WIRELESS FREIGHT SUPERVISION USING OPEN STANDARDS

Markus Becker, Koojana Kuladinithi, Thomas Pötsch, Carmelita Görg

Communication Networks, TZI, University of Bremen

Email: [mab|koo|thp|cg]@comnets.uni-bremen.de

Keywords: Freight Supervision, Wireless Sensor Network, Open Standards, Internet

Abstract: The supervision of the environment of freight goods directly at the goods gives a better shelf-life estimation than monitoring at the inlet or outlet of the refrigeration unit. Implementing the monitoring spatially close to the good is typically more complex for the logistic service provider. Either this is done using wires or so-called data-loggers. Wired sensors are usually only possible at static positions defined at installation time. Data-loggers only provide post-transport inspection. Within the *Intelligent Container* project Wireless Sensor Networks which allow inspection and event triggers are employed during the transport, thus reducing the exposure time to adverse transport conditions. Wireless Sensor Networks have typically been using closed and/or dedicated protocols for the supervision. Lately, there have been standardization activities around Wireless Sensor Networks in the Internet Engineering Task Force (IETF), that allow for networks of inter-operable hardware of different vendors. Additionally, the newly standardized protocols (6LoWPAN, RPL, CoAP) allow the easy integration of Wireless Sensor Network with the Internet and logistical applications. We will introduce the standardization efforts of the protocols relevant to the logistic application field and give a brief introduction to those protocols.

1. Introduction

Wireless Sensor Networks (WSNs) are networks of wirelessly communicating networks of nodes which have the ability to measure physical variables. The individual nodes are usually embedded devices consisting of a microcontroller, memory, a wireless network interface, a programming interface and batteries. WSNs can be applied in many application fields, e.g. Nature Monitoring, Smart Home/Office, Logistics, Agriculture, Traffic Management, and Healthcare [1]. In the research project *Intelligent Container*, WSNs are deployed to supervise the transport conditions of food.

Today, there exist several standards-based and proprietary protocols that can be used in WSN applications. Proprietary technologies are difficult to integrate into larger networks and also with currently available Internet-based services. Therefore, existing Internet Protocols (IPs) are used and enhanced to realize the WSN deployed in the *Intelligent Container* project. In general, the use of IP based technologies gives the following benefits:

- It allows the use of existing network infrastructure. IP based devices can be connected easily to other IP networks
- IP-based technologies have existed for decades, are very well known and have been proven to work and scale
- IP technology is specified in an open and free way, with standard processes and documents available to anyone
- Tools for managing, commissioning and diagnosing already exist

The WSN consists of a very constrained environment with limited processing power, energy, and data rate. Therefore, the existing IP based technologies cannot be applied in WSNs as they are. When deploying IP based technologies in WSNs, the following issues should be addressed:

- Bandwidth and frame size: Current IP networks require links with sufficient frame length (e.g. minimum of 1280 bytes for IPv6), while WSNs offer a low frame size (e.g. the IEEE 802.15.4 frame size is 127 bytes) and usually have limited bandwidth (e.g. 20-250 kbps).
- Power and Duty Cycle: An assumption of IP is that a device is always connected. This is not true for WSNs. The sensor nodes should go to sleep mode in order to save energy. Therefore,

the existing routing protocols have to be adapted to consider topology changes due to sleeping nodes in WSNs.

- Reliability: Today's Internet-based protocols achieve reliability with the Transmission Control Protocol (TCP). In general, TCP will not perform efficiently on WSNs since TCP is not able to differentiate between packets dropped because of congestion and packets lost on wireless links. In WSNs, most of the packets are lost due to link breaks. In this case, TCP unnecessarily reduces its transmission speed resulting in a poor performance. Therefore, lightweight reliable transmission protocols based on the unreliable User Datagram Protocol (UDP) have to be considered for WSNs also addressing the fact that unreliability occurs in WSNs due to a node failure and sleep duty cycles.
- Web Services: Internet services today rely on web-services, mainly using TCP, HTTP, SOAP and XML. These protocols are rather complex to be run on simple embedded devices.

Professional organizations which are working in the field of WSNs are coming from standardization bodies such as the Institute of Electrical and Electronics Engineers (IEEE), the Internet Engineering Task Force (IETF) and also from marketing alliances such as the ZigBee Alliance and the Internet Protocol for Smart Objects (IPSO) Alliance. The IETF Internet Protocol Version 6 over Low power Wireless Personal Area Networks (6LoWPAN) working group has made Internet Protocol Version 6 (IPv6) in WSNs possible by the Request for Comment (RFC) 4944.

