Services to the Field: An Approach for Resource Constrained Sensor/Actor Networks

Christian Buckl, Stephan Sommer, Andreas Scholz, Alois Knoll, Alfons Kemper
Technische Universität München
Boltzmannstr. 3, D-85748 Garching, Germany
{buckl,sommerst,scholza,knoll,kemper}@in.tum.de

Jörg Heuer, Anton Schmitt
Corporate Technology, Multimedia and Network Communication, Siemens AG
D-81730 München, Germany
{joerg.heuer,anton.schmitt}@siemens.com

Abstract

Nowadays more and more devices of daily life are connected to each other and are integrated into massively distributed networks of embedded devices. These devices range from consumer electronics such as digital picture frames or internet radios to embedded devices such as fridges or home control in general. The service-oriented paradigm is the main concept to implement complex, heterogeneous and large IT systems. However, if it comes to embedded devices, resource constraints imposed by the underlying hardware, such as 8-Bit micro controllers, require efficient protocols. This often prohibits the use of technologies known from the Web service domain, the major implementation of the service-oriented paradigm. Nevertheless, a quick and seamless information flow between embedded devices and Web services is already today an important requirement for many application scenarios, e.g., real-time aware production management or the Internet of Things. Within this paper, we present an approach that takes benefit of traditional SOA implementations, such as Web service interfaces and an IP compatible addressing schema. One advantage of the solution is that it can be implemented on resource constraint devices. The main innovations are combination of web service mechanisms with a data-centric processing paradigm to ϵServices at the device level and enabling a generic Service Bridge as an agnostic mediator between the Web Service world and the ϵService world on networked embedded devices.

1 Introduction

Traditionally, many devices were not network-enabled or were executed within an isolated device network. Such devices and systems were the domain of experts for embedded system. The ongoing decline in prices for network-enabled hardware devices has created several application domains for sensor/actor networks such as building automatisation. Currently, one can observe the trend to create interfaces between these physical devices on the one hand and enterprise systems on the other hand [6, 10].

Furthermore, interoperability is of great importance. The networks typically consist of heterogeneous components e.g. with different requirements and thus using different communication protocols or processors. To reduce the complexity, a modular architecture that supports the integration of different components, both related to software and hardware components, is required. Furthermore, also the developers have changed. While previously device domain experts were involved in the development of the systems, the systems must now be configurable and manageable by non-experts and even end users. In the area of home automation for example, the configuration of the sensor/actor network should be possible for the tenants. Therefore, new development and configuration tools are required that provide user-friendly graphical user interfaces on platforms they are used to. Nowadays commonly known platforms are, for instance, web browsers which, for instance, are already used by end users to configure wireless routers. Most of these requirements in the domains of interoperability, modularity and tool support can be addressed by the service-oriented paradigm. The emphasis of this approach is on reusable software components that can interact on a basis of com-

monly known protocols. The predominant implementation is based on web services and the different protocols of this area.

In different projects [9], implementations were developed for resource-constrained devices. However, the minimal requirements for implementing a web service stack are still very high due to the processing of character data and the required data conversion in data types required for the data processing. While many developers claim that due to the decline in prices the used hardware components will soon be powerful enough to run this software, this is only one side of the coin. For the other side the tremendous growth of applications and the limitations in bandwidth have to be taken into account which does not follow Moore's law. In many of the new application domains such as building control, a processor for sensors that can run a Web service stack will be too cost-intensive.

Therefore, approaches that support very resource-constraint controllers are required. Furthermore, not only the computational power of the controllers is constrained, but also the network bandwidth. ZigBee [16], one of the major standards for ad-hoc sensor networks, for example is designed to support only small messages in the range of a few bytes. Standard XML messages are much too long to be used in these networks. Within this paper, we suggest a solution that takes into account on the one hand the resource constraints of current sensor/actor networks but on the other hand adopts web service protocols to ease the interaction of web services known from IT systems with these sensor/actor networks. The proposed solution offers the following benefits:

1. Look&Feel of a Web Service: Web services can easily send data requests to devices. For the calling web service, it makes no difference whether a standard web service or a constrained device is the end point of the request.

