GATHERING PRODUCT DATA FROM SMART PRODUCTS

Jan Nyman, Kary Främling
Helsinki University of Technology, BIT research centre, P.O. Box 5500, FIN-02015 TKK, Finland
jan.nyman@tkk.fi, kary.framling@tkk.fi

Vincent Michel
École Nationale Supérieure des Mines de Saint-Étienne 158, cours Fauriel F-42023 SAINT-ÉTIENNE cedex 2
vmichel@messel.emse.fr

Keywords: The Internet of Things, distributed systems, smart products

Abstract: The enabling of data produced by product embedded sensor devices for use in product development could greatly benefit manufactures, while opening up new business opportunities. Currently products such as cars already have embedded sensor devices, but the data is usually not available for analysis in real-time. We propose that a world-wide, inter-organizational network for product data gathering should be created. The network should be based on open standards so that it can be widely adopted. It is important that a common, interoperable solution is accepted by all companies, big or small, to enable innovative new services to be developed. In this paper, the concept of the Internet of Things (IoT) is described. The PROMISE project is presented, and a distributed messaging system for product data gathering developed within the project is introduced. Practical experiences related to the implementation of the messaging system in a real application scenario are discussed.

1 INTRODUCTION

The emergence of technologies, such as Radio Frequency Identification (RFID), enable consumer products to be identified individually, and makes it possible for products to have some other information stored on them as well. For example, a box of prescription drug pills could have an embedded RFID tag, which contains the identity of that specific box as well as some metadata, for example an Internet URI. The identity of the box and the metadata could then be combined to fetch information about the specific manufacturing batch the box originated from. The increasing amount of product embedded information devices (PEIDs), which provide functionality such as self-diagnostics, could enable new services to be offered to customers, if the data produced by the PEIDs could be made available for real-time analysis.

To gain advantage from the possibilities of product data, let it be simple location information for a shipment, or more complicated information produced by a product embedded information device, a global, inter-organizational, open messaging network for data gathering must be created. The Internet is the ideal medium, as it is widely available. For the network to become widely accepted, it must be built upon existing standards, and the communication interface definitions must be open, so that all implementations can communicate with each other.

This paper has the following structure: First, the concept of the Internet of Things is presented and some requirements for different enabling technologies are outlined. Then work done in the PROMISE project regarding product data messaging systems and product embedded information devices is presented.

2 THE INTERNET OF THINGS

The Internet of Things (IoT) is a name for the phenomenon of physical objects gaining a "presence" on the Internet(Gershenfeld et al., 2004). The objects involved might be consumer products and industrial machinery, but also simple sensors spread out in the environment to form sensor networks. The "traditional" Internet consists solely of computers providing services for human users without any explicit link to things outside the electronic domain. When the

Internet is augmented with the possibility of physical objects referencing services on the Internet, and the ability for electronic services to track physical objects, and access information stored on them, we move towards a network that is no longer confined to the electronic domain but deals instead with objects and concepts that are present both in the electronic and physical worlds.

The emergence of the IoT requires a network that enables objects to reference electronic services on the Internet, as well as technologies that enable entities in the electronic domain to gain access to information stored on individual product items. At the low end of spectrum, "things", or physical objects participating in the IoT just have some kind of identification, which is machine readable in an automated fashion, such as RFID or bar code tags. These objects are "on-line" only momentarily, perhaps just by an one-way link that enables a tracker application to receive updates on the new position of the object. Examples include pallets with RFID tags in a logistics network. At the high end of the spectrum, the objects are intelligent themselves, they can gather statistical data about their usage and have means of communicating that information.

The IoT raises questions how the right to privacy can be ensured. Privacy should be taken into account already at the design phase of the product, by applying the principles of Notice, Consent, Anonymity, Locality, and Adequate security presented in (Langheinrich, 2001). The user should be made aware that data is being recorded, and recording should not happen without the user's express approval. Collected data should be anonymized, communicated using secure means, and the user should have control on the data also after it has been stored for analysis.

To enable the product items to have individual data and events associated with them, they must be identifiable in some manner. This means that a global registry of some sort must be created, with every product item having its own Globally Unique Product Identifier (GUPI)(Främling et al., 2007b). The identification is used to tie the physical object to some kind of proxy service, or product agent(Kärkkäinen et al., 2003) running on a host on the Internet. The proxy acts as a representant providing information or services on the behalf of the object.

