Content Creation on the Web: Mashing Up the Real World With the Internet

Vlad Trifa

Institute for Pervasive Computing, ETH Zurich
SAP Research CEC Zurich,
Kreuzplatz 20, 8008 Zurich, Switzerland
vlad.trifa@ieee.org,
Homepage: http://www.vladounet.com

Abstract. The creation and management of digital content has become more and more complex with the constant miniaturization of sensing and computing devices. With the heterogeneity of sensors, hardware and software platforms, middlewares, and protocols available, there are still many challenges that need to be solved before being able to live safely in a world with computers embedded everywhere. In particular, it is very difficult to develop systems that can be seamlessly integrated with existing infrastructures and other networks, and that could be unambiguously understood and used by other humans or machines. Therefore, we propose some insights on how one could use the Internet and its standard protocols (HTML, XML, etc) to connect heterogenous sensors/actuators networks together and to allow external user to use them easily.

1 Introduction

In the last decade, a tremendous progress in the field of embedded systems has given birth to a myriad of cheap, tiny, and generic computers (such as Arduino[1], Gumstix[2], SunSPOTs[3], etc), where virtually any type of sensors/actuators can be attached. Provided sufficient computational resources, and low-power communication capabilities, collaboration between multiple devices is possible, which opens a whole world of possible applications that could not be envisioned before. Such tools would be an invaluable help for field biologists, artists, structural engineers - if only they were simpler to program and to use. Indeed, usage of these off-the-shelf devices is limited to the few makers and geeks among us [1].

Although the state-of-the-art in wireless sensor networks does not allow these systems to comply with hard real-time constraints, the technology is widely available nowadays. Yet, most research projects are still devoted to improve the performance of existing tools rather than develop fully functional systems that hold on their promises. Why is that to use sensor networks one needs to hire

[1] http://www.arduino.cc/
[2] http://www.gumstix.com/
[3] http://www.sunspotworld.com/

an army of computer engineers who are paid to re-implement similar functions and interfaces over and over again? Simply because every project uses its own hardware and software; therefore interoperability and performance comparison is nearly impossible in such a heterogeneous ecosystem of protocols, platforms, and controllers.

In spite of the increasing popularity of the MAKE movement [2] and the flourishing of open source communities, progress is still being limited by the lack of clear, standardized, and interoperable communication protocols. For the realm of the Internet of Things to materialize (and be scalable), there is an unmet need to define a common language that can be understood by my fridge, your TV set, and her car.

Another complication is that for every project, a large amount of work is devoted to low-level programming on the one hand, and to the creation of flashy user interfaces on the other hand, which is a waste of resources that could be used by developers to focus on the application logic. Worse yet, most of these applications will never be reused again because it simply takes longer to upgrade them than to create new software from scratch.

The Internet is a stunning example of a global network of computers interoperate smoothly together, despite of the large amount of different software and hardware platforms available. For this reason, we describe in this paper how open standards that are commonly used on the Internet could be extended to operate with physical devices (e.g. sensor and actuator networks, mobile phones, etc). One of the advantages of using Web standards is that devices will be able to finally "speak" the same language as other resources on the Internet, therefore making it very easy to integrate physical devices with any other Web page.

With the large amount of devices that will be connected to this Internet of Things, manual management and configuration for each device will be simply impossible, therefore methods that will support automated discovery, configuration, and access to devices will be necessary. With this goal in mind, Semantic Web Technologies could be used to enable self-configuration of network devices and automated collaboration among them. In the following section we will discuss the important components that could be used to offer a simple system to seamlessly interconnect physical devices to the cyberspace by using Web standards.

2 Tools and methods

With the miniaturization and pervasiveness of sensing technologies, the gap between the physical and virtual world is constantly shrinking[3], however interoperability between devices from different manufacturers remains very limited because no unique standard to connect physical devices with Web applications has been adopted. We have been developing an initial prototype of such a distributed architecture with an emphasis on clear decoupling between the application logic, the physical devices, and the user interface to control it [4]. Access to a sensor network through a well-defined API would allow virtually any developer

to hit the keyboard and start coding right away while focusing on the application logic. And leave the presentation for visual designers. And the low-level programming to those who like it. Using Web Services, new interaction rituals can be conceived, where the user interface design is left to the end-user, and the code doesnt contain a hint about the presentation.