2. Resource minimal implementation of the SOA middleware: the device world differs from the web service world. The data flow is typically static and directed from the sensor to the actuator. This behaviour can be modelled with a data centric processing paradigm, in which services are implemented as operators that process and transform an underlying data stream.

2 Two Worlds Unite

While SOAs are a suitable programming paradigm for both, Internet-based Web services and embedded networks, the characteristics of individual services differ in both application domains. To avoid ambiguities, we will use the terms SOA and service for Web service (WS) based SOAs, and the terms ϵSOA and ϵService for SOAs in the embedded network domain. First, we want to show the similarities and differences between both notions of services and why it is a feasible approach to differentiate between the embedded world and the rest.

2.1 The Notion of Services

The first aspect we want to compare is the communication principles. In contrast to Web Services, ϵServices often only support asynchronous communication (fire&forget [1]), because of limitations in the employed network protocols and real-time constraints. When a service is called, it processes the incoming data and then changes its state and / or produces output data. Another aspect where Web Services and ϵServices differ is that ϵServices have long living instances, which are not persisted between service invocations, but kept in memory permanently. ϵService are instantiated once, when the corresponding application [2] is deployed and terminated when the application is stopped.

2.2 Data-Centric Embedded SOA

Our approach assumes that a data centric processing model, often also called an actor oriented programming model [13] or a data stream based programming model, is used in the embedded network. These programming paradigms allow the separation of functional components (ϵServices) from non-functional aspects like communication and management. Therefore, it is possible to restrict the data flow to the pure functional data and to avoid the necessity of sending semantic information. The components can be described in a domain specific model with well defined interfaces (ports) and are interconnected by a middleware. The used ϵServices can produce, consume and alter data to provide application functionality. Due to the step wise relisation of applications based on existing components i.e. services, most applications are realized by chaining small services which provide one specific artifact of functionality to the application. This approach also increases code reuse and inherently supports the distributed execution of applications, because the individual services can be distributed over the nodes in the network. In a data centric processing model, applications are built by chaining together services. An important aspect thereby is that each of the individual services is not aware of any preceding or subsequent services in this chain. This knowledge is kept in the message routing layer, which decides where to send the output data of each service. In the Web service domain on the other

[1] If a reliable message transmission is required, it is implemented within the middleware in our approach.

[2] Application here stands for services coupled into a chain to perform a specific action

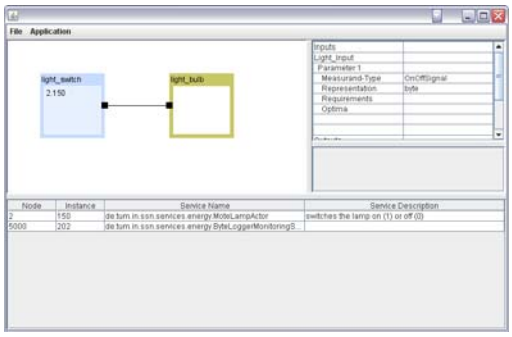

Figure 1. Service Composition

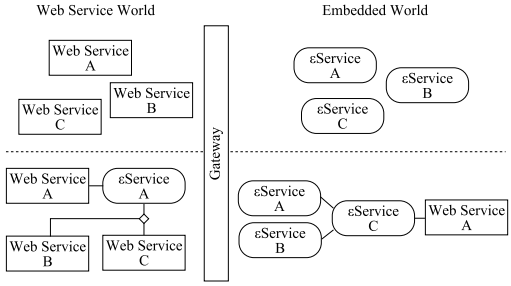

Figure 2. Web Services and Embedded Services - Two Views

hand, typically remote-procedure-call paradigms are used. In this case, the service itself specifies the next service in the data processing chain.

In the following sections, we will refer to our prototypical implementation of such a middleware, which is presented in [4]. However, the presented concepts are applicable for other middleware approaches as well, as long as these also persuade a data centric programming model. In order to get a more realistic model of the real world, our middleware provides two different kinds of ϵServices. The first kind of services are hardware services. These services provide an interface to bring hardware capabilities to the application (e.g. sensing).

Logic services consume data provided by hardware services (sensors) or by other logic services. The data provided by a logic service can be consumed by a further logic services or by a hardware service (actor). Figure 1 shows a simple application pattern following this paradigm comprising two services; one for a light switch sensor, one for the light switch relay.

