A Comprehensive Approach to Integrating Sensor Networks and Enterprise IT

NILS GLOMBITZA, DENNIS PFISTERER, STEFAN FISCHER
Institute of Telematics, University of Lübeck, Germany
{glombitza, pfisterer, fischer}@itm.uni-luebeck.de

It is envisioned that in the future, all kinds of devices ranging from resource-constraint wireless sensor nodes to powerful server-class computers will interact to form an Internet of Things (IoT). In such a setting, tiny devices will extend the Internet to the physical world and allow for a completely new class of applications. However, until today, no widespread deployment of such IoT applications can be observed. Major challenges to master issues of embedded programming, massive distribution, resource constraints, heterogeneity, and seamless integration with traditional Internet technologies. Orchestrating such a number of different devices to form an application can be arbitrarily complex. In the Internet and especially in Enterprise IT, the concept of Service-Oriented Architectures (SOA) has been applied successfully to address this orchestration problem. However, the technologies used to realize SOAs (such as Web Services, HTTP, XML, or BPEL) are to heavyweight to be used in resource-constraint networks. In this paper, we present a comprehensive, Web Service-compliant approach to use SOA technologies in WSNs. Our approach comprises self-description of sensor nodes, a light-weight Web Service transport protocol (called Lean Transport Protocol, LTP), an approach for the model-based programming of sensor nodes, and a solution for the integration with the Internet. We present an exhaustive simulation showing the first-rate performance of the approach.

Keywords: Design, Experimentation, Measurement, Performance, Reliability

1. INTRODUCTION

In logistics, countless companies are concerned with the storage and shipment of goods. For companies, it has become a central concern to track the whereabouts, condition, and proper handling of these goods. Such information is typically stored in enterprise information systems and keeping this information up to date is a tedious, error-prone task which today is often performed manually. By attaching wireless sensor nodes to or embedding such nodes in the goods themselves, data can be gathered and forwarded automatically to enterprises' IT systems. Beyond such sense-and-forward scenarios, novel services and applications could be realized and offered to customers.

Imagine a logistics setting where a company is concerned with storage and shipment of trailers containing temperature-sensitive goods. Here, trailers enter a company's premises, are stored at a certain location, and leave the premises again for onward transportation. The company must ensure that the temperature level of the goods remains below a certain critical level. Sensors are attached to trailers and/or goods and measure and detect environmental properties (such as location, temperature and goods stored nearby that periodically broadcast their nature). In case of an emergency, i.e. the temperature rises beyond a threshold or a forbidden cargo constellation has been detected, the corresponding forwarding agency or the police are informed. Often, such sensors are organized in larger structures, so-called wireless sensor networks (WSNs).

A fundamental challenge is the integration of WSNs into enterprise IT systems, mainly due to a large gap between embedded software and the software environments typically used in backend IT systems. While the latter increasingly use Service Oriented Architectures (SOA) based on Web Services, wireless sensors nodes are often programmed using low-level embedded languages and accessed using proprietary protocols. The main reasons for this are severe resource constraints of wireless sensor nodes. The desire for long lifetimes paired with constrained energy supply from batteries implies that computing power, memory capacity, and networking bandwidth are several orders of magnitude smaller than those found in backend IT systems – necessitating optimized

software and networking solutions.

In this paper, we present a solution for the seamless integration of wireless sensor nodes into backend IT systems. In contrast to related work, our approach is fully compliant to Web Service standards and therefore rightly consorts with widely deployed enterprise IT systems. In the following section we present a possible use case and the overall architecture of our approach. Following, the individual building blocks are described in detail and the paper concludes with an exhaustive evaluation demonstrating the applicability of our approach in real-world settings.

2. OVERALL ARCHITECTURE

Application development in WSNs could greatly benefit from using a SOA-based approach for a number of reasons: the integration with Internet-based applications is greatly simplified, the same programming methodology and accompanying tools are used, no low-level embedded programming skills are required, applications can quickly be composed out of existing functionality, and the time-to-market is significantly reduced.

However, the severe resource constraints and the embedded nature make it mostly impossible to use SOA technologies (such as Web Services, WSDL[1], HTTP, XML, and BPEL[2]) in WSNs. As a result, proprietary approaches have been proposed that offer similar functionality in WSNs. The problem with these is that they do not comply with standards used in the Internet. What is missing to support standard-compliant SOAs in WSNs is

(1) an efficient representation of WSDL files on the sensor node devices themselves in order to dynamically integrate the offered services into applications,
(2) a compact encoding and transport protocol for Web Service (i.e., SOAP) messages in order to allow for communication over ressource-constraint networks,
(3) support for business process execution (i.e., BPEL) inside a WSN, and
(4) a mechanism allowing programming-in-the-large (to avoid low-level sensor node programming)

Our approach differs from related work in that it is fully standard-compliant and blends particularly well with the Web Service technology stack. We use the standard extension mechanisms of this stack and provide alternatives for some of the standard's defaults. We even ensure by a special, adaptive proxy component that software which only supports the defaults interoperates with our scheme. Our extensions to the Web Service technology stack are shown in Figure 1 where they are highlighted in dark gray.

Our contribution is a comprehensive set of protocols and systems that cover exactly the missing building blocks mentioned above which are discussed in detail in the remainder of this paper. First, Section 3 discusses our approach for the discovery and self-description of sensor nodes. Then Section 4 introduces LTP (Lean Transport Protocol), our novel transport binding for Web Services in resource-constraint environments. Building on top of this transport binding, Section 5 presents our approach for modeling business processes in WSNs without programming individual sensor nodes. Following, Section 6 shows how the automatic conversion of messages and the web service transport protocol works. Section 7 contains an exhaustive evaluation and Section 8 concludes the paper. Related work is discussed for each partial solution in the corresponding section.