In the *Intelligent Container* project, we have focused on using IETF IPv6 standards with required enhancements to fulfill specific logistical applications. The following sections discuss three main protocols, namely 6LoWPAN, Routing Protocol for Low power and lossy networks (RPL) and Constrained Application Protocol (CoAP) which are deployed in the WSN part of the *Intelligent Container*.

2. Open Standards for WSNs

The architecture of the *Intelligent Container* (Figure 1) consists of a WSN, a Freight Supervision Unit (FSU) and telematic devices. A WSN in the container transmits information such as humidity, temperature of meat, fruits etc. inside a container during land or sea transportation. This information is useful to take logistical process planning decisions to deliver food to markets efficiently and cost effectively. The FSU is the main computing device which is also attached to the edge router for the WSN and manages the different kinds of communication between WSNs and the backend software. Figure 1 shows the adapted protocol stack that is deployed in the sensor nodes, edge router FSU and backend software.

The IETF is standardizing networking solutions on top of link-layer protocols in the form of RFCs. As solutions involving IEEE 802.15.4 based devices are likely to be deployed in the hundreds or thousands of nodes per installation, the addressing space available for the nodes needs to be of adequate size. The addressing space of IPv4 is limited to less than 2^{32} addresses and is currently almost depleted due to the inefficient usage of the addressing space. Thus, IPv6 with its address space of 2^{128} addresses was chosen to be the networking protocol to be adapted to the underlying link layer protocols [2].

2.1. 6LoWPAN Header Compression

6LoWPAN standards enable the efficient use of IPv6 over low-power, low-rate wireless networks on simple embedded devices through an adaptation layer and the optimization of related protocols (i.e. Neighbor Discovery). The first 6LoWPAN specifications were released in 2007 as RFC 4944, specifying the 6LoWPAN format and functionality of header compression and neighbor discovery.

The most important challenges imposed by layer 2 (IEEE 802.15.4) are the small Maximum Transmission Unit (MTU) of 127 bytes while IPv6 requires support for 1280 bytes MTUs, from which the need for fragmentation support and header (including addresses) compression can be inferred. This is handled by the definition of an adaptation layer. RFC 4944 defines the packet format for transmission of IPv6 frames. Additionally, the creation of IPv6 link-local addresses as well as statelessly autoconfigured addresses for IEEE 802.15.4 networks are defined. A basic UDP/IPv6/802.15.4 header has a length of 58 bytes, of which UDP/IPv6 make up 48 bytes. However, those headers contain redundant information across the layers, especially the length information and addresses (IPv6 addresses can be derived from 802.15.4 addresses), which can be removed. The compression of commonly used values (flow labels, address prefixes, multicast addresses, UDP ports and checksum) can be compressed. Thus in the best case the UDP/IPv6 header can be compressed

to 6 bytes, according to [3]. In more detail, when using link-local unicast, the UDP/IPv6 header is 6 bytes. When using link-local multicast 7 bytes and when using global unicast the header has a length of 10 bytes.

The 6LoWPAN concept of an edge router makes sure the WSNs in the container integrate easily with the FSU and the backend software. This allows us to use any IP based protocols on the FSU and the backend software.