2.3 Integration of Both Worlds

The upcoming challenge for application developers in general is the integration of both worlds, Web services on the one side and embedded Services on the other side. Real-time awareness[3] for manufacturing or logistics becomes an essential requirement: a break in information exchange between the embedded world and the business back-end is not tolerable anymore. Failures and delays on the device level have to be reported fast, in order to allow the timely execution of compensatory actions. This requirements is for example placed by flexible production environments, which have to be (re-)configurable from back-end services to reduce downtimes and support on-demand production.

[3]In this context, the expression "real-time" does not imply hard timing constraints as known from the embedded world, but should be read as "data should be supplied in a timely manner". This is due to the reason that the Web services consuming data from the embedded networks do not provide real-time guarantees at all.

The integration of the IT and the embedded system has to be performed in two ways, as shown in Figure 2. A developer familiar with Web service technologies should be able to interact with services from the embedded world just like he would interact with any other Web service. E.g., if a business process is modeled with BPEL [3] (as depicted in the lower left part of Figure 2), the process designer should be able to use ϵServices to acquire or submit information to field level devices. On the other hand, a developer familiar with application development for embedded networks should have access to services in the enterprise back-end in the same manner as he accesses other embedded services. E.g., if data has to be transmitted to a back-end Web service, it should be sufficient to simply route the corresponding data stream to the remote service (as depicted in the lower right part of Figure 2).

The Service Bridge [4]is the mediator between these two worlds: it has the capability to generally translates messages in a service agnostic way to faciliate communication between services in both Worlds. By that it provides an abstraction layer that supports both of the above mentioned views. This leads to the following four different interaction scenarios.

Continuous interaction with the embedded network In this scenario, an external Web service continuously interacts with one or more services in the embedded network, e.g., to retrieve measurement values or to submit externally acquired data. In order to keep the communication overhead low and to support non-periodic interactions, the communication between services is managed via subscriptions, i.e., a Web service developer subscribes to the output of an embedded service or announces data submissions to the input of an embedded service. The management of these data subscriptions can be done with established technologies like WS-Eventing[15], which is not in the focus of this paper.

[4]There may exist multiple bridges and a dynamic bridge selection to avoid a single point of failure. We differentiate between a bridge for the case where on both networks the same information is obtainable from a gateway where the gateway only provides a selection of information.

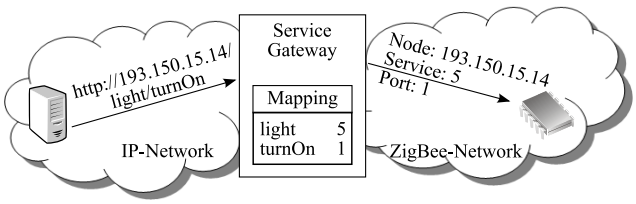

Figure 3. Service Bridge

Ad-hoc interaction with the embedded network In contrast to the previous scenario, the interaction between services is not planned beforehand via subscriptions, but occurs dynamically. RPC-style Web service invocations are an example for this kind of interactions, e.g., in order to retrieve the current measurement value of a sensor, an external service could invoke a *getData* method on an embedded service.

Continuous interaction with external Web services In this scenario, a developer from the embedded domain wants to retrieve data from or submit data to an external Web service in a periodic fashion. This interaction has to support the stream based paradigm used in the embedded network, i.e., to submit data to the external service the developer routes a stream to the Web service, to receive data he routes a stream from the Web service to the embedded service.

Ad-hoc interaction with external Web services The last interaction mode is not meaningful for data centric services as used in the embedded domain. In the embedded world, applications are installed by chaining services running on the individual nodes. As the individual services have no knowledge about the concrete wiring, reconfigurations of the application are only triggered by end-users (typically in the web service domain) or the middleware, but not by the ϵServices.

3 Bridging the Gap: Services to the Field

After stating the different interaction schemes between web services and ϵServices, this section discusses the main concepts of the implementation: IP-compatible addressing and the Service Bridge.