Web Services (WS) can be defined as loosely-coupled, modular, self-contained, and reusable software components that can be used to develop distributed application using standard Web protocols. We advocate the usage of Web Services for the communication between physical devices and applications, because automated discovery of new devices, metadata exchange for improved machine-to-machine communication, secure and reliable messaging, are only a few of the features offered by Web Services. Combined with a robust platform for the back-end system as is Java Enterprise, robust and flexible infrastructures for dynamic networks of heterogenous devices can be implemented (see Figure 1).

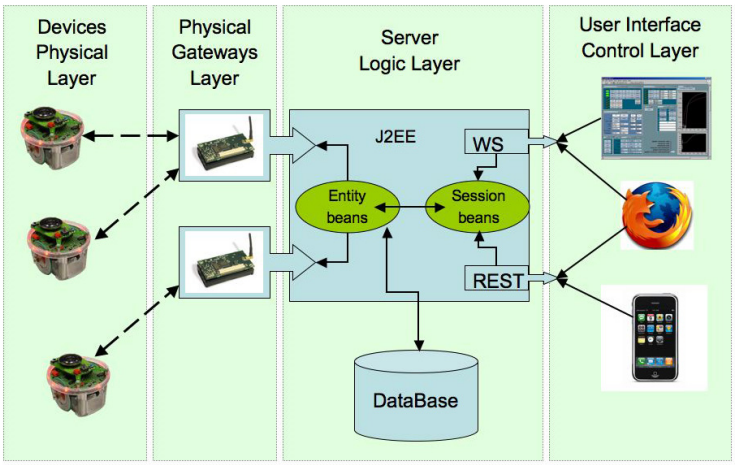

Fig. 1. The general architecture of our system is composed of four parts. From left to right: The Physical layer, consisting of the actual devices; the Gateway layer, which isthe connection between the physical devices and the system; the Logic layer, containing the back-end system that logs the data from the devices; and the Interface layer, which includes any device or interface for an external user or users. Reprinted with permission from [4].

Allowing any user to create mashed-up applications by composing them at a functional level on top of existing devices and networks, while abstracting the low-level implementation can result in significantly reduced development time. Moreover, using Web Services as a universal interface to access functionality of devices allows easy integration of networked computing devices from various constructors into different visualization and control interfaces, ranging from mobile

phones, to Web pages, to enterprise applications. In addition, flexible reconfiguration at run-time to adapt unexpected situations becomes simpler than it has ever been before.

2.1 Devices and resources on the Web

Unfortunately, Web Services are still not flexible enough to be commonly used on embedded systems, in particular their use is very limited in highly dynamic environments where new, unknown devices continuously appear, while other ones disappear. In addition, Web Services use the Internet only as a *transport protocol* instead of being directly integrated into it [5] in which case there would be no need for any additional API as . We intend to fully leverage widely adopted Web standards rather than creating new ones in order to facilitate the development of Web applications that merge physical world and the cyberspace, and are suited to work in highly dynamic environments. The idea of integrating physical devices in not quite new [6], however recent Web technologies have become enough mature and powerful to be used flexibly in highly dynamic contexts, as for example the stream feeds [7].

We advocate the development of future networked things using the "representational state transfer" (REST) architectural principle, first coined in Roy Fielding's PhD thesis [8]. The idea behind REST is that the different properties and functionalities of a devices can be viewed as resources that can be accessed using a global identifier (URI), and data can be exchanged between the different components of the network directly over HTTP with no need for an additional messaging layer (such as SOAP). The main advantage is that, this provides a much more loose coupling than Web Services, as devices are stateless, and functions can be called simply by accessing a specific URL. More information about how REST could be used to integrate physical things in into the Web can be found in [5]. A few examples of possible function calls will are shown. The return type of these function will ideally be suited to the context of the call (current user and application, with other contextual data, such as location, etc).

