Postprint

This is the accepted version of a paper presented at *14th ACM Conference on Embedded Network Sensor Systems*.

N.B. When citing this work, cite the original published paper.

Deploying a 6LoWPAN, CoAP, low power, wireless sensor network

Arthur Fabre, Kirk Martinez, Graeme M. Bragg, Philip J. Basford
Electronics and Computer Science
University of Southampton
{af1g12, km, g.bragg, pjb}@ecs.soton.ac.uk

Sebastian Bader
Electronics Design
Mid Sweden University
sebastian.bader@miun.se

Jane Hart
Geography and Environment
University of Southampton
j.k.hart@soton.ac.uk

Olivia M. Bragg
Environment
University of Dundee
o.m.bragg@dundee.ac.uk

ABSTRACT

In order to integrate equipment from different vendors, wireless sensor networks need to become more standardized. Using IP as the basis of low power radio networks, together with application layer standards designed for this purpose is one way forward. This research focuses on implementing and deploying a system using Contiki, 6LoWPAN over an 868 MHz radio network, together with CoAP as a standard application layer protocol. A system was deployed in the Cairngorm mountains in Scotland as an environmental sensor network, measuring streams, temperature profiles in peat and periglacial features. It was found that RPL provided an effective routing algorithm, and that the use of UDP packets with CoAP proved to be an energy efficient application layer. This combination of technologies can be very effective in large area sensor networks.

CCS Concepts

•Computer systems organization → Sensor networks; Embedded software;

Keywords

6LoWPAN, CoAP, low power, sensor network, deployment

1. INTRODUCTION

While wireless sensor networks have advanced in many areas, issues of standardization and heterogeneity are ongoing challenges. One possible solution to networking standardization is to use 6LoWPAN, which brings IPv6 compatibility and tools, with 802.15.4 physical and MAC layers. The next issue is using a standard application layer suitable for

data transfer over low bandwidth radio links. A good candidate for this is CoAP[3], which only uses UDP and replicates a REST HTTP-like interface optimized for low data rates through the use of binary requests. Data must also be encoded in an efficient, extensible, cross-platform manner, which can be achieved using Protocol Buffers[2]. We have implemented a complete environmental sensor network system and deployed it in the Cairngorm mountains of Scotland in order to test the feasibility and effectiveness of such a standards compliant but low energy design.

2. NODES

The sensor nodes consist of a main processor board, and a carrier board which can be connected to a number of smart sensors, as shown in Figure 1. The processor board features a microcontroller; flash for storing samples and the configuration. The carrier board provides a sub-GHz radio, real time clock, RS485 driver, and power supplies.

The first generation of nodes used a Zolertia Z1 (MSP430) as the main processor board, and were powered from 12V lead-acid batteries. The MSP430 proved to be RAM constrained, consequently a second generation of nodes was designed using an ARM Cortex-M3. This generation maintained pin-compatibility with the first generation, allowing for a heterogeneous network. Power is delivered either through a separate power controller or from a 12V lead acid battery. The separate power controller features an ARM Cortex-M0+, that handles the maximum power point tracking of a 5W solar panel, and LiPo battery charging. It also provides power to the smart sensors.

3. SMART SENSORS

The smart sensors all implement a common interface and protocol, using RS485, and feature a unique identifier. The nodes are configured with the identifiers of connected sensors, and receive a binary payload (typically a Protocol Buffer), that is stored as is. Its contents are decoded server-side. Consequently, new sensors can be created and deployed with no changes to node firmware or hardware. Typical sensors use an AVR to aggregate data from multiple sensors, such as an array of thermometers or accelerometers. The same protocol is used to communicate with the power controller.

