Middleware for Smart Gateways Connecting Sensornets to the Internet

Daniel Bimschas
Institute of Telematics, University of Lübeck, Germany
bimschas@itm.uni-luebeck.de

Horst Hellbrück
CoSA Research Group, University of Applied Sciences Lübeck, Germany
hellbrueck@fh-luebeck.de

Richard Mietz
Institute of Computer Engineering, University of Lübeck, Germany
mietz@iti.uni-luebeck.de

Dennis Pfisterer
Institute of Telematics, University of Lübeck, Germany
pfisterer@itm.uni-luebeck.de

Kay Römer
Institute of Computer Engineering, University of Lübeck, Germany
roemer@iti.uni-luebeck.de

Torsten Teubler
CoSA Research Group, University of Applied Sciences Lübeck, Germany
teubler@fh-luebeck.de

ABSTRACT

There is an increasing trend to integrate sensor networks into the Internet, eventually resulting in an Internet of Things. Recent efforts of porting IPv6 to sensor networks turn sensor nodes into equitable Internet peers and RESTful Web Services on sensor nodes allow a distribution of the application logic among sensor nodes and more powerful Internet nodes. The touching point between a sensor network and the Internet is the gateway which translates between the link-layer protocols used in the Internet (Ethernet, Wi-Fi) and sensor networks (IEEE 802.15.4). So far, the functionality of those gateways was fixed and simple. We propose to turn these gateways into smart gateways by enabling them to execute application code. As only the gateway has full knowledge of and control over *both* the sensor network and the Internet, smart gateways can act as performance-enhancing proxies and intelligent caches to preserve the limited resources of the sensor network. Also, the smart gateway can perform application-specific protocol conversion between highly optimized but non-standard protocols in the sensor network and standardized, but less efficient protocols in the Internet. In this paper we present the design of a middleware for smart gateways that allows the execution of application code on the gateway by offering simplified interfaces to the sensor network and the Internet. We also report preliminary performance results for key functions of the middleware.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process, Information filtering*; C.2.4 [**Computer-communication Networks**]: Distributed Systems—*Distributed applications*; C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems

General Terms

Design, Performance, Reliability

Keywords

Wireless Sensor Networks, Future Internet, Integration, Search

1. INTRODUCTION

There is an increasing trend to integrate sensor networks with the Internet. The term Internet of Things has been coined to express the vision of augmenting real-world objects, places, and people with sensors, actuators, and microcontrollers and to connect them to the Internet via wireless communication. Essentially, the Internet of Things offers online access to and control over the state of the real world. One indicator for this ongoing integration of sensor networks and the Internet is the recent trend of porting IPv6 to sensor networks by means of the 6LoWPAN [10] adaptation layer and by using routing protocols such as RPL [14] that have been specifically optimized for resource-constrained and lossy networks. The touching point between a sensor network and the Internet is a so-called gateway: a computer with two network interfaces that translates between link layer protocols used in the Internet such as Ethernet or Wi-Fi on the one hand, and link layer protocols used in sensor networks such as 802.15.4 on the other hand.

Applications in the Internet of Things are typically distributed over sensor nodes (e.g., data collection, preprocessing, and aggregation) and more powerful Internet nodes (e.g., complex data analysis and decision making). Hence, application-level protocols are also needed that span sensor networks and the Internet to integrate the distributed application logic. A recent trend here is the use of lightweight Web Services based on the REST principle [7] on top of HTTP. The term REST stands for *Representational State*

Transfer. *RESTful* architectures use the central abstraction of a *resource* (e.g. aggregated sensor data or a sensor on a sensor node). Resources can have arbitrary representations, e.g. XML, JSON or an application specific binary representation. Every resource is addressed by a URI and can be manipulated (i.e. have its state transferred) using the well-known HTTP request methods such as GET, POST, PUT, DELETE to retrieve, update, create or delete it. Sensor nodes then would directly support these light-weight Web Service protocols, thereby allowing the application to easily span over the sensor network and the Internet.