Retrieve information about the owner of a device (by default could returns the data using hCard microformat, see section 2.2):

```
http://mydevice/owner/
```

Get the current geo-location of the device in a format context-dependent format depending on the process or device that accesses this URL (for example global coordinates or the address of a building):

```
http://mydevice/location
```

Get the list of all italian restaurants in this city:

```
http://mydevice/location/city/restaurants/italian
```

Book a table for 4 persons at the third restaurant returned in the previous list:

`http://mydevice/location/city/restaurants/italian/3/book/people/4`

There is a need to provide a means to automatically retrieve the possible functions that can be called on a device, and in the context of Web Services this could be done by creating a machine-readable using the Web Service Description Language (WSDL). However, when it comes to tiny embedded devices with limited communication and computing power, it would be difficult to store a complete WSDL file locally. New devices that connect to the network should only send minimal information about themselves (device type), and a more powerful gateway will dynamically generate a proxy Web page for the device by using a WSDL retrieved on the homepage of the constructor for example. In addition to the static information retrieved from the WSDL file, real-time information about the device could also be embedded into the device homepage by data send directly from the device. The idea of associating a Web page for each device (be it stored locally or created on the fly by more powerful devices) is to allow any device to be indexed, searched, and accessed in a similar way that is done by current search engines. However, static indexing of devices is not possible in dynamic context where new devices (dis)appear continuously.

2.2 Semantic Web technologies

The Semantic Web is not a new technology, as it has been developed in parallel with Web Services to allow increased automation, by transforming the data published only for humans in a format that is machine readable. In order to enable digital content to be understood by software agents, one has to use a formal and unambiguous representation of knowledge, and for that languages such as OWL[4] or RDF[5] can be used, but we won't discuss these in detail here. Our idea, is to adapt these languages to be useful in the context of Ubiquitous computing, in particular to allow autonomous configuration of devices, and collaboration between them to execute high-level task with no human intervention. One could find initial efforts in this direction in [9]. However, in our research we intend to use Microformats[6], which are a set of simple and open data formats built upon existing standards, as they intend to extend existing Web, rather than building a new one. Also, the idea behind microformats is to embed simple semantic information within standard xhtml code, so that a single page can be simultaneously understood by a human and processed by a machine.

2.3 Dynamic discovery of services

The concept of discovery has been central in most distributed computing paradigms, and play an essential role in sharing and using resources on open networks. In particular, when it comes to highly dynamic environments, where many devices

[4] http://www.w3.org/TR/owl-features/
[5] http://www.w3.org/RDF/
[6] http://microformats.org/

connect and disconnect, there is a need for a robust mechanism that allows one to quickly identify, locate, and use devices or services that might not be continuously connected to the network. There are several efforts towards automated services discovery (for example WS-Discovery or UDDI), unfortunately many of them offer only very limited options when it comes to the "real world", where no centralized repository can be used and available services on the billions of devices connected to the Web can dynamically be searched on demand by the user. Several possibilities for dynamic discovery and search in the context of devices augmented with semantic information about their capabilities have been proposed, as for example mRDP [9]. To be flexible enough we identify here four different methods for finding appropriate services which need to be supported in our system:

- Listen to new devices that appear on the network, and "bookmark" them in a central repository to be reused later
- Search by matching keywords or textual information that describe static metadata (device type, available sensors)
- Browse through a tree classification based on different criteria and on the current context (location, hierarchy, etc.)
- Use a search string or query that partially describes both static and dynamic properties of devices (QoS, available battery life, network connectivity)

2.4 Ruby on Rails

We will emphasize the use of Ruby on Rails (RoR)[7] for developing different parts of the system. RoR is a framework based on the Ruby programming language for developing Web applications according to the Model-View-Control pattern. RoR offers many advantages that are particularly well suited for fast prototyping of Web-based distributed application that (REST support by design, easy integration with Web Services, content negotiation using URI templates), also because it is a simple and clean language which offers reduced development times.

