Demo Abstract: Plug&Play Site Management or, Why Your Solar Panel Should Be Like Your Webcam

Ettore Ferranti, Alessandro Montanari, Yvonne-Anne Pignolet, Igor Zablotchi
ABB Corporate Research
Segelhofstrasse 1K, 5405 Baden, Switzerland

1. MOTIVATION AND INTRODUCTION

Automation and monitoring systems for industrial and commercial sites (short: *Site Management Systems*) often grow organically, hence they need to be able to integrate large numbers of heterogeneous devices, based on both legacy and novel tools and systems. Currently, many standards are used in the site management field for both control (e.g., KNX, BACNet, LON) and communication (e.g., WiFi, Ethernet). Different systems are responsible for different tasks and parts of the site. Since they are usually not designed to interoperate, the site manager is forced to choose one that suits most of her needs, forgoing features offered by alternative solutions. I.e, the flexibility of the site manager is limited. Moreover, site management systems often require technical personnel for installation, calibration and configuration, and are not designed to be modified frequently to adapt to changes. Because of this, the site manager is discouraged from changing the settings of the system or from adding or updating devices, by lack of technical knowledge and by high costs. In other words, a site management system that makes adding new devices, e.g., solar panels, as easy as plugging in and using a webcam is needed.

A first step in this direction is presented in [3] showcasing a Zeroconf system for building automation. Our approach goes beyond [3] by integrating a large variety of devices and protocols. Other improvements include i) the fact that instead of using HTTP and TCP we rely on the more efficient pair CoAP and UDP, ii) that our DNS-SD implementation allows devices not only to periodically advertise their services but also to answer queries, iii) the solution proposed in [3] does not make use of a routing protocol specifically tailored for sensor networks such as RPL described below and iv) our system features a domain specific language to write applications easily and offering greater flexibility than the web application to write rules implemented for [3].

SenSys'13, November 11–15, 2013, Roma, Italy.
ACM 978-1-4503-2027-6/13/11 .

2. SELF-CONFIGURING SITE MANAGEMENT SYSTEM

To tackle these problems we propose a multi-agent architecture [1, 7] integrating different technologies to offer flexible interfaces to the system. Our solution is based on standard IP protocols and reduces the cost of deployment while improving flexibility. Hence we can take advantage of existing infrastructures and devices (e.g. a LAN inside a building). Below we discuss our recent extensions to increase the user-friendliness and flexibility of our system.

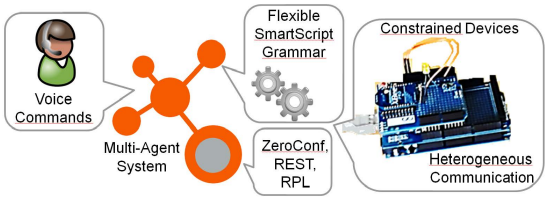


Figure 1: Extensions for more flexible site management.

Intuitive Interaction: Our approach provides the user with an innovative and simple way of controlling the system, called SmartScript [1]. SmartScript is a domain-specific language tailored for site management. It allows users to input intuitive, high-level commands to set or query the state of an appliance or groups of appliances. By design, SmartScript does not rely on the assumption that the user has in-depth knowledge of the technical details of the system. To this end, the syntax and keywords used by SmartScript, as well as device names and states were made as close as possible to natural speech. Moreover, the language features flow control through conditional statements and loops. Below, we present improvements made since the original implementation [1].

SmartScript, as well as the agent infrastructure it interacts with, were revised to add more flexibility. First, they adapt quickly to the dynamic addition or removal of devices. When such a change occurs, an agent is notified and the infrastructure related to the language is automatically re-generated and re-deployed. Thus, the user is supported by auto-suggestions for newly added devices. Second, SmartScript now has a modular grammar: the main grammar defining core functionality and a dictionary module for terminals such as keywords or device names. Therefore, it can be easily adapted to accommodate languages other than English (the current default) by implementing

dictionary modules for those languages, without changing the core grammar.

Another focus of SmartScript is ease of use. Therefore, users can now control devices by spoken commands in addition to the original version, where commands are given in written form. For this purpose, the previously strict syntax has been replaced by a less constrained one allowing for extra (non-essential) words (abundant in natural speech), arbitrary word order and approximative matching of the input to terminals. For instance, the command to lower the blinds in an office was previously *set G0147_shutter to 'DOWN'*, it can now have a more natural form: *Could you lower the shutter in g147 please?*

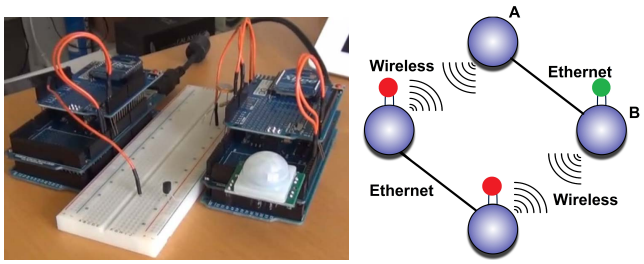
Auto-Device Discovery and Interoperability: For auto-configuration we use the mDNS and DNS-SD [2] protocols (Zeroconf) and apply a REST architecture over Coap[5]. These lightweight protocols are perfectly suited for constrained devices and allow us to keep the resource usage low.