We propose to extend this model by allowing the execution of application-specific code on the *gateway*, such that parts of the application logic execute on sensor nodes, parts on the gateway, and parts on powerful Internet nodes. This proposal is in contrast to the current understanding of a gateway, which is usually responsible for a fixed set of jobs such as to monitor traffic, firewalling, access control and security applications.

This proposal is motivated by the fact that the gateway is in the unique position to participate both in the sensor network and in the Internet, that is, application logic executing on the gateway can exploit knowledge of and information collected from those two networks. Application logic executing on sensor nodes or on Internet nodes does not have access to this combined information.

More concretely, the application code on the gateway can contribute to saving resources in the sensor network. One way to achieve this is by means of protocol conversion. Instead of supporting costly HTTP and REST on sensor nodes, a more efficient protocol such as the Constrained Application Protocol (CoAP) [11] could be used in the sensor network and the gateway would translate between CoAP on the sensor network side and HTTP+REST on the Internet side. Protocols such as CoAP are currently not yet fully standardized and drafts change frequently. Being able to deploy application-specific code on the gateway, one can easily change the protocol in the sensor network without affecting the interface offered to the Internet. Further, the gateway can act as a performance-enhancing proxy (PEP) by answering application requests to the sensor network without communication with the sensor network. By overhearing broadcast messages in the sensor network, the PEP could learn the state of the sensor network in an application-specific way and answer requests from Internet nodes without additional communication with sensor nodes. Finally, the combined information available at the gateway is often helpful in debugging.

To facilitate the development of application code for the gateway, we develop a middleware that offers simplified interfaces to the two networks connecting to the gateway. The architecture of this middleware and its individual components are described in the subsequent section. After that follows a preliminary evaluation of key functions of the middleware.

2. DESIGN

Figure 1 depicts the network architecture and the composition of the smart gateway middleware. Let us first focus on the network architecture. Sensor networks connect through a smart gateway to the Internet and there is one smart gateway for each sensor network. However, besides those wireless sensors, many sensors such as Webcams are directly connected to Internet nodes already today. Our goal is to offer a consistent interface to these two classes of sensors. Hence, we provide so-called *virtual sensor nodes* that mimic real sensor nodes to which individual sensors are attached. Those virtual sensor nodes are "connected" to the Internet through a smart gateway that is very similar to the smart gateway connecting real sensor nodes. Both smart gateways offer the same interface to application code executing on the gateway.

Let us now focus on the software architecture of the middleware running on the smart gateways. At the very base of the middleware is a Layer-2 gateway interconnecting IPv6/6LoWPAN-based sensor networks and the IPv6-based Internet, which we call *Forward-Sensor*. We have implemented this Layer-2 gateway as a virtual network card running on standard-PCs using the *Virtual Tunnel* [9] framework that provides so-called TUN (network-layer) and TAP (Ethernet-layer) interfaces, which give user-space access to received IP-packets and Ethernet frames and allow sending fabricated ones. On top of this virtual network card sits a user-space TCP/IP (version 4 and 6) communication stack called EZnet [13] which is implemented in Java. The stack is configurable in the sense that different protocol layers can be selected and combined. EZnet parses incoming packets on every protocol layer, passing it as structured object to the application, such that the application logic has full access to information from and full control over all networking layers. Finally, EZnet interfaces to the application code executing on the smart gateway through a well-defined application interface.

The basic operation of this application interface is as follows. All incoming packets (either from the sensor network or the Internet) are parsed by EZnet providing details from the individual protocols and passed to the application code. The application code can then inspect the packet and its headers and meta data and decide what to do. There are basically four actions possible: *discarding* the packet – possibly after extracting and storing some of the information from the packet; *forwarding* the packet unchanged or with changed headers to the original or a different destination; *translating* the packet into a different protocol and then sending the translated packet; or *replying* to the sender of the packet, e.g., using cached information. The application logic may also generate and send a message without an incoming message, for example, to maintain cached information up-to-date. In the remainder of this section we describe the individual components of our architecture.