3. DISCOVERY AND DESCRIPTION

The introductory scenario motivates that such an environment is not static but shows that mobility and ad-hoc interactions, where sensor nodes cooperate with different backend systems over time, are intrinsic properties. To support such ad-hoc interactions, a description of Web

[1]WSDL: Web Services Description Language [Christensen et al. 2001]
[2]BPEL: Business Process Execution Language [OASIS WS-BPEL Technical Committee 2005]

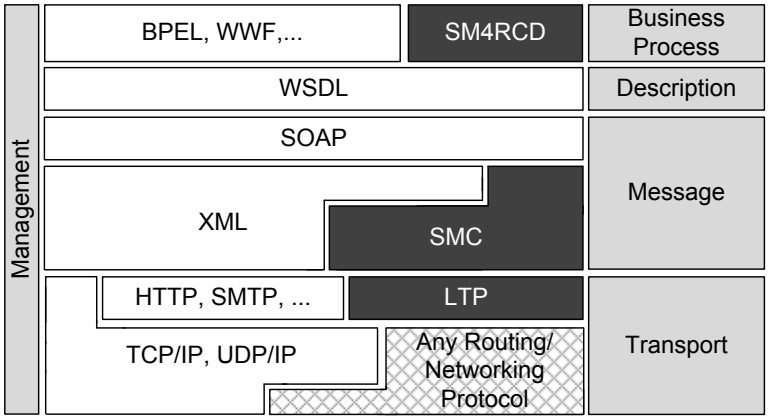


Figure. 1: Web Service Technology Stack (including our extensions highlighted in dark gray)

Services offered by a sensor node should thus be ideally stored on the node itself. A node can then advertise its presence and its services to the surrounding network.

WSDL is the standard language for Web Service description. However, being an XML dialect, it is very verbose and bulky. While generic (e.g., gzip) or XML-specific (e.g., XMill) compression techniques are readily available, they only work well on rather large documents. However, the Web Services offered by sensor nodes typically contain only few, simple operations. While the resulting WSDL documents are too large to fit into constrained memories of sensor nodes, they are too small for existing compression approaches to work effectively.

We therefore design a compression scheme that works well for relatively small WSDL documents. For XML documents with a known grammar, so-called schema-aware compression can be applied. Since WSDLs are XML documents with a well-defined structure, it can be expected that the compression ratio can be considerably improved. Although schemas for WSDLs are available, they make extensive use of the ANY data type to model large parts of WSDL documents to make it a flexible, generic and extensible format. However, the ANY type does not provide any information exploitable for compression. The key challenge has therefore been to derive an appropriate XML Schema description for WDSL files.

To eliminate the use of the ANY type, we employed an empirical approach. We downloaded a representative set of several hundred WSDL documents from `http://www.xmethods.net` and automatically analyzed the structure of the elements represented by ANY in the schema. Using the results of this analysis, we replaced ANY elements in the schema with an enumeration of the data types found in those documents. With this approach, a majority of WSDL documents can be efficiently compressed, while only few documents use the ANY fallback solution. In addition, we have derived optimized Huffman tables used for string compression from these documents.

A further important aspect is that sensor nodes only *store* compressed WSDL documents, the actual compression and decompression is performed by the backend IT system when programming the sensor node and when invoking it, respectively. Thus, the compressor does not have to run within the constrained resources of a sensor node. For a detailed discussion of the analysis and the resulting schema, we refer the reader to [Glombitza et al. 2010].

4. WEB SERVICES IN SENSOR NETWORKS

If one wishes to integrate sensor nodes into enterprise IT solutions, standard-compliant Web Service communications is required in the WSN. In the Internet, HTTP or SMTP on the transport layer and XML-serialized SOAP on the messaging layer are used virtually exclusively. However, these technologies are too resource demanding to be applicable in WSNs and only few works exist that address this issue.

In this section, we present an alternative transport protocol for Web Service messages that may

be used on resource-constraint devices and on the Internet allowing for seamless Web Service invocation. This transport binding – called Lean Transport Protocol (LTP) - is a lightweight protocol conveying Web Services messages and uses existing XML compression techniques in order not to use too many resources (i.e., bandwidth, memory, and processing power) of the sensor nodes.

LTP blends well with the existing Web Service technology as it is fully standard compliant and uses the defined extension points of the technology stack. LTP extends the set of transport protocols (such as HTTP) and XML compression (called SOAP message compression or SMC in the following) provides an alternative representation of the SOAP messages. The integration of LTP and SMC into the standard Web Service technology stack is depicted in Figure 1. In the following, we first discuss SMC and then LTP in detail.

4.1 SOAP message compression

Web Service messages are exchanged using the SOAP protocol. SOAP messages are XML infosets comprised of an optional header and a mandatory body element. In the Internet, these infosets are typically encoded using plain-text XML which are too bulky to be processed, stored, or transmitted by sensor nodes. Thus, compression techniques have to be applied.

General compressors (e.g., gzip) do not yield good compression rates. Better compression may be used by exploiting additional knowledge provided by the grammar that defines SOAP these documents. These grammars are defined using WSDL files which in turn contain XML Schema documents. Both are known at design-time and can hence be used to generate custom-tailored, superior compressors.

Our approach is suited for arbitrary XML compression schemes available for sensor nodes of which only few exist today. In our current implementation, we use microFibre [Pfisterer et al. 2007] compressor which features a mature code generation framework that emits (de-)serialization code for various WSN platforms. The resulting source code contains data type definitions and methods to (de-)serialize SOAP messages directly from/to a compressed binary representation to in-memory data structures. Additionally, for WSDL-defined Web Services, microFibre generates stubs and skeletons translating incoming messages automatically into function calls.

