasy for developers to build apps that run across heterogeneous IoT hardware. Linux does not scale down to small, energy-constrained devices, while alternatives such as Contiki yield a steep learning curve and lengthy development life-cycles because they rule out standard programming and debugging tools. RIOT is a new open source software platform bridging this gap. RIOT allows just about any programmer to develop IoT application with zero learning curve. This is achieved by allowing standard C and C++ application programming with multi-threading, using well-known debugging tools (gdb, Valgrind, profilers etc.), while requiring only a minimum of 1.5 kB of RAM. RIOT also provides built-in energy efficiency and real-time capabilities. These characteristics make this platform attractive in several contexts, including teaching in the field of the Internet of Things, and experimental research in the domain of sensor networks and the IoT.**

***Keywords*—Education, experiments, testbeds, operating system, IoT, WSN**

I. Introduction

Real-world systems allow us to complement insights gained from analytical studies and simulations. On one hand, researchers engage into real deployments of their approaches to show the applicability of mechanisms *in vivo*. On the other hand, practical experience typically helps students increase their involvement. Easy prototyping of network protocols and applications is thus needed in both, scientific experiments [1] and teaching scenarios [2] to better understand existing as well as future technologies.

Common tools that we know from the networking field for practical experiments cannot easily be applied in the field of Internet of Things (IoT). The IoT is composed of networks of heterogeneous, constrained hardware, e.g., wireless sensor networks (WSN). To comply with these constraints, most of the current operating systems such as Contiki or Tiny OS implement a system-specific layer that lacks native C/C++ support, full multi-threading, real-time support – standard functionalities which allow easier implementation of complex tasks.

In this paper, we argue that experiment-driven evaluation and hands-on teaching in the field of wireless communication can greatly benefit from an operating system which provides the principle programming functions expected on common Internet hosts. Our demo shows how RIOT, a new open source operating system for the Internet of Things, can be used to easily develop IoT scenarios, i.e., a smart watch communicating with commodity lightning system. We demonstrate key features of RIOT such as support for multi-threading, C/C++, virtual network topologies, open testbed integration, and application of standard debugging tools. Furthermore, we report about our first experiences with RIOT in teaching and experimentation.

II. Why Do We Need a New Platform?

a) Research Perspective: PhD students and senior researchers have specific expertise and are able to handle WSN-specific system environments. They benefit from systems that provide the following advantages (a) allowing reuse of implementations to increase comparability between different platforms and (b) setup of complex network environments using virtual networks, testbeds, or hybrid approaches which link simulations with field tests.

b) Teaching Perspective: In contrast to PhD students, bachelor and master students have only basic knowledge in system programming and little time to get deeply involved in specific programming environments. Courses on embedded systems discuss the characteristics of the WSN system landscape but typically this is not part of regular networking classes. When integrating lab exercises in courses, the corresponding platform should typically support (a) less initial training and (b) basic set of *standard* functionalities. Non-technical aspects e.g. a lively open source community may also increase the motivation of students to work with the platform and integrate their own contributions. In this context, Linux makes first steps easier as it is open and well-known. However, Linux is not suitable for CPU/memory constrained devices. Furthermore, the Linux kernel and network stack are quite complex, which conflicts with low entry level for students.

Researchers as well as students benefit from easy debugging and low-cost development. The latter does not only refer to low learning curves but also to the development on commodity hardware.

III. RIOT in a Nutshell

RIOT [3], [4] is an open source software platform that bridges the gap between operating systems for WSNs and full-fledged operating systems currently running on Internet hosts. In the following we present key features of the platform, which we will showcase.

a) Architecture Overview: RIOT provides a uniform programming interface across a wide range of devices, allowing multi-threading with standard POSIX API with very small memory footprint, starting from 1,5kB RAM and 5kB ROM (without network stack). By design it provides energy-efficiency, reliability, and real-time capabilities, based on a modular, microkernel architecture.

RIOT implements a microkernel architecture. In addition, RIOT add native support for C/C++ and provides a TCP/IP network stack. Advantages of the RIOT architecture thus include: (i) high reliability and (ii) a developer-friendly API. The modular microkernel architecture of RIOT makes it robust against bugs in single components. Failures in the device driver or the file system, for example, will not harm the whole system. RIOT allows developers to create as many threads as needed and distributed systems can be easily implemented by using the kernel message API. The amount of threads is only limited by the available memory and stack size for each thread, while the computational and memory overhead is minimal.