There are several approaches to the identity-to-proxy mapping. The address (as an Uniform Resource Locator, URL) of the proxy is either directly or indirectly obtainable from the object, the so-called direct or indirect sensing described in (Kindberg et al., 2002). In the direct sensing case, the object itself contains the complete URL of the proxy service. In the indirect sensing case the identity of the product is an Uniform Resource Name (URN), which is used as a search key to find the URLs of the proxies from a central repository of identity-to-URL mappings, much in the same way that the Domain Name System (DNS) of the Internet works for converting Fully Qualified Domain Names to Internet Protocol addresses.

The ID@URI naming scheme used in the Dialog collaborative logistics system(Kärkkäinen et al., 2003) developed at the Helsinki University of Technology represents the direct sensing approach. In the Dialog system, every item has an identity of the form `identity@URI`, where the URI part is the address (URL) where the proxy service can be found, and the identity part is used to identify that particular item to the proxy. This scheme has the advantage of the inherent simplicity of the direct sensing approach, i.e. no look-up service is required. On the other hand, if the URL of the proxy service was to change at some point, it would require a re-programming of every item with the new address, a huge task. The ID@URI scheme works well in a logistics network environment, where the existence of a particular item on the network is temporally limited, but in an Internet of Things environment, where the product is present on the network for its entire lifetime, the static nature of the URI could be problematic.

The EPCglobal network developed by the Auto-ID consortium is using the indirect sensing approach to map product identities to proxy services. In this case, the GUPI is the Electronic Product Code (EPC), which defines a way to encode existing product numbering schemes for use in RFID tags and inter-organizational data exchange. The EPCglobal implementation of the identity-to-proxy mapping is named the Object Naming Service (ONS), and is an extension to the DNS system(Armenio et al., 2007). This approach has the inherent advantage that the address of the proxy can change freely. On the other hand, this scheme requires an organization that controls the allocation of id's, because every id must be unique. This could make the scheme less attractive to manufacturers, because they are then dependent on an external organization for obtaining the unique IDs.

The World Wide Article Information (WWAI) system proposed by Trackway is a peer-to-peer network for propagating requests related to product information. This approach represents the indirect sensing approach, with the responsibility of maintaining the mappings from GUPIs to product agent URLs distributed among the nodes of the network. The information providers, manufacturers, form a network of trusted nodes, which are identified by company numbers defined in existing product numbering schemes.

The manufacturer is free to assign product identities inside its own sub-space of WWAI object identifiers, the object GUPI being a concatenation of the company prefix and node-specific suffix. Access to the network of information providers is through any node of the peer-to-peer network. The node propagates the requests to other nodes on the network, and gathers the results, (product agent URLs) and presents them to the client that issued the request(Främling et al., 2007b). The service provided by the WWAI peer-to-peer network is the mapping of WWAI identifiers, (keys) to proxy URLs, (values) in a distributed data structure called Distributed Hash Table, DHT. Finding the correct value (proxy URL) for a key (GUPI) involves routing the request in the peer-to-peer network to the node that contains the relevant mapping. Similar routing overlay systems for use as application-independent middlewares have been implemented previously (see (Coulouris et al., 2005), section 10.4 and 10.5). The peer-to-peer look-up mechanism might provide better availability than the DNS-based Object Naming Service in EPCglobal, which is dependent on availability of the ONS root server maintained by EPCglobal. Internet DNS root servers have been subject to various denial-of-service attacks in the past, and the ONS root servers might also be targeted in the future.

Agent look-up services to replace or supplement the ONS have been proposed, such as the Extensible Supply-chain Discovery Service (ESDS) by Afilias(Young, 2007). The ESDS system also specifies an event mechanism for passing information about the current life cycle of a tracked object.

The solutions discussed previously are all solutions that act on the application layer. The problem of GUPIs could also be addressed on the network layer, which is the level on which the Internet Protocol works. Ordinary Internet Protocol (IP) addresses as such cannot be treated as GUPIs, even though the address space in IPv6 is sufficiently large, because an IP address uniquely determines the host's location on the network, and tells how to route datagrams to the host. An IP address is confined to a particular network, even if the underlying data link layer technology would allow geographical movement of the host. On the other hand, using the Mobile IP approach, the identity (GUPI) of the product item would be the IP address of its product agent, or *home agent* in Mobile IP terminology, which must be a permanent address that is always reachable. The agent keeps track of the current IP address where the product can be reached by receiving a notification from the product every time the address changes.