4.2 Lean Transport Protocol (LTP) for Web Services

Next to an efficient message encoding, the transparent and efficient transport of SOAP messages is the second key challenge. With LTP, we propose a lean Web Service transport protocol enabling the exchange of Web Service messages between i) sensor nodes within a single WSN, ii) hosts in the Internet (e.g., enterprise-IT servers), iii) sensor nodes and hosts in the Internet, and iv) sensor nodes in different (remote) WSNs.

To support these four communication patterns, LTP is agnostic to the underlying network transport protocols. For instance, on the Internet, UDP or TCP over IP may be used. However, neither UDP nor TCP can applied in most sensor network environments. In addition, due to resource-constraints, different types of duty-cycling, clustering, MAC-layers, etc., the choice of a certain networking protocol is typically highly application-specific. As a result, LTP has been designed to support arbitrary networking protocols.

For a message exchange between the Internet and the WSN, a gateway converts the network protocol from TCP or UDP to the specific WSN protocol. In the (currently unlikely) case of the WSN and the Internet running the same networking protocol, no conversion is required. An example for this is the use of IPv6 in the Internet and 6LoWPAN (an IPv6 adaptation layer for WSNs). For Web Service communication between sensor nodes located in different WSNs, messages are routed via the respective gateways connected via the Internet.

To address Web Services, so called endpoint references are used. Endpoint references are typically URLs (e.g., `http://example.com/service`). To address endpoints located in WSNs as well as in the Internet, LTP uses an URL-based addressing scheme. For developers it is transparent, whether the endpoint is located on a sensor node or on an enterprise server in the

internet.

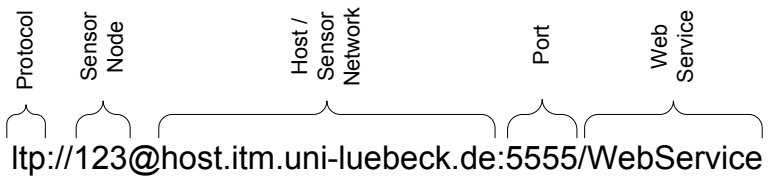


Figure. 2: URL of a *LTP* Endpoint

Figure 2 shows the structure of an LTP URL having *ltp* as protocol prefix. The host and port part are addressing the host and port where an endpoint is located. If adding the "@" sign in conjunction with a node ID ahead of the host, then the host and port are interpreted as the gateway to reach the sensor node via. The path part indentifies the Web Service since multiple services can be provided on the same port.

With this addressing scheme in conjunction with different underlying network protocols, the message transport is realized as LTP's core feature. In addition, LTP provides features to optimize message transport in environments of resource-constrained devices to minimize the resource consumption. To reduce the protocol overhead in terms of packet size, several optional header compression schemes are provided.

To reduce the overhead of the bulky URLs, one of LTP's header compression schemes replaces the string-encoded URLs with integer-based representatives called *aliases*. At gateways URLs are replaced with *aliases* when entering a WSN and vice versa when entering the Internet. The allocation of *aliases* is realized decentralized and dynamically. Next to compression, packet fragmentation is another important feature of LTP to cope with the limitations of radio interfaces in WSNs where payloads of one packet are around 100 Byte.

The payload conveyed in LTP packets are (optionally compressed) SOAP messages. A field in the LTP packet identifies the compression scheme used. In the Internet, plain text XML can be used while between a WSN and Internet-based hosts, compressed XML must be used to overcome resource limitations. The actual binding of a Web Service to LTP as its transport protocol and a certain XML serialization mechanism is specified in standard WSDL files. For instance, a WSDL file of a specific Web Service could choose LTP instead of HTTP and microFibre instead of plain-text XML. The only restriction is of course that both sender and receiver must support these choices. If this is not possible, e.g., since most Web Service stacks only support HTTP and XML, the approach presented in Section 6 can be used. For detailed information about LTP and SMC please refer to [Glombitza et al. 2010a].

As related work, Priyantha et al. [Priyantha et al. 2008] and Yazar et al. [Yazar and Dunkels 2009] present Web Service-based approaches for WSNs utilizing IPv6, 6LoWPAN, and HTTP. But as far as possible these approaches are following the REST [Fielding 2000] principle instead of standard Web Serivices that use SOAP messages. Amundson et al. [Amundson et al. 2006] present an approach allowing enterprise IT components to access the WSN via Web Services. But inside the WSN no Web Service technology is used.

5. BUSINESS PROCESS MODELING FOR SENSOR NETWORKS

Today, the predominant methodology in enterprise IT is to employ the concept of Service-Oriented Architectures (SOAs) for realizing business processes. In such architectures, self-contained distributed services communicate using standardized protocols (such as Web Services). These services can be (re)-composed to higher-level services to form business processes. Instead of re-implementing application logic each time from scratch, applications are implemented or adapted

by simply (re-)composing existing services. SOAs provide the flexible composition of existing functionality and are the basis for programming-in-the-large.

With Web Services available in both the sensor network and the Internet, the architectural pattern of SOAs can be used to implement applications that span both domains. However, to realize service composition on sensor nodes, classical programming languages like C or C++ are still required. This requires expert skills and specially trained programmers and due to the embedded nature often takes more time to develop working, error-free applications compared to traditional software running on PCs. Since the early 1990's, a number of business process concepts have been developed. Here, services are composed to business processes using domain-specific (mostly graphical) languages that can be directly executed by a business process execution runtime or be translated to source code. As a result, software quality, time-to-market and flexibility were improved while dramatically reducing costs.

In order to achieve the same goals in sensor networks, we suggest that using such business process modeling should be used. We present how to integrate sensor networks into business processes which are partly executed in the Internet as well as on sensor nodes. Our contribution is completely based on Web Service technologies and uses well-known standard business process modeling languages.