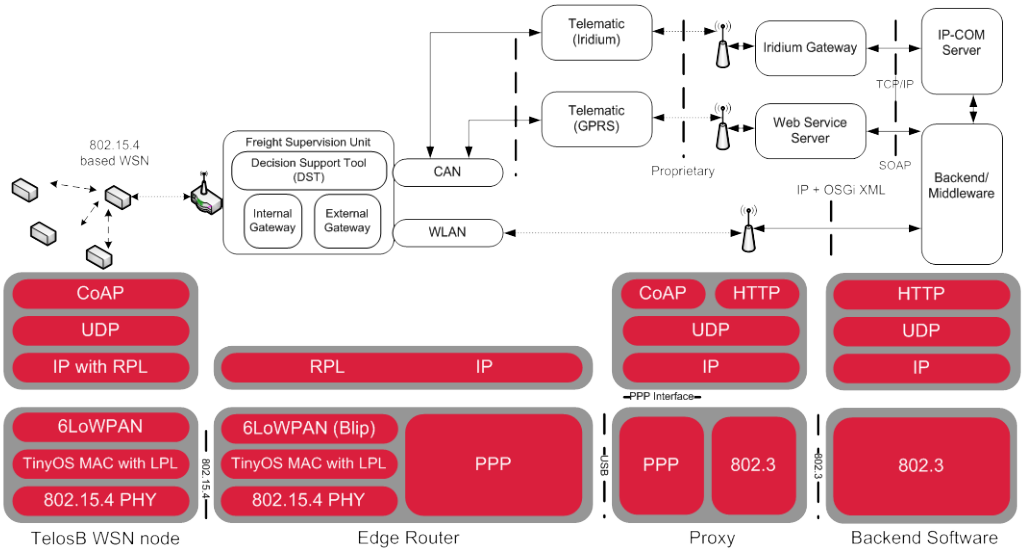

Figure 1: Overall Architecture with Open Standards in the Intelligent Container

2.2. Routing Protocol for Low power and lossy networks (RPL)

6LoWPAN allows for the forwarding and routing processes to be implemented at the link layer (termed 'Mesh Under') and at the network layer (termed 'Route Over'). We have used the Route Over concept, which does the forwarding and path computation at the layer 3, i.e. the IP layer above the 6LoWPAN adaptation layer. Every node in the WSN of the container acts as an IP router.

The specification of a routing protocol for WSNs is performed in the Routing Over Low power and Lossy networks (ROLL) working group. ROLL has specified a routing protocol for Low power and Lossy networks (LLN), termed IPv6 Routing Protocol for Low power and Lossy Networks (RPL), in the standardization documents from RFC 6550 until RFC 6554. RPL [4] relies on source routing by making use of the distance vector mechanism to calculate the routing paths. The RPL protocol is used to create a multi-hop network in the WSN part of the *Intelligent Container*. The most basic routing topology of RPL forms a Destination Oriented Directed Acyclic Graph (DODAG). A DODAG consists of only one root node and has no directed cycles. The 6LoWPAN edge router acts as the RPL root node. RPL uses two main objective functions, called OF0 [5] and MRHOF [6] for selecting the parent node and calculating the ranks.

OF0 is a basic objective function where nodes select their default route considering only the link quality to the possible parent together with its rank. Contrary, the path costs in MRHOF are obtained by summing up the link cost (e.g. in terms of expected number of transmissions (ETX)) to the possible parent and also considering the parents' link cost to the root node. This allows the use of MRHOF to select a parent considering the characteristics of a path rather than only considering an individual link quality. Both objective functions additionally use a hysteresis to limit ping-pong parent selection where the parent in the routing tree is switched back and forth.

RPL uses the Trickle algorithm [7] to regulate the sending of RPL control messages. If a node receives a control message and the information in the message differs from the receiver's state, an inconsistency is detected. In this case, the Trickle algorithm will send information more frequently. If the message agrees with the recipient's state, then there is no inconsistency in the network and the Trickle algorithm will slow down the transmission of control messages. With the Trickle algorithm, a node can detect inconsistency efficiently and, when there is no inconsistency, the control messages would be sent less frequently. It does not only save energy in sensor nodes, but also reduces traffic load in the network which leads to a better performance.

2.3. Constrained Application Protocol (CoAP)

The Constrained RESTful Environments (CoRE) working group of the IETF is working on the standardization of an application level protocol for constrained nodes and networks (such as WSNs) termed Constrained Application Protocol (CoAP), which is based on the Representational State Transfer (REST) [8] architecture. The current status of the CoAP protocol is specified in [9].

CoAP is a lightweight web protocol which is specifically designed to meet the requirements of constrained networks and nodes. Its main features compared to other web based protocols are low message overhead, simplicity and limiting the use of fragmentation. Furthermore, it provides reliable and asynchronous message exchange using UDP, Uniform Resource Identifiers (URIs) and content-type support as well as multicast support and built-in resource discovery.

CoAP is based on the same client/server model as HTTP and represents its interaction model in a similar manner. Resources are requested and identified by URIs using the REST methods GET, PUT, POST and DELETE, whereas response codes (e.g. 201 Created) indicate success or failure. In contradiction to HTTP, CoAP exchanges messages asynchronously over UDP. To provide a lightweight reliability mechanism, it makes use of an exponential back-off with the common stop-and-wait retransmission scheme. Furthermore, it implements detection of duplicate messages by using randomly generated, unique message identifiers.