Combined with the auto-configuration features and larger address space of IPv6, and a standard application layer interface for delivering messages, such as the interface described in the next section, Mobile IP could provide a neat solution for data gathering. The drawback would be that every product item would have to have its own product agent (or home agent) with a distinct IP address, and it probably does not make sense to allocate an IP to every conceivable product item because of the associated administrative overhead.

A challenge for the design of the messaging system is how to create a single solution that spans the entire spectrum of products, from intermittently connected products with a simple smart tag that can store an ID and a limited amount of information, to products that have a powerful embedded computer. The messaging system must have some standard data model for sensor readings and events, in other words there must be a standard way for presenting information created by the products. There must also be mechanisms for gaining information about what kind of information a particular device provides.

Physical products in the physical world constitute hierarchies. For example, a car is made from several different sub-assemblies, which can be seen as separate products that sum up to make the car. To properly support hierarchies, the underlying messaging system must be able to constitute things of a higher-order from lower-order "building blocks". Consider for example the box of pills with a RFID tag. The box can be thought as an independent object in the network. But when a shipping pallet is filled with these boxes of pills, it does not make sense that every one of them is reported separately, but rather they should appear as sub-objects of the shipping pallet. These product hierarchies can be modeled by applying the Composite design pattern(Främling et al., 2007a). The messaging system to enable IoT must be able to convey product life cycle events, such as the product becoming a sub-product of an other product, or the end-of-life of a product, in addition to the gathering of product data such as sensor readings for the needs of product monitoring systems. The data model associated with the EPCglobal network, the EPC Information Services, has support for product events that are needed in inter-organizational logistics applications and inventory management. A generic event model for product life cycle monitoring is not part of the standard, nor is any mechanism for data gathering from product instances(Armenio et al., 2007).

In the following section we shall take a look at a proposed system for product data gathering that aims also to address the needs of distributing product life cycle events.

3 THE PROMISE PROJECT

The PROMISE project is financed by the European Union sixth framework programme, in which research is being conducted in to ways of gathering and processing of product data created by product embedded sensors and information systems. The aim is to create a framework of solutions that would enhance the usability of product data at various product life-cycle phases and to create sufficiently generic interfaces for data transfer, so that the solutions could be applicable in any product, let it be mass-produced household appliances or custom-built complex pieces of machinery. The PROMISE project is focusing on specifying interfaces between components and how data is stored and processed, in contrast to EPCglobal, which defines a GUPI and agent look-up infrastructure, and specifies data exchange and event interfaces focusing on consumer product logistics.

In the PROMISE world (Fig. 1), the messaging between the participants, e.g. products and the Product Data Knowledge Management systems, is done by passing messages between nodes over a messaging infrastructure defined by the PROMISE Messaging Interface, PMI. PMI uses the SOAP remote procedure call system, which uses the HTTP protocol as transport medium. The choice of SOAP over other RPC mechanisms is above all dictated by the presence of network firewalls, which typically block many types of traffic other than requests to port 80. For the messaging system to become widely adopted, it cannot require changes to network firewalls, which are usually tightly controlled by the network security staff. A defining characteristic of PMI is that nodes do not have predefined roles, as it follows the ”peer-to-peer” approach to communications. A ”full” PMI node capable of sending as well as receiving requests does have to include both HTTP client and server functionality, but a more limited node can just have the HTTP client functionality, if it is assumed that it will only send messages to other nodes. An example of such ”limited” nodes are ones associated with RFID tag readers, or generally, nodes that are unreachable from the outside because of a firewall, which periodically send product data to a product monitoring system according to a ”subscription” that is specified when the product is installed.

The current incarnation of the PMI is a collection of methods that can be called over the SOAP remote procedure call mechanism. The methods represent different operations such as a read or write of the value of a particular *info item*. The info items represent actual values such as sensor readings of a *device*. Methods also exist for querying metadata, such as the

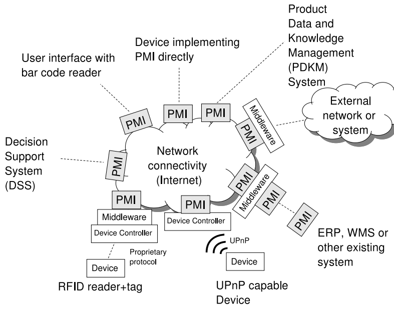

Figure 1: The PROMISE world featuring some possible participants. All participants can send messages to each other by using the Promise Messaging Interface (PMI).

list of info items that a device provides. A PMI *node* is a communications end-point in a PMI network, and manages communications for one or several devices. The parameters for the method calls are XML strings whose structure is defined by an XML schema. The XML string conveys additional request information, such as the involved device, information item, subtype of request, etc. As PMI messages are in essence XML documents transmitted over the HTTP protocol, all the technologies for making HTTP traffic secure can be applied.