5.1 Business Processes on the Internet

In enterprise IT, two major widely-accepted standards are currently used: BPEL and WWF (Windows Workflow Foundation, [Microsoft Corporation 2009]). Both are based on standard Web Services and allow a flow-based definition and execution of business processes respectively workflows. In the Web Service technology stack (cp. Figure 1), they are located on two of WSDL and SOAP as they use both technologies to realized business processes.

Basically, a business process composes a number of services and again exposes the composed services as a single process that may be used just like a normal Web Service. Such processes are typically executed by so-called business process execution runtime (BPER) on enterprise IT servers. A BPER performs two major tasks: it executes the process logic and realizes the communication with other processes and services. To integrate services located in sensor networks into such business processes, neither BPEL nor WWF require any adoptions. The only requirement is that the Web Service runtime environment used by the respective BPER implementations supports a Web Service transport binding which is applicable in WSNs. In our case, support for LTP is required. If this is not possible, the approach presented in Section 6 may be used.

To realize, test and evaluate our approach, we have integrated support for LTP and SMC into the widely-used Apache Axis2 Web Service framework. This in turn is used by the open source BPER "Apache ODE" (`http://ode.apache.org`). In addition, we added support for LTP and SMC to the WWF framework which is part of Microsoft's .NET distribution. As a result, both BPERs can now seamlessly interact with Web Services that use LTP as their transport protocol. This includes sensor nodes as well as hosts on the Internet. For more information on implementation details, please refer to [Glombitza et al. 2009].

5.2 Business Processes on Sensor Nodes

Now that WSNs are gradually moving from academic research to industry deployment, the requirements have changed dramatically. First, programming of WSN applications must become easier, not requiring specially trained programmers anymore, faster, and less error-prone. Second, the application logic of sensor nodes is not static or custom-tailored for a single experiment anymore but must be constantly adapted to changing customer and market requirements. Third, the technologies used to program sensor network applications must be based on industry standards to blend well with existing technology deployed in enterprises and customers.

Currently, virtually no approaches exist that tackle these challenges. To address these, we propose to utilize business process modeling concepts using graphical domain-specific DSLs. These can then be used to design WSN applications and to run business processes on sensor nodes

without requiring low-level programming languages or specialized embedded programming skills.

Our approach is comprised of two different components: a flow-based composition of services and a state machine capturing the application's state and transitions to other states. For the first component, we use BPEL that allows modeling the flow-based composition of existing Web Services to higher-level aggregated services. The advantage of BPEL is its wide acceptance in industry and the availability of mature graphical modeling tools. For the second component, no such widely used equivalent to BPEL is available. Hence, we have designed our own domain specific language that we call "State machine for resource-constraint devices" or SM4RCD.

This XML dialect allows modeling a state machine that captures an application's state and transitions to other states. Upon entering or leaving a state, so-called *actions* can be executed. An action is a simple C++ code fragment or an invocation of a Web Service. Transitions to another state are triggered either by *timer events*, *code events*, or by incoming Web Service invocations. Transitions can trigger actions, too.

Consider the center part of Figure 5 where an exemplary state machine is depicted. Here, the three states *measure*, *alarm*, and *sleep* are shown. The initial state (measure) is denoted by the arrow from the black dot. Upon entering the state, the Web Service "measurement" is invoked and a *timer event* is used to periodically re-enter this state. A *code event* is used to transition to a "sleep" or "alarm" state. As a result, it is possible to define the states of a business process graphically using SM4RCD. Since all actions can be Web Service calls, they can also be defined graphically using BPEL. Thus, the whole WSN application can be simply "clicked together" using graphical modeling tools (cp. Figure 3). In this example Microsoft's Visual Studio, Sun's Netbeans IDE, and our custom SM4RCD designer are shown.

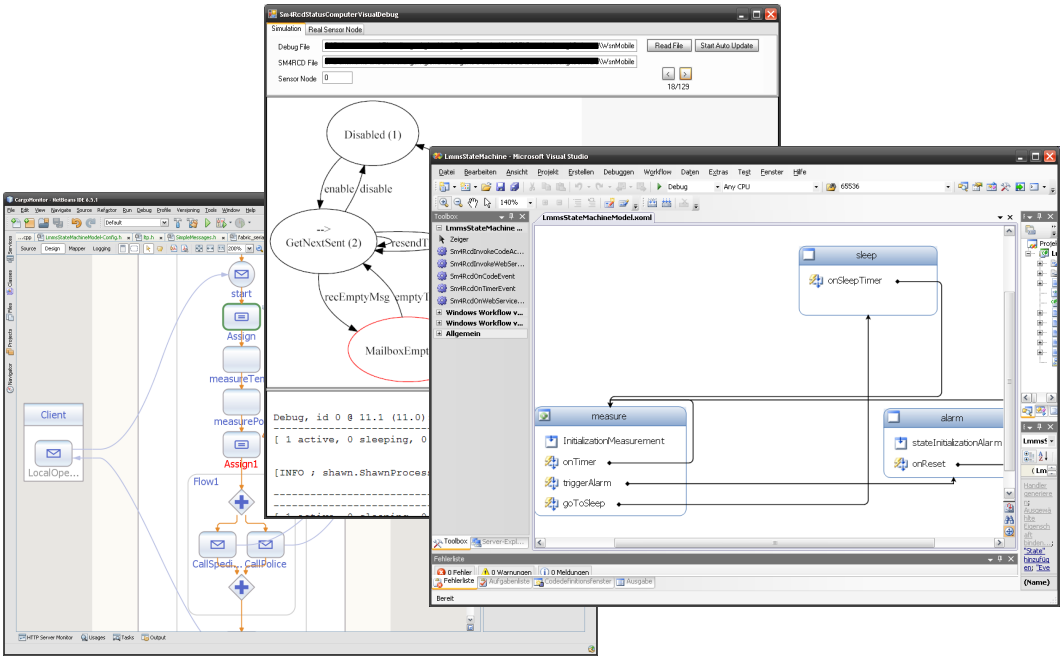


Figure. 3: Designing and debugging WSN business processes graphically.