2.1 Layer-2: Forward-Sensor

The primary task of the Forward-Sensor component (see Figure 1) is to interconnect the Internet and the Wireless Sensor Network. Therefore it works as a protocol translator between Ethernet and vendor specific wire formats, depending on the sensor node hardware in use. A typical use-case is that a gateway sensor node is connected to a PC via USB. This layer-2 functionality is implemented as a virtual network card running on standard-PCs using the *Virtual Tun-*

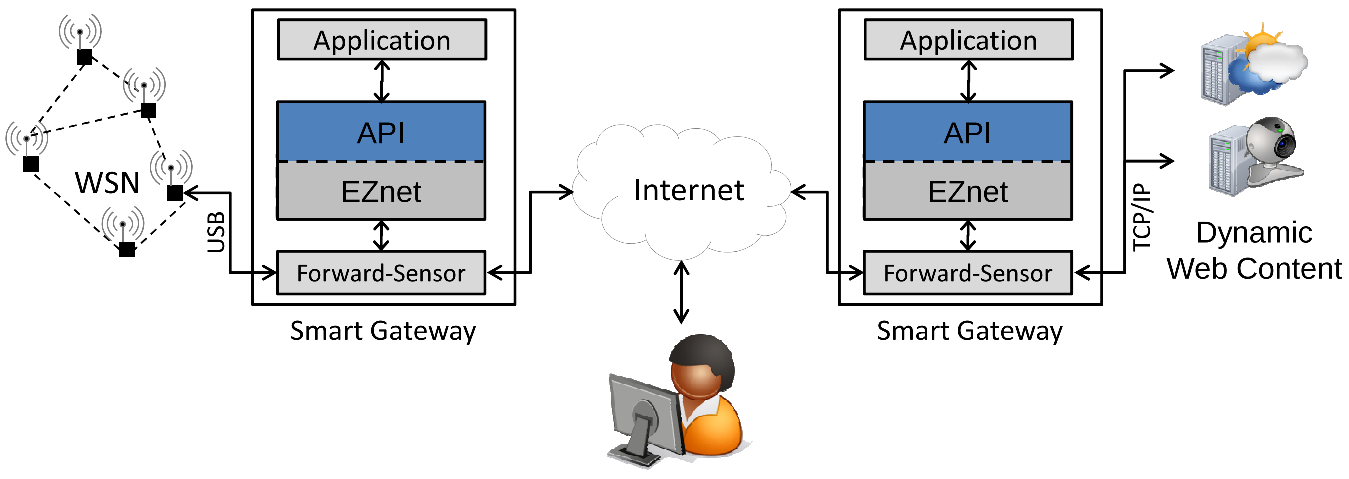

Figure 1: Overview of the proposed architecture

nel framework that provides so-called TUN (network-layer) and TAP (Ethernet-layer) interfaces. In addition to this well-known protocol translation functionality we've built extra functionality that allows us to intercept incoming/outgoing Ethernet frames and IP-packets from and inject fabricated ones into the communication stream between sensor network and the Internet through a standardized interface.

The Forward-Sensor component therefore interacts with the sensor node and creates a virtual network card on a standard PC to receive IPv4/IPv6 packets destined for the sensor network. It passes both the data received from the sensor network and that received on the virtual interface to an arbitrary application (such as our EZnet Stack). In addition, it accepts data from such an application and transmits is either to the attached sensor node or passes it to the operating system for routing towards the Internet.

Applications running on top may perform arbitrarily complex tasks ranging from simple traffic filtering (i.e. firewall functionality), to application-level protocol translation or proxy functionality. Applications receive all traffic as is, i.e. including frames of all layers up from layer 2 of the networking stack, therefore enabling deep packet inspection or cross-layer protocol translation. In order to support the development of those kinds of applications, we have extended the *EZnet* protocol stack which runs on top of the Forward-Sensor component.

2.2 Layer-3 upwards: IPv6 EZnet Stack

Once individual frames are passed from the virtual network interface mentioned above, further processing is required to adapt protocols for use in WSNs. Examples are the translation from IPv6 to 6LoWPAN, HTTP over TCP to HTTP over UDP, and HTTP over TCP to CoAP. A smart gateway performing this translation must therefore be capable of interacting with peers in the Internet on the same protocol layer. For instance, in the case of a translation from HTTP over TCP to CoAP, the gateway must act on behalf of the sensor node and it must handle the TCP/IP connection setup, data transmission, and teardown (i.e., accept TCP/IP connections for the sensor node) as well as the HTTP request and response cycle. On the other hand, the gateway communicates with the sensor nodes via CoAP. Apart from being the communication endpoint for the WSN and the Internet hosts, it must translate from one protocol to the other also with respect to the possibly different protocol semantics.