In addition to ordinary ”synchronous” reads and writes, the PMI also provides a callback method for asynchronous communications. Examples of asynchronous communications include a ”subscription” read, a call to the read method with parameters that specify that the target node should not respond directly with a value, but rather send multiple responses at a specified interval. The callback method interface also provides a mechanism for nodes to send events to each other (with or without a prior subscription, subject to the particular node implementation). Indeed, the callback method is an embodiment of the Observer pattern in (Främling et al., 2007a).

The PMI implementation created at the Helsinki University of Technology, called PMI-Dialog, is based on earlier work done in the Dialog project, in which a messaging infrastructure for logistics was created. Dialog is licensed under the GNU Lesser General Public License. The Dialog architecture (Fig. 2) is centered around agents, which communicate with remote agents by sending and receiving messages. Different send and receive mechanisms are supported, one of which is the SOAP protocol. The software is implemented in Java, using the Apache Axis SOAP implementation running under the Apache Tomcat servlet container. To support PMI, a Dialog agent has been created for handling

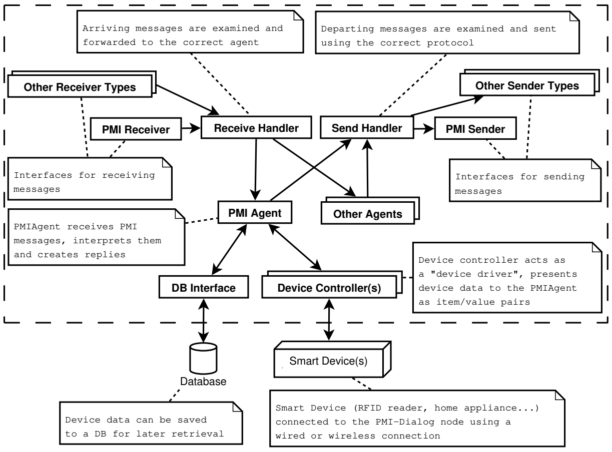

Figure 2: The Dialog architecture. The dashed line shows the node boundary. The Dialog architecture can support many different interfaces for receiving and delivering messages.

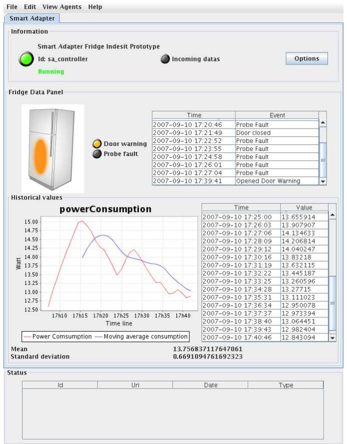

Figure 3: Information gathered from the refrigerator-embedded information device displayed on a product agent's user interface.

PMI messages.

We have been involved the development one of the PROMISE technology demonstrators, in which support for data gathering from an intelligent refrigerator's statistical data collection unit was integrated to the PMI-Dialog messaging system. The refrigerator prototype was provided by Indesit, a major European domestic appliance manufacturer, and has functionality similar to what can be expected of smart household appliances available in the near future. By integrating to the refrigerator a device capable of collecting, storing and communicating statistical data obtained from the various embedded sensors, the manufacturer plans to enhance final product testing performance at the manufacturing plant, as well as the possibility of offering a preventive maintenance service contract to customers. This latter business scenario is the most interesting from a distributed systems point of view.

The refrigerator-embedded information device was interfaced to the Dialog software using a purpose-built device controller agent, and for gathering data from the product embedded information device, a product agent was created. The agent's graphical user interface can be seen in Fig. 3.

It is assumed that in the future, product embedded devices such as the one presented here, will be using some wireless technology (eg. WLAN), and a communications software stack capable of auto-configuration, (eg. Universal Plug-and-Play, (UPnP). To support UPnP, the PMI-Dialog implementation will just have to be changed at the device controller level, by adding a UPnP-aware device controller agent.