As BPEL and SM4RCD are XML dialects, their storage and execution on sensor nodes is not possible due to resource constraints. To overcome this limitation, we have implemented a code generator that translates BPEL and SM4RCD models to C++ code for a number of sensor node hardware platforms. This code then contains the business processes' logic and the functionality to execute the process executions. In our implementation, the generated code uses LTP to invoke

and offer Web Services and microFibre XML compression to reduce the size of the SOAP messages significantly as described in Section 4.

Using a graphical process modeling ensures the quality of the business process model and the resulting software given that the translation tools work. However, a monitoring of processes during testing and debugging is essential. As shown in Figure 3, we provide a graphical monitoring tool to visualize the process execution and debugging information of recorded as well as of real-time data. The debugging tool can be used for simulations as well as while executing a process on a sensor node. For more information about this tool, please refer to [Glombitza et al. 2010b]).

5.3 Related Work

In literature, there are some papers presenting business process modeling or model driven software development (MDSD) approaches for resource constrained devices. But to the best of our knowledge, there is no contribution realizing business processes which are executed inside the WSN using standard-compliant Web Service technologies.

With *Flow* [Naumowicz et al. 2009], Naumowicz et al. present a graphical editor and different domain-specific languages for a model-driven WSN application development. But neither Web Services nor the SOA concept is utilized in that approach preventing seamless integration into enterprise IT systems. Pandey et al. [Pandey and Patel 2009] present an approach using BPEL to integrate WSNs into business processes. The BPEL processes are executed in the backend and proprietary communication protocols are used inside the WSN. Spiess et al. [Spiess et al. 2006] present an approach for executing BPEL processes distributed inside the WSN as well as in the backend. But no WS-based communication is utilized inside the WSN. Hackman et al. [Hackmann et al. 2007] are presenting a lightweight BPEL engine called *Sliver*. Sliver is targeting PDA-class devices and is too resource-demanding for WSNs.

6. AUTOMATIC CONVERSION OF MESSAGES AND THE WEB SERVICE TRANSPORT PROTOCOL

With LTP and SMC, we proposed an approach for efficient Web Service communication in WSN and enterprise IT environments (cf. Section 4). As we will show in the evaluation, these approaches even outperform the widely used HTTP (Web Service transport protocol) and XML (SOAP message format) in terms of processing time and message size.

However, as HTTP and XML are predominant in the Internet, LTP and SMC are not going to gain a widespread acceptance or will even replace them. However, LTP and SMC are highly beneficial settings where the performance- and resource-impact of HTTP and XML are not negligible and actually prevent their use. So, it is imperative to support a seamless interaction of HTTP/XML- and LTP/SMC-based Web Services.

To guarantee interoperability between arbitrary transport protocols, we propose a gateway-based approach for the automatic and transparent conversion between different transport bindings as shown in Figure 4. Such a gateway is often called *Enterprise Service Bus* (ESB) in the enterprise IT domain and we will use this term in the following. In our system, the ESB has to support (i) addressing and routing, (ii) protocol conversion, and (iii) payload conversion, which are discussed in the following.

Addressing and Routing. To allow Web Services reachable via LTP to be used by Web Services engines not supporting LTP, a gateway component has to perform an address conversion. As mentioned before, URLs are used as a unifying addressing abstraction. As all Web Service invocations must be routed through a certain gateway (or ESB) that performs the translation, the LTP-URL must be encoded as a HTTP-URL.

A client on the Internet that wants to consume an LTP-based service must therefore use an HTTP-based address pointing to the ESB. Besides pointing at the ESB, the URL must also contain information about the LTP endpoint. This endpoint is simply encoded as a parameter of the HTTP-URL called *url*. Consider the URL `http://esb.example.com/?url=ltp://123@`

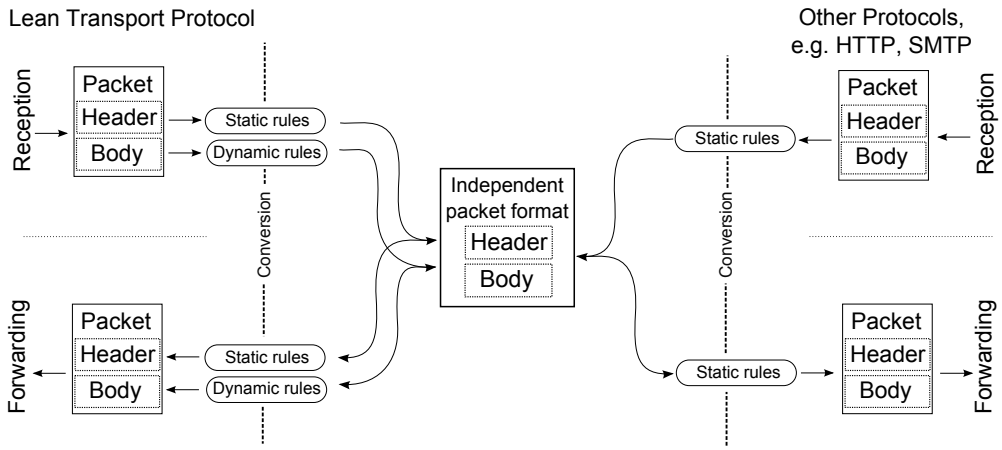


Figure. 4: Protocol conversion component.

`host.gateway.com:5555/Service`. The HTTP URL addresses a host "esb.example.com". The real LTP endpoint (`ltp://123@host.itm.uni-luebeck.de:5555/WebService`) is encoded as a parameter. Please note that to maintain readability, we did not escape special characters in the URL. The ESB then extracts the LTP endpoint and after conversion, it routes the message to the sensor node `123@host.itm.uni-luebeck.de` using LTP.