An implication of this is that we cannot use the standard protocol implementations available in different operating systems but that some custom implementation is required. For our implementation of the smart gateway, we use a user-space Java implementation of TCP/IPv4 called EZnet, which was proposed by Walther et al. [13]. We have extended EZnet to support IPv6, TCPv6, and ICMPv6 and to interact with our Forward-Sensor component. The user-space implementation simplifies development in various ways. It allows convenient debugging and exception handling and provides fast analysis of errors where a kernel implementation in C/C++ just halts or crashes the kernel on errors. In addition, any available Java library can be used in conjunction with EZnet (e.g., HTTP proxies, RESTful application servers, etc.).

2.3 Virtual Sensor Nodes

Deploying sensor networks to build up the Internet of Things is a costly and therefore long process. To jump-start the Internet of Things, we should therefore also exploit other types of sensors that are already deployed. For example, many webcams and weather stations are connected to Internet nodes and can be accessed through the Web. Often, not the raw "sensor data" (e.g., image from a Webcam) are interesting, but derived state such as light intensity or number of people on an image.

However, access to such dynamic web content is quite different from access to sensors attached to real sensor nodes. To offer a homogeneous interface to all types of sensors, we offer so-called *virtual sensor nodes* that mimic real sensor nodes to which different sensors can be attached. Essentially, a virtual sensor node is a program executing on an Internet node reading data from Internet sensors such as a Webcam.

A virtual sensor node could for example poll a Web page containing an image from a Webcam at regular intervals, parse the page to extract the raw image, and process the data to obtain the desired sensor output such as the light intensity. We provide a software framework that allows to poll Web pages at configurable intervals and to apply a chain of filters to the downloaded page to parse it and to process the raw data in order to obtain a virtual sensor reading. The details of this framework go beyond the scope of this paper.

Even though virtual sensor nodes are – in contrast to real sensor nodes – hosted on Internet nodes with plentiful resources and use standard Internet protocols, there are cases where the functionality of a smart gateway would also be helpful for virtual sensor nodes. Therefore, we can optionally introduce a smart gateway also between virtual sensor nodes and the rest of the Internet as depicted in Figure 1. Both, the smart gateway connecting the virtual sensors and the one for the real sensors, are executing the same middleware. In particular, the API of the middleware is the same such that the same application code can execute on all smart gateways. From the perspective of the application code, virtual sensor nodes are indistinguishable from real sensor nodes.

3. APPLICATIONS

In this section we outline three possible applications of our smart gateway: i) protocol conversion, ii) request caching, and iii) intelligent caching and discovery.

3.1 Protocol Conversion

When observing the current standardization tracks for WSNs, a clear trend is that the integration of wireless sensor nodes and other networks into the Future Internet will use (or at least resemble) existing, widely deployed technologies such as TCP/IP and HTTP. Based on these technologies, the RESTful architecture style [7] provides an excellent abstraction of the service paradigm and only uses TCP/IP and HTTP to implement distributed applications. RESTful HTTP-based Web Services have been broadly adopted in the Internet recently. Compared to other service abstractions such as SOAP-based Web Services the HTTP-based RESTful Web Service architecture style is relatively conservative regarding resource consumption in terms of energy and processing power.

The term REST stands for *Representational State Transfer*. The central abstractions in a RESTful architecture are resources. A resource is addressed by a URI and it can be manipulated (i.e. transfer its state) using the request methods of the HTTP protocol. The most common operations are i) GET to read a resource representation, ii) POST to update it or create a new resource, iii) PUT to update an existing resource, and iv) DELETE to delete an existing resource. Furthermore, a resource can have different representations, e.g. XML, JSON, or some efficient binary format for machine processing or an HTML representation for humans. Relations between resources are expressed by simple links. Applying the RESTful architecture pattern to WSNs, every node, every sensor on the node, and every service provided by the node are addressed as a (read-only) REST resource.

