

Demo Abstract: uDPWS - The Devices Profile for Web Services for Resource-Constrained Devices

Christian Lerche, Nico Laum, Guido Moritz, Elmar Zeeb, Frank Golatowski, Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
`{firstname.lastname}@uni-rostock.de`

Abstract—Web Service technologies are state of the art in distributed applications. To integrate highly resource-constrained devices, such as sensor nodes, into distributed applications via Web Services, gateways are commonly used. We present a novel solution, that enables the intermediate, homogeneous communication between enterprise applications and highly resource-constrained devices. Therefore, our demonstration presents uDPWS, a prototype implementation of an embedded enterprise compatible W3C Web Services profile. uDPWS is designed to work on sensor nodes. It requires only 8.5 KiB of ROM and 3 KiB of RAM.

I. INTRODUCTION

A. Motivation

Recent investigations on IP-based communication for Wireless Sensor Networks (WSNs) promise to bring syntactic interoperability at the network layer of low-power wireless communication systems in the near future. In this area, the IETF 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) working group has defined, how efficient IPv6-based communication can be accomplished on top of IEEE 802.15.4 based wireless radio [1]. It is questionable, however, how syntactic interoperability can then be achieved at the higher application layers. From our point of view, Web Services are one of the most promising technologies to be used as application protocol in this domain. Therefore, we demonstrate that even complex W3C Web Services can run on sensor nodes. In conclusion, our demonstration shows, how a pulse oximetry sensor can be wirelessly connected to a IP-network and further be used on any kind of a network connected Host PC.

B. Devices Profile for Web Services

The Devices Profile for Web Services (DPWS) defines a subset of the WS-* specifications suitable for “resource-constrained endpoints” [2]. Even though there is no information about performance prerequisites given in the DPWS standard, it is obvious that devices with around 10 KiB of RAM are out of scope. DPWS proposes a Web Services based architecture for networked embedded devices and tries to foster interoperability between small embedded devices and fully Web Service-capable computers. The DPWS standard states that a DPWS-Device is represented through a

Web Service which is called Hosting Service. The Hosting Service is found by DPWS-Clients in local networks using WS-Discovery. Furthermore, the Hosting Service provides the device description through WS-MetadataExchange. The device offers access to its functionality through Hosted Services. For publish-subscribe communication DPWS uses WS-Eventing.

C. Application Domain and Scenario

The presented demonstration is part of the German OSAMI project demonstrator [3]. The aim was to enable interoperability in distributed systems in the e-health and ambient intelligence domain. The demonstrator allows patients to perform an ergometer bicycle training at home under medical supervision, for example, in ambulant cardiac rehabilitation.

II. UDPWS IMPLEMENTATION

We implemented uDPWS¹, a very lightweight implementation of the DPWS standard, suitable for sensor nodes. uDPWS is designed to implement DPWS-Devices using the Contiki operating system [4]. As a result, uDPWS runs on all Contiki supported platforms, such as the Tmote Sky Board, Atmel RZ Raven, the Modular Sensor Board (MSB430) and the Embedded Sensor Board (ESB). For IP, as well as TCP and UDP communication, the uIPv6 TCP/IP stack and SICSLOWPAN for IETF 6LoWPAN [1] header compression is utilized. We demonstrate that enterprise Web Services can run directly on sensor node hardware. This enables internet- or network-connected computers to address sensor-nodes based on their IPv6 address and connect to their published Web Service interfaces.

The modular architecture of uDPWS is shown in Figure 1. The core (highlighted in grey) consists of the following four main modules:

1) *The uDPWS-Process-module*: handles the distribution of incoming connections and messages. It is the only module that interacts with the operating system (OS) Contiki. Thus, porting to other operating systems, such as TinyOS, can be accomplished by adopting the uDPWS-Process module.

¹uDPWS is open source and available at <http://www.ws4d.org/udpws/>

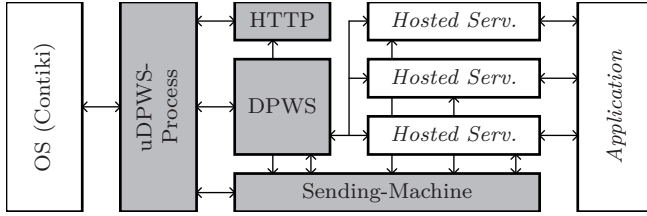


Figure 1. uDPWS Architecture

Contiki comes with an embedded TCP/IP stack and an implementation of 6LoWPAN which provides standardised IPv6 based communication over IEEE 802.15.4.