For Web Service consumers, this solution is transparent since they simply use a standard HTTP-URL. That they are addressing an ESB which only forwards request and response messages to/from an endpoint using a different protocol is hidden from the consumer. The same is true for the Web Service endpoint which does not "know" that it is communicating with and ESB and not with the ultimate Web Service consumer.

Protocol Semantic Conversion. In addition to the conversion of the transport protocols' headers and the message routing, our conversion component deals with protocol semantics as well. In particular, some protocols such as HTTP are synchronous request-response protocols while others are asynchronous one-way protocols (e.g., SMTP and LTP).

Thus, our ESB adds transaction identifiers to asynchronously match request and reply messages. Furthermore, it keeps track of the state of active transactions. In addition, the ESB must delay replay to the synchronous HTTP request until the asynchronous reply is received.

Payload Conversion. The rules for performing conversion of the transport protocol and its semantics are known, static, and independent of specific Web Services. Hence, they can be implemented once to support a number of transport protocols such as HTTP, SMTP, and LTP.

However, the payload of Web Service messages is specific to each individual Web Service and cannot be supported generically. The most challenging part of the protocol conversion is the conversion of the payload. While most well known Web Service transport bindings use XML serialized SOAP messages, LTP uses compressed SOAP messages. As mentioned before, the most efficient compression technique is to exploit knowledge from the WSDL definitions to generate custom-tailored, schema-based compressors for each service. To be able to convert the payload, this knowledge must be available to the ESB which means that the WSDL and the contained XML Schema definitions are required to derive the conversion rules for a specific Web Service.

To provide a transparent, dynamic and configuration-free conversion, we retrieve the WSDL descriptions from the Web Services at runtime. In a second step, the conversion rules are generated dynamically from these descriptions. To minimize the overhead, these dynamic conversion rules are cached for further usages. For further details about the implementation of such a transport

binding protocol converter, we refer the reader to [Glombitza et al. 2010].

As related work, Amundson et al. [Amundson et al. 2006] present an approach providing SOA-based communication not using Web Service-based communication inside WSNs while using Web Services in environments of enterprise IT systems. A gateway is needed to convert between Web Services and the proprietary middleware solution. Furthermore, this solution is not a generic conversion component. It can only convert messages of explicitly deployed and configured Web Services. De Souza et al. [de Souza et al. 2008] introduce a middleware to integrate a Web Service based shop floor into enterprise IT. This middleware includes a DPWS-enabled (Devices Profile for Web Services, [Driscoll and Mensch 2009]) service bus which mediates between the interfaces of the different components including WSNs. However, as in Amundson's approach, no Web Services are used inside the WSN.

7. EVALUATION

In this section, we present evaluation results which demonstrate advantages as well as costs of our approach. The use case for our evaluation is based on the one introduced in the introduction. The model of this use case is shown in Figure 5. It is comprised of two Web Services modeled in BPEL and one state machine modeled in SM4RCD. These models have been produced exclusively using the graphical process modeling tools presented above. The Web Services use LTP as their transport protocol and microFibre [Pfisterer et al. 2007] for XML compression.

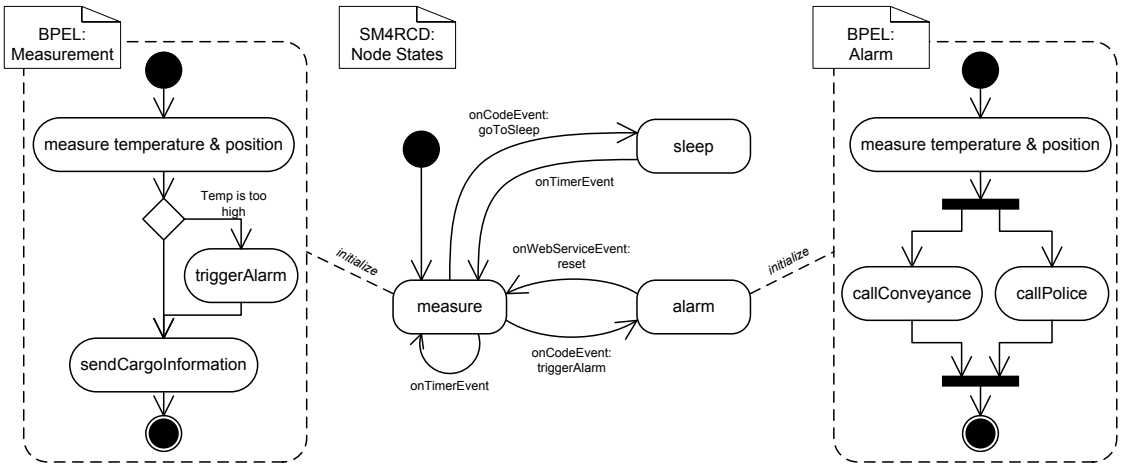


Figure. 5: Logistics business process modeled in BPEL and SM4RCD

7.1 Web Service Discovery and Self-Description

The implementation of our discovery approach fulfils the requirements of sensor networks as well. With 27 B messaging overhead for a request and a reply packet, the protocol overhead is very small. Since a response message carries the service description information, additional payload data is transferred in response messages.

The processing time (measured on a Pacemate sensor node) for the de-serialization of the compressed WSDL descriptions is not larger than 10 ms. Thus, these messages can be serialized just in time on sensor nodes. However, in typical scenarios, these documents can also be generated at design time and be stored compressed.

To evaluate our WSDL compression approach, we compared it with gzip (general compressor), XMill (general XML compressor, not schema-aware) and the original size of the WSDL document. With 50% for small documents and about 30% for larger documents, our approach outperforms

the competitors. Compared to the original WSDL documents, we achieved compression ratios between 84.3% and 85.9%. For a detailed analysis, we refer the reader to [Glombitza et al. 2010].