Using RESTful Web Services that require HTTP and TCP/IP inside WSNs however is still suboptimal because of the resource demands and the connection-oriented nature of TCP. For both technologies, adaptation layers have been proposed and are being standardized currently. For instance, instead of native IPv6, 6LoWPAN can be used inside WSNs and instead of HTTP the Constrained Application Protocol (CoAP, [11]) is a viable option. CoAP adheres to the RESTful architecture style while catering for the resource constraints of the WSN.

However, the fundamental nature of WSNs and the Internet is vastly different and inside the WSN, strict resource constraints apply. Hence, exchanging data with an IPv6/6LoWPAN-enabled sensor node requires that the application payload is as small as possible to be forwarded, stored, and processed on sensor nodes. Hence, a client on the Internet would be forced to use CoAP to interact with sensor nodes. This is a loss of transparency since different classes of devices require a different way of interaction. To alleviate this situation, our smart gateway approach could transparently perform this adaptation and expose the services of CoAP-enabled sensor nodes as standard HTTP-based RESTful Web Services towards the Internet. We are currently implementing and evaluating protocol translation functionality, using the EZnet protocol stack that will map the protocol semantics between TCP/HTTP and CoAP. By that, the services provided by the sensor nodes inside the WSN will be transparently available to every HTTP Client, including Web browsers.

3.2 Request Caching

Caching aims at reducing response time and resource consumption of the sensor network by means of the smart gateway answering a request to the sensor network without actually communicating with the sensor network by exploiting the results of previous requests to the sensor network.

In the simplest mode of operation an application caches requests and responses. When new requests arrive, it answers with cached responses on behalf of real sensor nodes if the cached information is still valid. An optimization is to either request data from the sensor nodes once in a while or to instruct nodes to send data periodically to update the cache.

Such a cache application can already improve the integration of sensor networks in several ways:

- **Reduce Response Time**: As an agent for sensor nodes the cache application answers requests immediately instead of forwarding messages to the low-bandwidth/high-delay sensor network.
- **Reduce Load/Traffic in the Sensor Network**: The more messages answered directly by caches the more the traffic in the sensor network is reduced.
- **Increase Life Time of the Sensor Network**: Less messages in the sensor network means less energy consumption leading to improved lifetime of the sensor nodes and the whole network.

A drawback of simple request caching mechanisms that, e.g.

use time-based cache invalidation, is their potential lack of accuracy regarding the actual value and the responded value. To alleviate this drawback, we investigate the use of *prediction models*, which are described in the next subsection.

3.3 Intelligent Caching and Discovery

Instead of simply repeating the result from the last request, a more effective and accurate approach is to *predict* current sensor values from past values of the same sensor or from current values of other correlated sensors. The smart proxy can collect such information by overhearing all reply messages from the sensor network.

Here, we are interested in a special type of prediction where not only the current sensor value is predicted, but the *probability distribution* of the current sensor output is estimated. For example, for a binary occupancy sensor the output of the prediction model could be that the current value is "occupied" with a probability of 0.8 and is "free" with a probability of 0.2.

The reason for this approach is that it does not only allow to predict the current value of a sensor (e.g., by using the value with the highest probability), but it enables an efficient form of *sensor discovery*, which is the problem of finding sensors that currently output a certain value. Knowing the probability distributions, we can identify sensors that currently output a sought value with high probability. By contacting sensors in the order of decreasing probability to find out their actual current state, we spend communication overhead primarily for sensors that are very likely to match.

We are currently exploring the use of correlations between sensors for prediction. From past observations of a pair of sensors A and B the smart gateway can compute a correlation function that allows to estimate the current value of B given the current value of A and vice versa. We are modeling the correlations using a linear Bayesian Network (BN), where each node of the BN represents a sensor with its possible output values and their probabilities. Whenever the actual value of a sensor becomes known, this information is entered into the BN and propagated to the whole network, updating probabilities for each possible output value at every node.