Figure 1: Block diagram of the sensor nodes

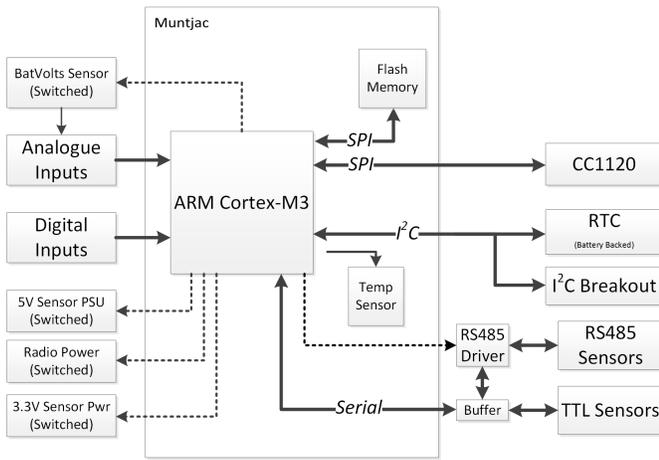**Figure 2: Default RPL routes**

4. SOFTWARE

The node firmware is based on Contiki-OS, as it supports 6LoWPAN, RPL and CoAP (through Erbium). Drivers were written to support the CC1120, flash, RTC and RS485 peripherals. A common abstraction layer was developed, allowing multiple platforms to be supported by the firmware. A sampling process uses the real time clock to ensure that nodes always begin sampling at predictable times. In order to maximize sample storage, Coffee file system headers were reduced by removing micro log related entries, allowing a sample to fit in a single logical page (128 bytes). This allows up to 16,384 samples to be stored, with a typical size of 100 bytes.

5. NETWORK

Our previous research[1] led us to use an 868 MHz radio based on the CC1120 transceiver. This provides the long range links required (up to 3km) at an acceptable data rate (50 kbit/s) and is a commonly used frequency for environmental sensing. The nodes form a mesh network using RPL, with a border-router providing public IPv6 connectivity.

The network is built up from a number of layers:

- 802.15.4: Physical & MAC layer.
- ContikiMAC: Provides low energy radio duty cycling, allowing the radio to receive incoming packets at any time, whilst significantly reducing the power usage.
- 6LoWPAN: Encapsulates IPv6 traffic.
- IPv6 / Routing Protocol for Low power and Lossy Networks (RPL): Network layer: Sets up the routing for the network, enabling nodes to discover neighbors, and select the best available route to the border router.
- UDP: Transport layer. Provides low latency, stateless data transport. Reliability is not provided, in contrast with TCP, as it is offloaded to the application layer.
- CoAP: Application layer. Provides a stable, discoverable API to the nodes.
- Protocol Buffers: Provide an extensible encoding for samples and the node configuration.

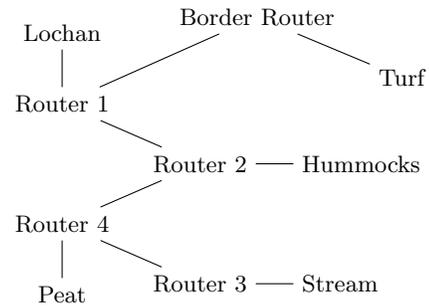The gateway micro-PC running the border-router GETs and DELETEs available samples from the nodes using a CoAP API. The Protocol Buffer encoded samples are then pushed over a satellite link to a MySQL database, where they are decoded. GETing from the nodes instead of having the nodes POST, allows centralized network scheduling thereby alleviating the strain of multiple nodes sending samples simultaneously through the network. Erbium also lacked support for blockwise POST requests to clients at the start of the project, limiting POSTable samples to 64 bytes.

6. DEPLOYMENT

Nine second generation nodes are currently deployed in the Cairngorms, replacing the original eight[1]. Their default routes can be seen in Figure 2. Nodes sample every 20 minutes, and data is fetched from them hourly.

6LoWPAN and RPL have allowed a mix of hardware to be deployed, the border router is MSP430 based, while the remaining nodes are second generation ones.

The use of CoAP has allowed us to deploy nodes with heterogeneous firmwares, using undefined resources returns an error code indicating the resource is not present. The Protocol Buffer encoding also allows the introduction of new fields, for example a power board ID was added to the configuration to support second generation nodes, with no ill effects on previously deployed nodes.

End to end IPv6 connectivity eases system maintenance, allowing nodes to be configured, monitored and rebooted from computers in the lab. Standard network tools such as ping can also be used.

7. CONCLUSION

The deployment has shown that using 6LoWPAN together with CoAP brings many benefits compared to the previous approaches using a selection of WSN algorithms. The low throughput of the 868 MHz radio (approx 20 kbit/s) was not a hindrance to using IP protocols and IETF standards, while providing long range, heterogeneity and ease of deployment.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] G. Bragg, K. Martinez, P. Basford, and J. Hart. 868MHz 6lowpan with contikimac for an internet of things environmental sensor network. 2016.
[2] Google. Protocol buffers, 2016.
[3] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (coap). Technical report, 2014.