7.2 Lean Transport Protocol (LTP)

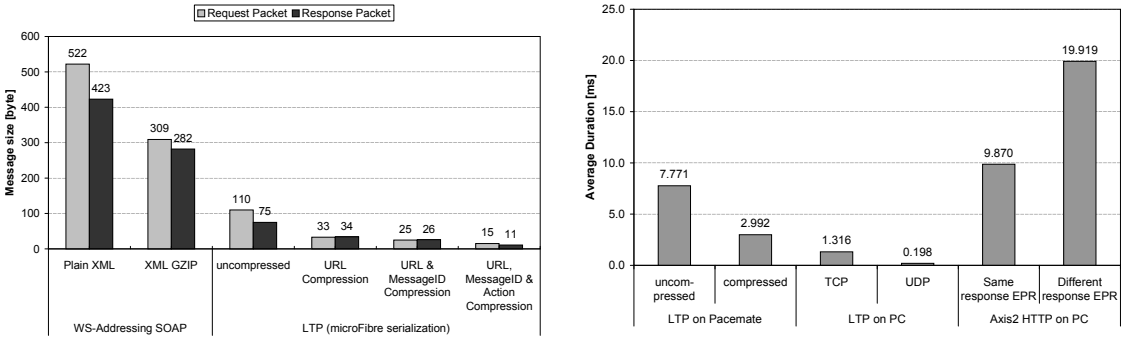
Code Size: Table I shows the compiled code sizes for the Pacemate[3] sensor network platform. For LTP about 7.1 kB are needed and 2.0 kB for the XML compression of SOAP messages. In addition to the communication components, 5.3 kB are needed for the *Measurement* and *Alarm* BPEL processes and 1.8 kB for the SM4RCD-based state machine representing the process' states. In summary, with about 16.3 kB, the total code size of our use case process is very small and easily fit on typical sensor network platforms.

Table I: Code sizes of the use case [in byte]

Communication		**Process Logic**		$\sum$
Compr. SOAP Messages	LTP	State Machine	BPEL	
2,036	7,052	1,816	5,372	16,276

Payload Size: At the same time, the sizes of the exchanged (compressed) SOAP messages are very small. In our example, three Web Service messages are exchanged. The *Measurement* BPEL process sends cargo information to another Web Service (in the Internet), the *Alarm* BPEL process sends an alarm message to the conveyance department and to the police. Finally, the SM4RCD state machine receives a reset message to leave the alarm state. The message to send the cargo information to the backend carries a temperature value as well as the latitude and longitude which are represented with three restricted integers (degree, minute, second) each. Such a compressed SOAP message is serialized using 7 B. The alarm messages contain the same information as the cargo information message plus an alarm code which is represented as integer. This message is serialized with 8 B. To serialize the reset message only 1 B is needed.

Memory Usage: Next, we evaluated the RAM requirements during the execution of LTP while processing an incoming and an outgoing packet. The minimum memory usage after initialization is 40 B. During processing, the maximum memory consumption stays below 523 B if using LTP's header compression capabilities as well as 1,283 B without header compression (due to uncompressed URLs).



(a) Comparison of packet sizes.

(b) Roundtrip times for a Web Service invocation.

Figure. 6: Message overhead and processing performance of LTP

[3]32 bit ARM controller operating at 60 MHz.

In terms of communication bandwidth, LTP's overhead is very low. Since LTP implements features comparable to WS-Addressing, we present measurements of both. Figure 6 (a) shows the packet overhead of an LTP packet and a SOAP message with WS-Addressing. In both scenarios, a request and a response message is transferred with a payload of 0 B. Even with no header compression, LTP has compression ratio of 82% for the response and 78% for the request message compared to WS-Addressing. Using full header compression, LTP's compression rate can be improved to up to 97%.

Processing Performance: To measure the performance of LTP, we implemented a simple Web Service using HTTP and SOAP (measured on a standard PC) as well as LTP and SMC as transport bindings. Typically in a HTTP-based Web Service communication, the same connection is used for request and response. With 1.3 ms using TCP and 0.2 ms using UDP, compressed SOAP over LTP is performing way better on a PC with an average of 9.9 ms. LTP provides the flexibility to send response messages to a different receiver which can be realized with HTTP and WS-Addressing by using two HTTP connections. In that scenario HTTP is performing even worse with a roundtrip time of 19.9 ms. Even the performance on a Pacemate sensor node with 7.8 ms (using no header compression) and 3.0 ms (using full header compression) is better than HTTP on a PC.

7.3 Running Business Processes on Sensor Nodes

The biggest challenge to realize a Web Service-based business process-driven development for sensor networks was to realize Web Service communication inside the WSNs. As shown and discussed in the code size evaluation of our use case, the code size of the application logic generated with a graphical tool support is very small and thus fulfills the resource requirements of sensor nodes.

We evaluated the dynamic memory consumption of a simple stateful BPEL process which receives and answers two Web Service messages. After initialization, the memory consumption of the BPEL process is 582 B which is only 1.7% higher than of the same process which was implemented manually. The same is true for the maximum memory consumption. With 2,025 B BPEL has only a 0.5% higher memory consumption. In both, the manual as well as the model-driven process implementation, with about 1.3 kB, LTP generates a fraction of the total memory consumption.

7.4 Protocol Conversion

To evaluate our implementation of our protocol conversion approach, we measured the runtime performance of an example Web Service and the conversion between the transport bindings HTTP/SOAP as well as LTP/SMC. The Web Service takes the ID of a good as input and returns time of loading, ID of the producer, number of stored units, as well as a URL pointing to a Web page with further information about the good.