4. EVALUATION

In this section we will present a first evaluation of our new architecture and a technique for the *intelligent caching module*.

4.1 Request Caching

We evaluated the performance of our smart gateway with a simple caching application running on top thus acting like the performance enhancing proxy (PEP) mentioned in the introduction. The setup, including the IPv6 addresses used, is depicted in Figure 2. An IPv6-enabled browser issues HTTP requests to a sensor node running Contiki OS with IPv6 support and a web server [4, 5] via a smart gateway. The application caches HTTP requests and the corresponding HTTP responses that are tagged with expiration date, so the cache can answer future requests with the cached information until it expires.

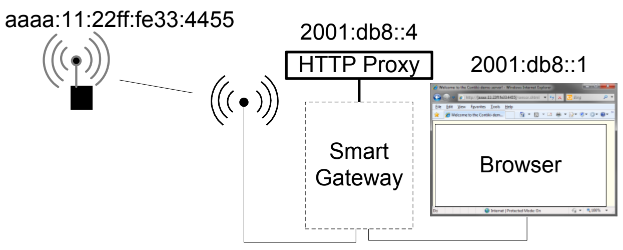

Figure 2: The Test Setup for Performance Evaluation.

We measured the time needed to retrieve different amounts of payload from the sensor node i) with and ii) without the gateway in between and additionally, we iii) measure the time for a response when the HTTP proxy sends the cached information without interacting with the sensor network. In this setup we take into account the best case where the information is requested from a sensor node within the one-hop range of the gateway. However, in a large-scale WSN there will be multi-hop connections to the desired node resulting in even longer response times and higher amount of messages to be sent.

The results can be seen in Figure 3. The direct request from the sensor node required less than $2sec$ in average. If we add the gateway in between, the request time goes up to $5sec$ and even more for larger payloads. It seems that is due to a problem between the used network device driver or firmware for the wireless connection and libpcap which EZnet is connected to. We expect that if this issue is resolved, the time is approximately the same as before. If a cached result is used, the time required is less than $0.25sec$ showing that the parsing performance of our smart gateway is good.

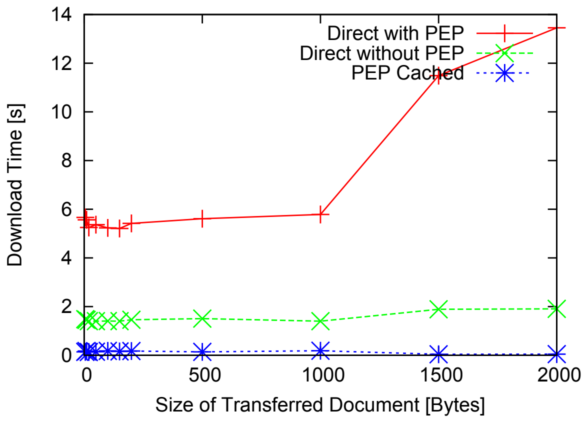

Figure 3: PEP Time Measurement

4.2 Intelligent Discovery

In addition to simple caching, we investigate the effectiveness of intelligent discovery using a prediction model that exploits correlations between the outputs of sensors as described in Section 3.3.

As a data set we are using traces from sensors in the stations of the Bicing [3] bicycle rental system in Barcelona. Each of the 384 sensors outputs the current number of available bicycles at each of the rental stations over a period of three months. The values of those sensors are published in real-time on a Web page. Using virtual sensor nodes, sensors can be integrated into an Internet of Things.

For the evaluation we sought to discover 10 sensors currently reading a certain number of available bicycles. Using the prediction model, the smart gateway picks the sensor with the highest probability of currently reading the sought value. A message is sent to that sensor to find out its actual current value. This actual value is then entered into the BN and propagated through the network to update the predicted probabilities. Then again the not-yet-contacted sensor with the highest probability is sent a message to request its value and the BN is updated and so on until 10 matching sensors have been found or all sensors have been contacted.