CoAP is mainly used in the WSN part of the *Intelligent Container* to manipulate resources (e.g. temperature, humidity) by using the following methods:

- The **GET method** is used to retrieve resources from WSN nodes or the FSU. The resource is identified by the requested URI. For example, */r* resource is used to retrieve sensor data (temperature, humidity, etc) from the sensor nodes. Here, the sensor nodes act as CoAP servers.
- The **PUT method** is used to modify an existing resource on a sensor node or the telematic device. For example, */ni* resource is used to send configuration information to the FSU. A newly joined sensor node sends a *PUT* to the */ni* resource to inform the FSU about joining the WSN. Here, the sensor node acts as a CoAP client.

3. Deployment of Open Standards

3.1. Hardware and Software

We have used TelosB platform as shown in figure 2 to realize the WSN. The nodes are running the TinyOS operating system., which is an open-source operating system developed for WSNs with nodes of limited amount of flash memory and RAM. The operating system uses a component based structure and an event driven execution model. Some object oriented features are realized by utilizing a new language built upon C, which is called NesC. The TinyOS Simulator TOSSIM [10] has been used to evaluate the performance of 6LoWPAN, RPL and CoAP protocols with configured parameters used in the *Intelligent Container*.

Figure 2: TelosB Wireless Sensor Node

A 6LoWPAN implementation is available for TinyOS from the University of California Berkley called BLIP (Berkley IP Implementation) [11]. This includes a IPv6 protocol stack including 6LoWPAN header compression and neighbor discovery.

The RPL implementation available for TinyOS, called TinyRPL [12], is used to achieve multi-hop connectivity between sensor nodes. We have done several tests to investigate the optimum configurations of RPL to be used in the field tests. It was shown that RPL performs better with MRHOF objective function [13]. Further, we have observed that the batteries would have been discharged after approximately 7 days due to the ambient temperature of 13°C inside the container and the retrieval of

CoAP resources (e.g. */r* requests) every 5 minutes. Therefore, we have used an energy-conserving mechanism called LPL (Low Power Listening) to save power and to extend the lifetime. This LPL implementation uses the asynchronous duty cycling protocol BoX-MAC-2 [14] with a `LPL_SLEEP_INTERVAL` of 512 ms.

In the *Intelligent Container* project, the application installed on the nodes is called `CoapBlip`. It is based on the CoAP library `libcoap`, which is made available for the two major embedded operating systems Contiki and TinyOS. For TinyOS, we have adapted the `libcoap` library to match the typical TinyOS split-phase operation and programming style (e.g. wiring resources to the server) [16]. The initial evaluation of CoAP compared to HTTP based resource retrieval has shown the feasibility of using CoAP in an environment with constrained nodes and networks [17]. Table 1 and table 2 highlight the CoAP resources that are used on the sensor nodes and the FSU.

Table 1: CoAP Resources on Sensor Nodes

Resource	GET	PUT	Comments
/st	X		Temperature
/sh	X		Humidity
/sv	X		Voltage
/r	X		Temperature, Humidity and Voltage together
/l	X	X	LEDs
/ck	(X)	X	AES Encryption Key
/rt	X		Routing Table

Table 2: CoAP Resources on the FSU

Resource	GET	PUT	Comments
/ni		X	Inform about node integration into 6LoWPAN network
/ri		X	Inform about node's routing table
/warntemplow		X	Temperature is Below Warning Temperature Low
/warntemphi		X	Temperature is Above Warning Temperature High

3.2. Overhead Comparison of Standardized and Vendor-Specific Protocol

Telematics systems often use vendor specific protocols to interface to a backend system. To evaluate the usability of CoAP here, typical messages sizes were compared with a telematics-protocol named CBTP used at Cargobull Telematics. CBTP uses two basic message exchange schemes, i.e. *Live Data* and *Interval Data*. A *Live Data* request is initiated from the customer-side and represents a basic message exchange, where one response matches one request. An *Interval Data* request results in a subscription-based message exchange where the customer subscribes to a specific resource on a server and receives updates of the resource representation in specified intervals.