2.5 Processing

In future work, we plan to explore the programming environment PROCESSING[8], developed by Fry and Reas [10], because of the endless possibilities offered in terms of visualization and integration with other programming languages and platforms. Also, PROCESSING is a amazing platform for fast prototyping and interfacing with real devices. We are investigating different connectors within PROCESSING that will allow any user to design their own interfaces for configuring and controlling various devices that are connected with TCP/IP. In particular, we plan to start the visualization of large amounts of high frequency data, and then move on towards decentralized control of devices.

[7] http://www.rubyonrails.org/
[8] http://www.processing.org

3 Discussion

It is essential to work together towards the development of a set of reusable, modular, and interoperable software components that will allow to seamlessly interconnect any type of embedded devices with back-end systems, data-bases, computing clusters, and visual interfaces. To offer a solid basis for new application developers, one should stop considering such applications as monolithic and unique-usage software that serves only for a special applications with a special hardware, and end in a cupboard afterwards. Collaboration between researchers and hackers, and sharing of simple and tested software components will allow extremely short prototyping times, and be useful to the whole community.

In this direction, we propose to reuse widely used Web standards to build an interoperable network of heterogenous devices that can be found and used both by machines and humans. Using adopted standards as opposed to creating yet other specific ones, will allow to easily make any physical device "Web-ready", thus significantly reducing the development time and costs of distributed Web applications. Besides, by using a REST approach as opposed to Web Services, we further increase the modularity and interoperability of the different contents, which result in an increased scalability and integrate better in available Web content.

References

1. Igoe, T.: Making things talk. O'Reilly (2007)
2. Gershenfeld, N.: FAB: The Coming Revolution on Your Desktop - From Personal Computers to Personal Fabrication. Basic Books (2005)
3. Ishii, H., Ullmer, B.: Tangible bits: Towards seamless interfaces between people, bits and atoms. In: CHI. (1997) 234–241
4. Trifa, V., Cianci, C., Guinard, D.: On the usage of web services for flexible and dynamic reconfiguration of robotic swarms. In: Proceedings of the International Symposium on Artificial Life and Robotics (AROB 13th). (2008)
5. Wilde, E.: Putting things to rest. Technical Report UCB iSchool Report 2007-015, School of Information, UC Berkeley (November 2007)
6. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B., Spasojevic, M.: People, places, things: web presence for the real world. Mob. Netw. Appl. **7**(5) (2002) 365–376
7. Dickerson, R., Lu, J., Lu, J., Whitehouse, K.: Stream feeds: an abstraction for the world wide sensor web. In: Proceeding of the 1st Internet of Things conference (IOT), Zurich, Switzerland (2008)
8. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, Irvine, California (2000)
9. Vazquez, J.I., de Ipiña, D.L., Sedano, I.: Soam: A web-powered architecture for designing and deploying pervasive semantic devices. IJWIS - International Journal of Web Information Systems **2**(3-4) (2006)
10. Reas, C., Fry, B.: Processing: a programming handbook for visual designers and artists. MIT Press (2007)

4 Biography

Vlad Trifa is a Research Associate with SAP Research in Zurich, Switzerland, and studies Wireless Sensor Networks in the context of Enterprise Service-Oriented Architectures. In the meanwhile, he is also a PhD student at the Institute for Pervasive Computing at the Swiss Federal Institute of Technology (ETH Zürich). Prior to that, he designed sensor networks and software to monitor and recognize tropical antbirds in the Mexican rainforest and in Californian natural reservations with the Center for Embedded Networked Sensing Group and the Department of Biology at the University of California, Los Angeles (UCLA). Afterwards, he spent a year at ATR International Research Center, in Kyoto, Japan, working on multimodal human-computer interaction, humanoid robotics, and computational neurosciences. He graduated with a Ms.C. degree in Computer Science with a concentration in robotics and artificial intelligence, from the École Polytechnique Fédérale de Lausanne (EPFL).