3.1 IP-compatible Addressing

A main requirement for a seamless integration of the web service world on the one hand and the ϵSOA world on the other hand is a compatible addressing scheme. As the Internet Protocol is the de facto standard for most used protocols, we advocate an IP-compatible addressing scheme. This approach resembles the approach in [12]. However in contrast to this approach of virtual IP addresses; the IP address is actually used for the routing algorithms within our middleware. All devices in the ϵSOA world have an IP address, even if the underlying network uses a different addressing scheme[5]. The bridge monitors all incoming messages at the IP layer and intercepts messages targeted at an ϵService. These messages are translated into suitable packet formats and forwarded to the ϵService. Figure 3 shows a possible scenario. A web service call is routed via the bridge. The bridge monitors the messages and translates the message to the ZigBee protocol. The mapping of the Web service call to the specific service and port is performed by analyzing the WSDL description.

3.2 Service Bridge

If a developer wants to access an external Web service from the embedded world, the Service Bridge creates a virtual embedded service representing this Web service. The virtual service's in- and outputs are created according to the WSDL description of the Web service. For continuous interaction, the *One-way* and *Notification* WSDL port types are supported. A *One-way* port in a WSDL specifies a port, which only receives messages. The virtual service will therefore possess a corresponding input. Analogously, an output is created for every *Notification* port of the WSDL. The correlation between these ports is stored in an internal mapping table in the Service Bridge. From the view of the embedded network, the virtual service is offered by the node hosting the Service Bridge. In order to send data to the external Web service, an embedded service can send data to the input of the virtual service running on the Service Bridge node. The arriving messages are intercepted by the Service Bridge and converted to a SOAP call. The destination Web service is determined with the mapping table and the message is forwarded to its destination in the Web service world. Incoming SOAP messages are treated analogously: they are intercepted by the Service Bridge and converted to embedded network messages. These messages are injected to the network, as if the output of the virtual service created them.

The Service Bridge does not directly support ad-hoc interaction with external Web services. We do not think, if needed at all, it will reflect rare cases as this mode violates the data centric processing paradigm in the sensor network. Several benefits of data centric systems, like free placement of services, splitting and re-using of data streams, etc are only achievable if the individual services which operate on a data stream are implemented "locally", i.e., produce

[5] In this case, the routing to the next hop is realized by the addressing scheme of the underlying network

their outputs solely depending on the data received. An ad-hoc interaction would require the service to decide which external Web service it should address, what violates this paradigm. If the ad-hoc interaction is needed anyway, it can be mimicked by installing temporary data streams for the duration of the invocation. The message exchange in this case is the same as in the continuous interaction scenario.

In order to make an embedded service accessible from the Web service world, a WSDL generator creates a WSDL document describing the embedded service's interfaces. It will contain a *Notification* type port for every output of the service and a *One-way* port for every input of the service. Analogously to the interaction with external Web services, the correlation between these ports is added to a mapping table. Additionally, the newly generated WSDL is made available through a UDDI based discovery interface, which allows users form the Web service world to search for specific embedded services. The message exchange in the continuous interaction mode is the same as described in the previous paragraphs.

Finally the support for ad-hoc interactions initiated from Web services requires mediation between the pull based request/response invocation scheme in the WS domain and the push based communication paradigm in the embedded world.

In this case, the Service Bridge will install a caching service and extend the WSDL with a "getter" method for the corresponding output. The caching service has two inputs and one output. The data input is connected with the output of the target service. The caching service will always store the latest data received at this input. If a message is sent to the second input, the trigger input, the caching service will send the stored data from the cache output. The last measurement produced by the target service is therefore pullable via a call to the trigger input. If an embedded device supports on demand data acquisition, i.e., data acquisition can be triggered via submission of a message, the cache service is not needed. Upon the arrival of a request the Service Gateway will trigger the measurement at the target service and send the reply to the Web service.

Summarizing this section it has to be concluded that even though the Service Bridge has to provide means to intermediate between Web services and ϵService this mediation is generic thus service agnostic and does not require adaptations if new services are added.