When the first drafts of the PROMISE messaging interface were created, the Web Services/SOAP technology was chosen as the RPC mechanism. It was thought that the interface should include a multitude of methods to represent different operations, with atomic parameters. But as the project evolved, it was discovered that with a complex multi-method, multi-parameter interface it is hard to keep the interface in synch between the nodes during deployment, because if the interface changes (eg. number of parameters for a method change), different implementations are no longer compatible. As a solution to this problem the decision was made to keep the multi-method interface, but change the parameter format to XML strings, with the format governed by an XML schema. This way, even if different participants would use different schema versions, some level of compatibility (ie. "understanding" of the transmitted messages) would be preserved. This enables nodes implementing an older PMI version to continue operating with nodes implementing a newer version without the need for the newer PMI versions to include a "legacy interface" for compatibility. But a multi-method interface with "rich" XML parameters led to the evolution of the interface to the point where it was no longer useful to have several methods, but instead express different operations purely by XML requests defined by the PMI XML schema. In practice the multiple methods just caused more complex code than a single-method interface. Also, if the meaning of a message can be deduced from the message contents itself, without a need to know to which method the message was posted to, it makes adapting PMI to work on some other RPC mechanism easier, which is good for a generic message interface specification.

When we end up with a SOAP interface with just one method accepting an XML string as a parameter (and returns XML too), the question arises, that what is the point of having the SOAP layer between the web server and the application. Our current understanding is that it is useless overhead, and the alternative solution would be to just send operations expressed in XML using the POST method offered by the HTTP protocol.

The real performance "acid-test" would be to simulate an environment where the node must process messages arriving continuously from a very large group of nodes. That would represent a situation where a service company has a preventive maintenance contract with a large group of customers, or when an appliance manufacturer wishes to monitor their products for the purpose of product development. In any case, the performance of PMI implementations in general should not differ from other programs using SOAP as RPC mechanism. When PMI implementations from other PROMISE partners become available, measurements should be carried out to obtain information about the relative performance of our implementation. Measurements against other messaging systems is not relevant, as messaging systems with a similar scope do not exist.

4 CONCLUSIONS

In this paper the concept of the Internet of things was explained, and various approaches to product identity management were presented. Generic requirements for a messaging system to support the IoT were outlined. A messaging system interface specification, the PMI, created within the PROMISE project was presented, and experiences related to the implementation of a messaging system implementing the PMI were discussed. In addition to a messaging interface for exchanging data between entities in the IoT, a look-up mechanism for obtaining the address of the software agent is required. It was argued that the look-up system should be globally distributed for reasons of availability, and that a peer-to-peer architecture seems most suited for such a system.

It was also argued that dependence on a particular RPC mechanism, in this case SOAP, is not good for a generic messaging interface, and the interfaces should be defined so that any communications protocol could be used.

5 ACKNOWLEDGMENTS

The PROMISE project is funded by the European Union sixth framework (IST). The PMI has been created as joint effort between the PROMISE partners.

REFERENCES

Armenio, F., Barthel, H., Burstein, L., Dietrich, P., Duker, J., Garrett, J., Hogan, B., Ryaboy, O., Sarma, S., Schmidt, J., Suen, K., Traub, K., and Williams, J. (2007). *The EPCglobal Architecture Framework, EPCglobal Final Version 1.2 Approved 10 September 2007*. Available at `http://www.epcglobalinc.org/standards/architecture/architecture_1_2-framework-20070910.pdf`, referenced on September 17, 2007.

Coulouris, G., Dollimore, J., and Kindberg, T. (2005). *Distributed Systems, Concepts and Design*. Addison-Wesley, fourth edition. ISBN 0-321-26354-5.

Främling, K., Ala-Risku, T., Kärkkäinen, M., and Holmström, J. (2007a). Design patterns for managing product life cycle information. *Commun. ACM*, 50(6):75–79.

Främling, K., Harrison, M., Brusey, J., and Petrow, J. (2007b). Requirements on unique identifiers for managing product lifecycle information: comparision of alternative approaches. *International Journal of Computer Integrated Manufacturing*, 20(7).

Gershenfeld, N., Krikorian, R., and Cohen, D. (2004). The internet of things. *Scientific American*, 291(4):76–81.

Kärkkäinen, M., Ala-Risku, T., and Främling, K. (2003). The product centric approach: a solution to supply network information management problems? *Computers in industry*, 52(2):147–159.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., and Spasojevic, M. (2002). People, places, things: web presence for the real world. *Mob. Netw. Appl.*, 7(5):365–376.

Langheinrich, M. (2001). Priacy by design - principles of privacy-aware ubiquitous systems. *LNCS 2201, pp. 273-291.*

Young, M. (2007). *Extensible Supply-chain Discovery Service Concepts (Internet Draft)*. Available at http://www.ietf.org/internet-drafts/draft-young-esds-concepts-00.txt.