2) *The HTTP-module:* is responsible for receiving and buffering the request message. As soon as the complete message is buffered, it is forwarded to the DPWS-module.

3) *The DPWS-module:* forwards the message content to the corresponding service implementation, if the message is dedicated to one of the Hosted Services. Otherwise, if the message is a DPWS message (e.g., WS-MetadataExchange, WS-Discovery), the DPWS module itself handles the message. The DPWS module is the main module of the uDPWS-Core

4) *The Sending-Machine-module:* is finally used to create the response messages in a very resource-saving manner: As most parts of the XML messages are invariant (e.g., XML namespaces) and are already stored in ROM, this module avoids a separate response message buffer by concatenating memory references of XML message fragments to a list. Therefore, only dynamic message fragments (e.g., measurement values) need to be buffered in RAM.

For implementing a Device and its Hosted Services, the DPWS-module provides a generic API. The application itself only uses the interfaces of the Hosted Services, which can be designed in any way.

III. DEMONSTRATION SETUP

In Figure 2 the demonstration setup is shown. It consists of an Atmel RZ Raven sensor node which is connected to a Corscience ChipOx [5] pulse oximetry sensor. The RZ Raven runs a DPWS-Device that offers two hosted Web Services: (a) a SpO₂ Service to retrieve the patient's blood oxygen saturation and (b) a Pulse Service to retrieve the patient's heart rate. As 6LoWPAN is used for IP communication over IEEE 802.15.4, a TelosB mote serves as IPv6-over-IEEE 802.15.4 interface for the Host PC. The IPv6 packets are transmitted from the TelosB to the Host PC via USB by using the Serial Line Internet Protocol (SLIP).

The DPWS-Client, used in this demonstration, is the DPWS-Explorer [6], a generic DPWS-Client to find DPWS-Devices in local networks. The DPWS-Explorer generates a graphical interface to the Hosted Services of a DPWS-Device. It is used to retrieve and show the real-time pulse oximetry values of the RZ Raven sensor node.

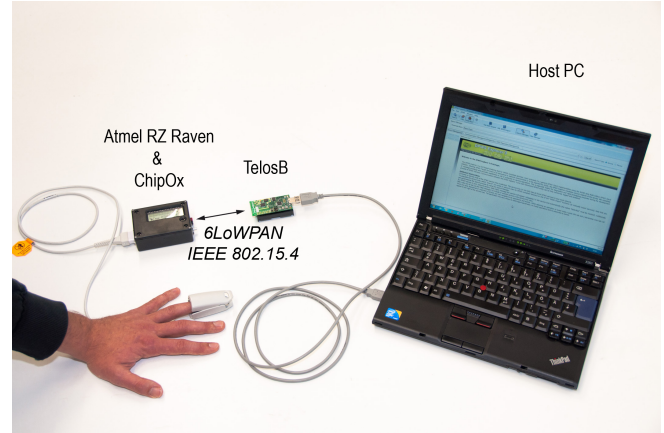


Figure 2. uDPWS Demonstration Setup

IV. ACKNOWLEDGEMENT

This work has been achieved in the European ITEA2 projects uSERVICE and OSAMI and has been funded by the German Federal Ministry of Education and Research under contract numbers 01—S0902F and 01—S08003I.

REFERENCES

- [1] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Internet Engineering Task Force, Sep. 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4944.txt>
- [2] D. Driscoll and A. Mensch, "Devices profile for web services version 1.1," Tech. Rep., July 2009. [Online]. Available: <http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf>
- [3] M. Lipprandt, M. Eichelberg, W. Thronicke, J. Krueger, I. Druke, D. Willemsen, C. Busch, C. Fiehe, E. Zeeb, and A. Hein, "OSAMI-D: An open service platform for healthcare monitoring applications," in *Human System Interactions, 2009. HSI '09. 2nd Conference on*, May 2009, pp. 139–145.
- [4] (2011) The Contiki OS. [Online]. Available: <http://www.sics.se/contiki/>
- [5] (2011) ChipOx Development-Kit. [Online]. Available: <http://www.corscience.de/de/medizintechnik/oemodm-loesungen/oem-module/chipox-development-kit.html>
- [6] (2011) DPWS-Explorer. [Online]. Available: <http://www.ws4d.org/dpws-explorer/>