We repeat this discovery at different times and compute the average number of sensors that need to be contacted to find 10 matches. We compared this with a naive approach, where the smart gateway would contact sensors in random order until enough matches have been found. The results are shown in Figure 4. On the left hand side, the results of the individual discovery operations are shown, the right-hand side shows box plots with median, first and third quartiles and min/max values. We can clearly see that the prediction model is effective as it approximately halves the number of messages and also has a lower variance.

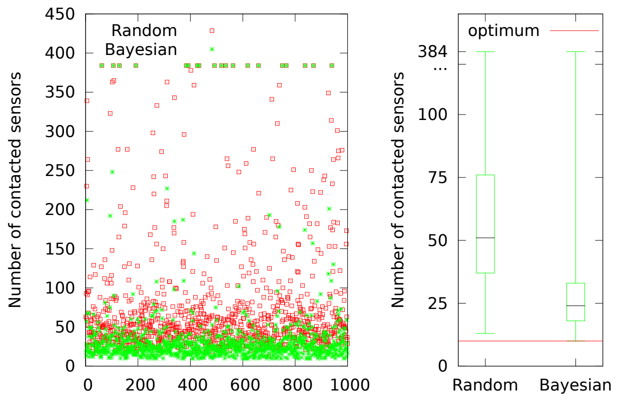

Figure 4: Number of sensor contacted when using *Random* and *Bayesian* search.

5. RELATED WORK

The overall goal of our research is the seamless integration of (wireless) sensor networks into the Future Internet. By that, we explicitly also focus on higher level integration beyond MAC- or IP-based connectivity.

There has been a lot of research regarding IP connectivity for resource constrained devices that lead to standardization of protocols such RFC 4944 [10], better known as 6LoWPAN, that made the utilization of IPv6 in sensor networks feasible. [2] gives a good overview on research in this field in the last years.

Looking at the question of how to integrate the actual data delivered by WSNs into applications in the Future Internet it is easy to argument that a service abstraction, built on standardized protocols, can help managing the complexities and constraints of the scenario. There has been a lot research recently on how to make the currently most popular service middlewares feasible for usage in WSN.

Dogan Yazar and Adam Dunkels try to employ standard compliant RESTful HTTP Web Services inside the WSN [15]. For improved performance of the underlying TCP connection they adapt the duty-cycling of the MAC-layer protocol to adhere to the session and reliable transport semantics to stay awake for the whole TCP Session. Furthermore, they use the *Conditional HTTP GET mechanism* to transport only response payload if the ressource has changed since the last request to improve energy efficiency. Glombitza et al. employ SOAP-based Web Services inside the WSN by using the Lean Transport Protocol [8], an efficient Web Service transport protocol explicitly targeting resource constrained devices. The Internet Engineering Task Force (IETF) has recently started a new working group by the name of *Constrained RESTful environments* (CoRE) which is currently working on the *Constrained Application Protocol* (CoAP) [11] that can be used as a middleware to implement RESTful Web Services, based on a compressed subset of HTTP methods and headers, thereby further optimizing payload size. CoAP also tries to implement transport guarantees to make the use of TCP obsolete inside the WSN. In comparison to the aforementioned approaches we try to exploit unused potentials by optimizing the glue line (e.g. the gateway) between the WSN and the Internet. Our goal is to further enhance the overall system efficiency by bridging the gap between the partially contradictory resource requirements and constraints of WSNs and the Internet. Please note that our approach is easily transferable to other middlewares such as SOAP-based Web Services.

The SENSEI EU-FP7 project [12] proposes an architecture that represents sensor nodes as semantic annotated resources that provide services. SENSEI provides means to search and to semantically query for sensor nodes and their data. It makes no assumption about how the services of the sensor nodes are actually provided and is by that compatible with various solutions, such as our RESTful service oriented approach that is transparently performance enhanced.

Sensor Ranking [6] sorts sensors according to the probability that they fulfill a given query. By examining historic data of the sensors for periodicity of measured values, prediction models are generated which try to foresee the currently measured value. The sensors are then contacted in order of decreasing probability. This can eventually reduce the total number of messages needed to find enough sensors that fulfill the given query. Our approach additionally tries to take potential correlations between sensor measurements into account. This allows us to use existing knowledge about sensor output of a node A at a given point in time to improve prediction of the sensor output of a node B if they are correlated.