4 Demonstrator

A smart building to support energy efficiency and an excellent quality of life at the same time is one of the main drivers in home and building automation and characterized by buzzwords like Intelligent House or Smart Home. Although the necessary technologies for implementing this vision are already available, general rollout has not taken place yet. One reason may be the significant installation costs especially in private households. On the other hand, sophisticated building automation is increasingly installed in industrial and business buildings, especially to lower management costs. Traditional building automation approaches emulate classical wired switched circuits and become significantly complex with an increasing or even changing number of sensors and actuators. This becomes even worse if further optimization criteria such as power efficiency are considered. We therefore selected the building automation scenario to demonstrate our ϵSOA as we believe that our approach fulfills the various requirements in this domain.

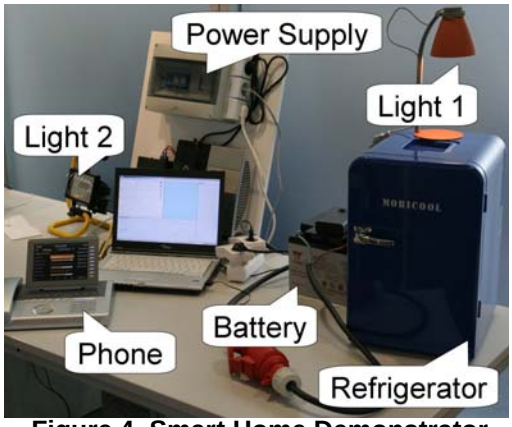

Figure 4. Smart Home Demonstrator

Based on our ϵSOA platform, we developed a demonstrator, which covers a future home automation scenario. The assembling of our demonstrator is shown in Figure 4. We assume that in the next years, energy providers use dynamically changing energy prices in order to influence the overall energy consumption depending on the energy availability and the wholesale energy market. We further assume that some kind of power storage system, such as the battery of an electric car, will temporarily be available in future homes. Based in these assumptions, we implemented the following scenario: A household comprising a battery and loads (a refrigerator and 2 lights) with different power consumption and energy saving options. One task of the automation logic is to optimize the power consumption costs throughout the day. If energy prices are low, the battery is charged and the refrigerator cools down to a lower bound. If prices are high, the house is disconnected from the power grid if the battery is fully charged and then draws its energy in peak price situations from the battery. Additionally, the refrigerator is put to energy saving mode, i.e., it stops cooling until an upper temperature bound is reached. The electricity prices are provided by an external Web service, which is ”bridged” by a Service Bridge into the network. This is an example for scenario of continuous interaction

with an external Web service as described in the previous section.

Furthermore, tenants can monitor the system by establishing a connection to an ϵService that provides the current operation state of devices in the household abd also the overall cost of energy consumption. This represents the scenario of continuous interaction with an ϵService as discussed in the previous section.

Finally, the tenants can also query the current temperature of the refrigerator or configure the control system e.g. setting the lower temperature bound. This this is an example of the ad-hoc interaction scenario in which the web services interacts with the device network in an ad-hoc fashion.

There are other functional requirements not presented in detail here, e.g., the home has to connect to the power grid if the summed consumption of all devices exceeds the power of the battery, the battery should not be completely depleted depending of the mobility requirements of the eCar, etc. The used ZigBee based motes possess a set of I/O devices used to read signals from the switches and turn on or off the loads. The programmable phone, for instance, can be used to monitor sensor readings and to adjust thresholds, such as the pricing parameters to trigger reduced power consumption of devices such as the fridge.

5 Related Work

The major contribution of our approach is the integration of the data-centric embedded world and the Web service world. Most existing approaches try to map one approach to the other world.

For the embedded world, standardized middleware architectures target specific application domains, e.g., KNX[1] for the building automation domain or AUTOSAR[2] for automotive applications. These approaches work on a very low abstraction level and therefore lack support for a seamless integration with external services. Especially, because the data processing paradigm used in these approaches is not directly compatible with service oriented principles.

Similar to our approach, a service / component oriented approach is followed by the OASiS[11], MORE[14] and RUNES[5] projects. However, these projects do not provide means for service agnostic binding with external Web services.

A web service based approach is persuaded by other projects, such as SIRENA[9] or SOCRADES[7]. These projects are based on the idea that an adopted Web service stack is present at the device level, the DPWS[8] stack. Due to this assumption, the resource requirements on these devices are rather high. Current implementations of a DPWS stack including the OS and a TCP/IP stack require a static memory footprint of 500 kB [9]. In contrast, a prototypical implementation of our middleware requires only 12 kB. Therefore, applications, where dynamic invocations of services provided by devices play a central role, will most probably be implemented using approaches such as DPWS. If however the service wiring is predominantly static and resource constraints influence the design, our approach is more suitable.