Each device is able to answer mDNS/DNS-SD queries and exposes a RESTful API for its sensors and actuators. To discover a new device, each node publishes a service (`node._coap._udp`) identifying the IPv6 address and port of its RESTful interface. In the agent system, a specific agent browses the network for the service type `_coap._udp` and as soon as it detects a new device it performs a `GET` operation as described in [4]. This returns a list of links of the resources hosted by that device. This way, the system knows what the device can do, obsoleting manual configuration. After the discovery phase, the system can interact with any device using standard HTTP-based methods `GET`, `POST`, `PUT` and `DELETE`. Disappearing devices are handled by the DNS-SD protocol automatically as well. Moreover, the RESTful interface guarantees a homogeneous access to the sensors and actuators hosted by the devices and facilitates interoperation with other web technologies.

Heterogeneous Communication: Often the high cost for the installation of cables and the configuration of the communication infrastructure is an obstacle for brown field site management. Furthermore, adaptable links between components are an essential feature of a flexible distributed system. As a consequence, we extended the networking stack of the embedded Contiki OS, in order to offer flexibility regarding the communication technologies used and to satisfy the constraints of Low Power and Lossy Networks. More precisely we modified the stack to allow the IPv6 routing protocol RPL [6] to use more than one communication interface, e.g., wireless and Ethernet. RPL is a self-healing Distance Vector protocol that can be optimized for various applications by changing the objective function it uses to compute suitable paths to border routers. It ensures that new devices join an existing network automatically and discover communication paths to and from any node in the network.

3. DEMONSTRATION DESCRIPTION

For the demonstration, we set up a mini office-like environment, complete with various sensors (e.g., brightness, movement, temperature) and actuators (e.g., light switch, shutter motor) and showcase the main features of our system. In particular, we show how the system can be controlled by issuing voice commands, how plugging devices in and out is automatically recognized, and how communication seamlessly switches between wireless and wired links.



a) b)

Figure 2: a) Setup for service discovery and REST demo, with two Arduinos equipped with sensors b) Schema of RPL demo with four Arduinos with 802.15.4 and Ethernet shields, two cables, sending a message from A to B over one of the two paths

Part 1 - SmartScript + Voice Recognition We showcase the voice command capability. Users can control the appliances in the miniature office environment by issuing commands with a headset connected to a computer. Users are encouraged to experiment with different wordings to illustrate the flexibility of the interface.

Part 2 - Service Discovery + REST We show how to realize a flexible and auto-configurable system with standard protocols and technologies. New sensor and actuator devices will be turned on and their presence will be detected automatically by the system. The new devices will appear in the Web user interface along with all the information they carry. Moreover, the newly added devices become addressable in SmartScript commands, therefore it is possible to use them in the language without any manual intervention.

Part 3 - Reliability over Heterogeneous Links We demonstrate how RPL switches paths depending on the availability of reliable links. Our setup (Figure 2.b) consists of four nodes, equipped with an 802.15.4 shield and an Ethernet (ENC28J60) shield, forming a ring connection (using MAC address filtering to ensure that no other messages are received). When transmitting a message from node A to a node B connected by Ethernet, RPL chooses this direct path, as can be seen by a blinking LED on node B. If the cable between them is unplugged, the LED on the other nodes start blinking, signaling that they are forwarding packets.

4. REFERENCES

[1] D. Adolf, E. Ferranti, S. Koch. SmartScript-A Domain-Specific Language for Appliance Control in Smart Grids. *IEEE SmartGridComm'12.*

[2] S. Cheshire, M. Krochmal. Multicast DNS *(IETF RFC 6773)*, DNS-Based Service Discovery *(IETF RFC 6762), '11.*

[3] L. Schor, P. Sommer, R. Wattenhofer. Towards a zero-configuration wireless sensor network architecture for smart buildings. *BuildSys'09.*

[4] Z. Shelby. Constrained RESTful Environments (CoRE) Link Format, *IETF RFC 6690, '12.*

[5] Z. Shelby, K. Hartke, C. Bormann. Constrained application protocol (coap), *draft-ietf-core-coap-18, '13.*

[6] T. Winter, P. Thubert, A. Brandt, T. H. Clausen, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. *IETF RFC 6550, '12.*

[7] D.-Y. Yu, E. Ferranti, H. Hadeli. An intelligent building that listens to your needs. *ACM SAC '13.*