Global Sensor Networks (GSN) [1] is a middleware that abstracts from the underlying, heterogeneous sensor network technologies, provides efficient distributed query processing and combination of sensor data and dynamic deployment of new sensor nodes and networks. The middleware is data-oriented, providing means to gather, analyze and publish data from sensor networks. Sensor nodes are connected to the GSN via a GSN-specific wrapper. In contrast to GSN our focus lies more on deploying non-proprietary and Internet standards-based middleware such as RESTful Web Services in an efficient manner. This will eventually allow us to use existing data acquisition and mining software built upon standard Internet protocols, alleviating the need for custom-tailored solutions that explicitly target sensor networks. Please also note that by implementing a RESTful HTTP Web Services wrapper for GSN, networks based on our architecture could easily join the GSN network.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced our smart gateway architecture for an integration of WSNs into the Internet. The key idea is to turn the formerly simple protocol translators to smart gateways. These exploit their full knowledge of *both* the sensor network and the Internet to preserve the limited resources of the sensor network. We presented three different applications running on top of our smart gateway: protocol translation, request caching, and intelligent caching with sensor discovery. We have evaluated our approach and presented first promising evaluation results showing that smart gateways can speed up response times, reduce the number of messages sent inside the WSN, and thus prolong the lifetime of the WSN.

Future work will include more applications on top, performance optimizations, in-depth evaluations, and integration into our outdoor testbed. Furthermore, we will create new and improve existing prediction techniques for the prediction module. In the end we will build a search engine with different testbeds and virtual sensors connected to proof that our approach is feasible.

7. ACKNOWLEDGMENTS

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01BK0905, GLab). The authors alone are responsible for the content of the paper.

8. REFERENCES

[1] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In *Proceedings of the Mobile Data Management (MDM 2007)*, Mannheim, Germany, 2007. IEEE Computer Society.

[2] P. A. C. da Silva Neves and J. J. P. C. Rodrigues. Internet protocol over wireless sensor networks, from myth to reality. *Journal of Communications*, 5(3), 2010.

[3] B. de Serveis Municipals. Bicing, 2009. http://www.bicing.cat/.

[4] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, NOV 2004.

[5] M. Durvy, J. Abeillé, 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), poster session*, Raleigh, North Carolina, USA, NOV 2008. Best poster award.

[6] B. M. Elahi, K. Romer, B. Ostermaier, M. Fahrmair, and W. Kellerer. Sensor ranking: A primitive for efficient content-based sensor search. In *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 217–228, Washington, DC, USA, 2009. IEEE Computer Society.

[7] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[8] N. Glombitza, D. Pfisterer, and S. Fischer. Ltp: An efficient web service transport protocol for resource constrained devices. In *Seventh Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (IEEE SECON' 10)*, 2010.

[9] M. Krasnyansky. VTUN – Virtual Tunnel, 2010. http://vtun.sourceforge.net/.

[10] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007.

[11] Z. Shelby, B. Frank, and D. Sturek. Constrained application protocol (coap). An online version is available at `http://www.ietf.org/id/draft-ietf-core-coap-01.txt` (08.07.2010), jul 2010. Expires: January 9, 2011.

[12] The SENSEI project. The SENSEI real world internet architecture. An online version is available at `http://www.sensei-project.eu/index.php?option=com_docman&task=doc_download&gid=83&Itemid=49` (13.08.2010).

[13] U. Walther and S. Fischer. EZnet: A Framework for Rapid Protocol Protyping. In *IEEE International Conference on Networking (ICN 2002), Atlanta, USA*, aug 2002.

[14] T. Winter, P. Thubert, and the ROLL Team. Rpl: Ipv6 routing protocol for low power and lossy networks. An online version is available at `http://tools.ietf.org/html/draft-ietf-roll-rpl-11` (08.07.2010), jul 2010.

[15] D. Yazar and A. Dunkels. Efficient Application Integration in IP-based Sensor Networks. In *Proceedings of ACM BuildSys 2009, the First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings*, Berkeley, CA, USA, NOV 2009.