With about one second, the initial generation of the dynamic conversion rules is very expensive. The conversion of an incoming LTP/SMC message to HTTP/SOAP requires 0.61 ms while vice versa requires 13.35 ms (measured on a standard PC). The significantly higher amount if time for the conversion from HTTP/SOAP to LTP/SMC is mainly generated by expensive XSLT and DOM manipulations which are not required while converting from LTP/SMC to HTTP/SOAP.

8. CONCLUSION

In this paper, we have present a comprehensive, Web Service-compliant approach to use SOA technologies in WSNs. Our approach comprises self-description of sensor nodes, a light-weight Web Service transport protocol (called Lean Transport Protocol, LTP), an approach for the model-based programming of sensor nodes, and a solution for the integration with the Internet. Our exhaustive simulation showed the first-rate performance of our implementations.

REFERENCES

Amundson, I., Kushwaha, M., Koutsoukos, X., Neema, S., and Sztipanovits, J. 2006. Efficient integration of web services in ambient-aware sensor network applications. In *3rd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BaseNets 2006)*.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. 2001. Web Services Description Language (WSDL) 1.1. Tech. rep., World Wide Web Consortium W3C. http://www.w3.org/TR/wsdl.html.

de Souza, L. M. S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., and Savio, D. 2008. Socrades: A web service based shop floor integration infrastructure. In *IOT* (2008-04-15). Vol. 4952. Springer.

Driscoll, D. and Mensch, A. 2009. Devices Profile for Web Services V 1.1. Tech. rep., Organization for the Advancement of Structured Information Standards (OASIS).

Fielding, R. T. 2000. Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine.

Glombitza, N., Mietz, R., Römer, K., Fischer, S., and Pfisterer, D. 2010. Self-Description and Protocol Conversion for a Web of Things. In *Proceedings of 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2010)*.

Glombitza, N., Pfisterer, D., and Fischer, S. 2009. Integrating Wireless Sensor Networks into Web Service-Based Business Processes. In *MidSens '09: Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*. ACM, New York, NY, USA, 25–30.

Glombitza, N., Pfisterer, D., and Fischer, S. 2010a. 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)*. Boston, Massachusetts, USA.

Glombitza, N., Pfisterer, D., and Fischer, S. 2010b. Using State Machines for a Model Driven Development of Web Service-Based Sensor Network Applications. In *Proceedings of ICSE '10: 32nd International Conference on Software Engineering*.

Hackmann, G., Gill, C., and Roman, G.-C. 2007. Extending BPEL for Interoperable Pervasive Computing. In *Proceedings of the 2007 IEEE International Conference on Pervasive Services*. 204–213.

Microsoft Corporation. 2009. Windows Workflow Foundation. WWW: `http://msdn.microsoft.com/en-us/netframework/aa663328.aspx`.

Naumowicz, T., Schröter, B., and Schiller, J. 2009. Poster Abstract: Prototyping a Software Factory for Wireless Sensor Networks. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*.

OASIS WS-BPEL Technical Committee. 2005. Webservices – Business Process Execution Language Version 2.0. Tech. rep., WWW: `http://www.oasis-open.org/committees/wsbpel`.

Pandey, K. and Patel, S. 2009. A Novel Design of Service Oriented and Message Driven Middleware for Ambient Aware Wireless Sensor Network. In *International Journal of Recent Trends in Engineering (IJRTE)*.

Pfisterer, D., Wegner, M., Hellbrück, H., Werner, C., and Fischer, S. 2007. Energy-optimized Data Serialization For Heterogeneous WSNs Using Middleware Synthesis. In *Proceedings of The Sixth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net' 07)*. 180–187.

Priyantha, N. B., Kansal, A., Goraczko, M., and Zhao, F. 2008. Tiny web services: Design and implementation on interoperable and evolvable sensor networks. In *SenSys'08: Proceedings of the 6th ACM conference on Embedded network sensor systems*.

Spiess, P., Vogt, H., and Jutting, H. 2006. Integrating sensor networks with business processes. In *Real-World Sensor Networks Workshop at ACM MobiSys*.

Yazar, D. and Dunkels, A. 2009. 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.

Nils Glombitza received his diploma in Business Informatics at the Carolo-Wilhelmina University of Technology at Brunswick, Germany. Since June 2007 he is working as Ph.D. student and research assistant at the Institute of Telematics, University of Lübeck lead by Prof. Dr. Stefan Fischer. His research is focused on service orientation and business process management in sensor networks.

Dennis Pfisterer is a senior researcher and tenured lecturer at the Institute of Telematics, University of Lübeck, Germany headed by Prof. Stefan Fischer. After his degree in Information Technology in 2001, he worked as a research assistant at the European Media Laboratory on resource adaptive systems. In 2003, he joined Prof. Fischer's group at the Braunschweig Institute of Technology, and followed him in 2005 to Lübeck. Before his PhD, he worked on algorithmic aspects of large-scale sensor networks with results such as the Shawn simulator or the SpyGlass visualization framework. After his PhD, he broadened his research interests and now works on sensor networks in general, their integration into the Future Internet, and Service Oriented Architectures in resource-constraint environments. He has published more than fifty peer-reviewed publications in his area of expertise. More information is available at http://www.itm.uni-luebeck.de/users/pfisterer.

Stefan Fischer is a full professor in computer science at the University of Lübeck, Germany, and the director of the Institute for Telematics. He got his diploma degree in "Wirtschaftsinformatik" and his doctoral degree from the University of Mannheim, Germany, in 1992 and 1996, respectively. After a post doctoral year at the University of Montreal, Canada, he joined the newly founded International University in Germany, one of the first private universities in Germany, as an assistant professor in 1998. In 2001, he became an associate professor in computer science at the Technical University of Braunschweig, where he stayed until 2004, when he joined Lübeck University. His research interest is currently focused on new network and distributed system structures such as ad-hoc and sensor networks. He has (co-)authored about 100 scientific books and articles. Dr Fischer is a member of ACM and the German Gesellschaft für Informatik (GI).