Table 3: CBTP and CoAP Message Size Comparison in Bytes

	Live Data		Interval Data			
	Request	Response	Registration		Data in Each Interval	
			Request	Response	Request	Response
Vendor specific telematics protocol (CBTP)	30	41	31	41	41	2
CoAP	7	31	30	34	34	6

For both, *Live Data* and *Interval Data*, CoAP can be used to support the same functionality as prevalent in CBTP. For this CBTP parameters are mapped to CoAP options, e.g. JobID Opcode → Uri-Path. The payload remains unchanged for the comparison. Table 3 shows the message sizes for both message exchanges and both protocols. It can be seen that the standardized protocol offers competitive overhead, while allowing for inter-working with the general Internet.

3.3. Interoperability with other OSs

A WSN used in logistic applications consists of different hardware and software components of different vendors. Therefore, the interoperability between different platforms is most the critical issue in this project. We have also shown the feasibility of using WSNs consisting of mixed networks of nodes with TinyOS and Contiki operating systems (latter is also used by project partner Virtenio).

4. Conclusion

The authors highlighted the use of open standards developed by the IETF in M2M enabled logistic applications. CoAP is used on the FSU and the sensor nodes in the container to retrieve resources in both directions. RPL is used to enable multi-hop connectivity in the sensor nodes embedded in food items. These protocols are designed to work with IPv6, enabling easy integration with back-end software. 6LoWPAN is used to compress IPv6 packets over 802.15.4 wireless technologies. These protocols are tested and evaluated in numerous field tests carried out in real logistic transportations with bananas and meat. We showed the use of open standards in M2M enabled logistic applications in the *Intelligent Container* project to provide multi-player and multi-vendor wireless acquisition of transport conditions.

5. Acknowledgments

The *Intelligent Container* project is supported by the Federal Ministry of Education and Research, Germany, under reference number 01IA10001.

6. References

[1] Z. Shelby and C. Bormann, 6LoWPAN: The Wireless Embedded Internet. Wiley Series on Communications Networking and Distributed Systems, 2009.

[2] M. Durvy, J. Abeill, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making sensor networks IPv6 ready," in Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008), Raleigh, North Carolina, USA., November 2008.

[3] J. W. Hui and D. E. Culler, "IP is dead, long live IP for wireless sensor networks," in Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys). New York, NY, USA: ACM, 2008, pp. 15–28.

[4] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[5] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552 (Proposed Standard), Mar. 2012. [Online]. Available: http: //www.ietf.org/rfc/rfc6552.txt

[6] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," RFC 6719 (Proposed Standard), Sep. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6719.txt

[7] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206 Proposed Standard), Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6206.txt

[8] R. Fielding, "Representational State Transfer (REST)," Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, 2000.

[9] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)," Internet Engineering Task Force, Dec. 2012, Available at: http://tools.ietf.org/html/draft-ietf-core-coap-13. [Online].

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems. ACM, 2003, pp. 126–137.

[11] S. Dawson-Haggerty, "Design, implementation, and evaluation of an embedded IPv6 stack." Master's thesis, 2010.

[12] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler, "ContikiRPL and TinyRPL: Happy Together," in Proceedings of 'Extending the Internet to Low power and Lossy Networks' (IP+SN 2011), Chicago, USA, Apr. 2011.

[13] T. Pötsch, K. Kuladinithi, P. Trenkamp, M. Becker, and C. Görg, "Performance Evaluation of CoAP using RPL and LPL in TinyOS," in Proceedings of New Technologies, Mobility and Security (NTMS), 2012 5th International Conference, 2012.

[14] David Moss and Philip Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking," Stanford Information Networks Group Technical Report, Available at: http://csl.stanford.edu/~pal/share/spots08.pdf, 2008.

[15] Z. Shelby, B. Frank, and D. Sturek, "Constrained Application Protocol (CoAP)," Internet-Draft (work in progress), Available at: http://tools.ietf.org/html/draft-ietf-core-coap-03, Jul. 2011. [Online].

[16] T. Pötsch and M. Becker, "TinyOS CoAP Installation Instructions," Available at: http://docs.tinyos.net/index.php/CoAP, Last accessed 2011-02-11. [Online].

[17] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker, and C. Görg, "Implementation of CoAP and its Application in Transport Logistics," in Proceedings of Extending the Internet to Low power and Lossy Networks (IP+SN 2011), 2011.