On the high end in terms of hardware CPU and memory capacities, RIOT competes mainly with Linux. Compared to Linux, RIOT can scale down to orders of magnitude less memory requirements and supports built-in energy efficiency and real-time capabilities. On the low end in terms of hardware CPU/memory capacities, RIOT competes mainly with Contiki, TinyOS, and FreeRTOS. Compared to Contiki and TinyOS, RIOT offers real-time capabilities and multi-threading. In contrast to FreeRTOS, RIOT provides native energy efficiency and a full-featured OS including up-to-date, free, open-source interoperable network stacks (e.g., 6LoWPAN), instead of just a kernel. RIOT also offers standard POSIX APIs and the ability to code in standard programming languages (C and C++) using standard debugging tools, thus reduces the learning curve of developers and the software development lifecycle process.

b) Standard Protocols and Libraries: Default protocol integration in RIOT is mainly driven by latest IETF/IRTF activities not only to guarantee standard-compliant communication but also to have a solid base for researchers [5]. Currently, RIOT supports basic networking protocols including 6LoWPAN, RPL, IPv6, TCP, UDP, CoAP, and provides CCN-lite to experiment with content-centric networking.

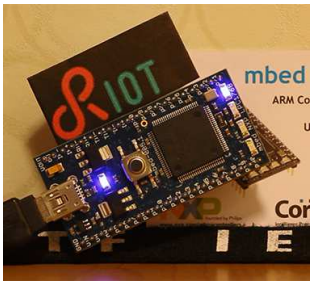
The C++ capabilities of RIOT enable powerful libraries such as the Wiselib, which includes algorithms for routing, clustering, timesync, localization, and security.

c) Native Port: To overcome the issue of specialized hardware availability, RIOT can also be run as a native process on Linux and MacOS. This facilitates development because such a native process can be analyzed using readily available tools (e.g., gdb, Valgrind). A RIOT process is accessible via shell, the UART interface, or the virtual link layer interface (TAP).

d) Large-scale Experiments: RIOT runs on hardware used in several open testbeds, e.g. SensLAB and DES. SensLAB is distributed among four sites involving more than 1,000 nodes. DES-Testbed follows a hybrid approach providing more powerful embedded PC boards connected to wireless sensor nodes. Our experiences shows that it is easy to integrate RIOT in existing testbeds.

In addition to testbed environments, RIOT allows for the setup of arbitrary virtual networks. Multiple RIOT native processes can run on a single PC and are automatically interconnected by TAP interfaces. A generator for common topologies is available using DES-Virt. The emulation of network properties such as packet loss is provided by netem.

(a) RIOT on Texas Instruments EZ430-Chronos



(b) RIOT morses R-I-O-T letters on mbed NXP LPC1768

Fig. 1. RIOT running on heterogenous IoT hardware

e) Contributing to the RIOT Community: The RIOT community is explicitly open for new contributions, making it particularly suitable for students to gain professional experiences early. The software is licensed under LGPLv2 and hosted on GitHub (https://github.com/RIOT-OS/). Transparent development procedures and coding conventions control updates of the master branch. New features and fixes to the master branch are submitted as pull requests and verified by RIOT maintainers. Regular meetings including remote participation coordinate long-term development strategies. This mimics the IETF spirit i.e., participation is open to all interested people, including bachelor and master students.

IV. Demonstration

The demonstration scenario consist of a sensor, a watch, a simple lightning system, and a PC. All three components interact together to show seamless and standard-compliant integration of IoT devices in the current Internet. An external sensor recognizes physical intrusion and alarms the other components. The watch is able to turn on and off the lightning system. The PC implements a more complex management software. This network is extended by a larger virtual network to show more complex ad hoc routing.

Acknowledments: This work is partly supported by ANR and BMBF within the project SAFEST (http://safest.realmv6.org), and by the German Academic Exchange Service (DAAD).

References

[1] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," in *Proc. of the 4th International Symposium on IPSN*. Piscataway, NJ, USA: IEEE Press, 2005, pp. 483–488.

[2] S. S. Panwar, S. Mao, J. dong Ryoo, and Y. Li, *TCP/IP Essentials. A Lab-Based Approach*. New York, USA: Cambridge University Press, 2007.

[3] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. of the 32nd IEEE INFOCOM. Poster*. Piscataway, NJ, USA: IEEE Press, 2013.

[4] "RIOT - The friendly OS for the IoT." [Online]. Available: https://www.riot-os.org

[5] A. Y. Ding, J. Korhonen, T. Savolainen, M. Kojo, J. Ott, S. Tarkoma, and J. Crowcroft, "Bridging the Gap Between Internet Standardization and Networking Research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 1, pp. 56–62, Jan. 2014.