6 Conclusion and Future Work

In this paper we presented a new approach of ϵServices and Service Bridge for a generic integration of IP-based Web service and embedded services which use specialized transport protocols. For this approach we assume that a service oriented programming paradigm is required on device level due to the high degree of decentralization in embedded networks.

Contrary to quite a number of other projects we do not think that service oriented approaches ends on the gateway to the embedded domain. On the contrary we demonstrate that it is reasonable to run complete Web service stacks on embedded devices even if theses possesses only reduced capabilities like DPWS stack for example.

The presented approach in this paper rather combines a service oriented approach with a data centric processing paradigm. Even though closely related to Web services the data centric paradigm requires mediation between the Web service world and the embedded world.

An outstanding feature of the chosen approach is that the mediation between Services and Web Services can be provided by a Service Bridge which is agnostic to the actually bridged service. Thus we can offer application developers for embedded networks the possibility to seamlessly interact with Web services located outside of the embedded network, and Web service developers the possibility to access services in the embedded network just like any other Web service without taking care of the bridge in between. The paper discusses the possible interaction schemes between Web Services and ϵServices at device level, presents the main tasks of the Service Bridge, and presents a demonstrator that shows the feasibility of this approach using the example of an energy building management. To further the optimization potential of the presented integrated approach we are currently developing tools that select appropriate routing techniques and calculate a eService placement based on non-functional requirements such as maximal latency or reliability for data links. Furthermore, we are developing concepts for a semantical description of eServices to automate the chaining of services in view of configuration costs reduction.

References

[1] *KNX Handbuch Version 1.1*. KNX Association, 2004.

[2] AUTOSAR – Automotive Open System Architecture. http://www.autosar.org/.

[3] BPEL, Business Process Execution Language. http://www-128.ibm.com/developerworks/library/specification/ws-bpel.

[4] C. Buckl, S. Sommer, A. Scholz, A. Knoll, and A. Kemper. Generating a Tailored Middleware for Wireless Sensor Network Applications. *SUTC*, pages 162–169, 2008.

[5] P. Costa, G. Coulson, C. Mascolo, G. P. Piccoand, and S. Zachariadis. The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems. In *PIMRC'05*, 2005.

[6] S. de Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker. Soda: Service-oriented device architecture. *IEEE Pervasive Computing*, 5(3):94–C3, 2006.

[7] L. de Souza, P. Spiess, D. Guinard, M. Khler, S. Karnouskos, and D. Savio. SOCRADES: A Web Service Based Shop Floor Integration Infrastructure. *IOT'08*, pages 50–67, 2008.

[8] Devices Profile for Web Services. http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf.

[9] F. Jammes and H. Smit. Service-oriented architectures for devices - the sirena view. In *3rd IEEE International Conference on Industrial Informatics*, pages 140–147, Aug 2005.

[10] S. Karnouskos, O. Baecker, L. de Souza, and P. Spiess. Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, pages 293–300, Sept. 2007.

[11] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits. OASiS: A Programming Framework for Service-Oriented Sensor Networks. In *COMSWARE'06*, 2007.

[12] S. Lei, W. Xiaoling, X. Hui, Y. Jie, J. Cho, and S. Lee. Connecting heterogeneous sensor networks with ip based wire/wireless networks. In *Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and Second International Workshop on Collaborative Computing, Integration, and Assurance*, pages 127–132, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J. Liu, J. Eker, J. W. Janneck, X. Liu, and E. A. Lee. Actor-Oriented Control System Design: A Responsible Framework Perspective. *IEEE Transactions On Control Systems Technology*, 12, No. 2, March 2004.

[14] MORE – Network-centric Middleware for Group communication and Resource Sharing across Heterogeneous Embedded Systems. http://www.ist-more.org/.

[15] WS-Eventing. http://www.w3.org/submission/ws-eventing/.

[16] ZigBee. http://www.zigbee.org, 2007.