Poster: Smart Buildings, Sensor Networks, and the Internet of Things

Bernhard Firner[1]
bfirner@winlab.rutgers.edu

Robert S. Moore[2]
romoore@cs.rutgers.edu

Richard Howard[1]
reh@winlab.rutgers.edu

Richard P. Martin[2]
rmartin@cs.rutgers.edu

Yanyong Zhang[1]
yyzhang@winlab.rutgers.edu

[1] WINLAB, Rutgers University
671 Route 1 South
North Brunswick, N.J. 08902-3390

[2] CS Dept., Rutgers University
110 Frelinghuysen Rd.
Piscataway, NJ 08854

Abstract

In contrast to traditional sensor networks, the "Internet of Things" focuses on interactions between humans and physical objects rather than on sensing and reporting low level information. While several middle-ware systems have been created to simplify the task of managing and aggregating data from multiple sensor networks that use different hardware and software, management of the data is not sufficient to build an Internet of Things.

To build a smart home or office application, information from sensors and existing infrastructure, such as networked power supplies and printers, need to be associated with physical objects. As low-level data is processed, it can be fed back into the system to support higher-level results. In this paper we introduce *Octopus*, an open-source, dynamically extensible system that supports data management and fusion for Internet of Things applications.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.2.13 [**Software Engineering**]: Reusable Software—*Domain Engineering*

General Terms

Design, Management

Keywords

Sensor Network, Internet of Things, GRAIL

1 Introduction

The Internet of Things [4] envisions a data-rich world where the status of common physical objects is always available over a network. People will interact with an Internet of Things, not for low-level data such as the current temperature and dew point, but for high-level semantic information - for instance it is raining and your window is open. Realizing this vision requires a general-purpose system architecture that can dynamically incorporate new data sources and data processing software. The system must be general enough to support infrastructure found in many buildings, such as network printers and network controllable power supplies, while also supporting newly deployed sensor networks. Realistic deployment scenarios also require systems that span multiple administrative domains, such as across multiple university campuses or between wards in a hospital.

1.1 "Internet of Data" or "Internet of Things"

Managing data from large numbers of sensors is a challenging problem by itself so there are several solutions that provide interfaces to query sensors and combine sensor data. One example is the Global Sensor Networks (GSN) middleware [2], which uses a virtual sensor abstraction to hide the specifics of sensor hardware and present a single interface for querying and subscribing to data. Octopus also supports sensor abstraction through an aggregation layer, as shown in Figure 1. Through this layer Octopus can use GSN or similar systems as data sources to reuse existing work.

Sensor abstraction is found in many systems but Octopus differentiates itself through its *world model*, which bridges the gap between an internet of data and the Internet of Things. In this world model all data must be associated with an object and all data analysis modifies the world model, allowing the system to build layers of more complex semantic knowledge. Relationships between objects are expressed through a hierarchical URI naming scheme. This grouping allows users to search, query, and subscribe to information by region and context, simplifying processing of low-level data into higher-level semantic information.

2 Data Management and Analysis

Anticipating every possible sensor and application is impossible so Octopus provides network API that supports adding and removing analysis modules, called *solvers*, in a running Octopus system. This means that as requirements change new solvers can be introduced to a running system to process new sensor data or perform new data analysis.

SenSys'11, November 1-4, 2011, Seattle, WA, USA.
ACM 978-1-4503-0718-5/11/11

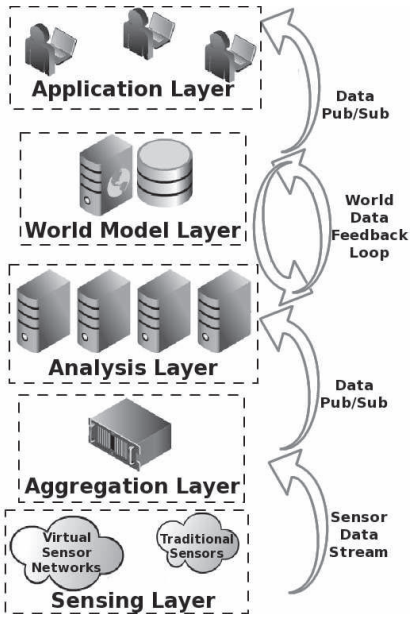

Figure 1: The layers of the Octopus stack with data publish/subscribe relationships. Multiple tiers of solvers run in the analysis layer to process lower-level world model information into higher-level information.

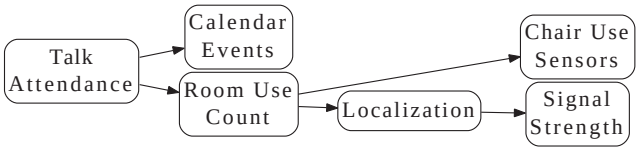

Figure 2: An example set of sensors and solvers with data dependencies.

Solvers can subscribe to data from the aggregation layer, the world model, as shown in Figure 1, or from existing infrastructure, such as networked power switches or online weather reports. When solvers subscribe to attributes of URIs in the world model they create a feedback loop where low-level data bubbles up to affect high-level information.

An example using a talk attendance solver appears in Figure 2. In the example, a talk attendance solver uses talk schedules and room use counts from the world model. The room use count solver in turn uses data from chair use sensors and localization results (for chair locations).

2.1 Data Typing and Constraints

Inspired by the popular HTTP protocol, the Octopus system uses MIME types to document data types in the world model. Many data types exist in current MIME standards, and new types that are required for the Internet of Things can be added and documented in a standard way.

The world model consists of UTF-16 Uniform Resource Identifiers (URIs) [1] with associated sets of attributes that can be searched, accessed, and updated. URIs correspond to real-world objects and events to ensure that Octopus manages an Internet of Things and not an Internet of Data.

Octopus URIs are hierarchical, like directory structures, to implicitly provide object relationships. A building named "Brick House" owned by a company named "3 Pigs" could be called "3 Pigs.Brick House" and the living room in the house could be called "3 Pigs.Brick House.living room".

3 System Clients

Clients in the application layer of the system can connect to the world model and can request or subscribe to the current or historic state of URIs. Clients can also search through URIs and their attributes using glob regular expressions - the string "company.building.*" would return the URIs of everything in a building. Clients can also search for attributes, for instance to obtain a list of objects with temperature data.

Clients are free to present data in any format, allowing Octopus to support different kinds of interfaces. Currently there are GUI clients for the desktop and browser and event notifications through twitter, text messages, and e-mail.

4 Deployment

Our system is currently deployed on two different University campuses. Our sensing layer is mostly pipsqueak radio tags [3], chosen for their small size, cost, and lifetime (1 year on a coin cell battery). Octopus supports other sensing hardware, such as Wi-Fi transmitters, but the pipsqueak's longevity makes it more suitable for long-term deployments.

Currently we collect various sensed data, such as temperature, wireless signal strength, and power consumption in wall outlets. Magnetic switch sensors detect when doors are open or closed and physical switch sensors detect when chairs are used. The deployment also pulls data from existing systems, such as printers and email event announcements.

Once the system was deployed it was easy to add both additional sensors and additional analysis modules. For example, the initial deployment only tracked sensor mobility and location but temperature sensors, power monitoring sensors, switched sensors, and new solvers to use the new data were all added incrementally without interrupting the system.

5 Conclusion

Octopus has been actively used, developed, and expanded for over 6 months and demonstrates a practical system for collecting and analyzing multiple sources of information for several Internet of Things applications. Octopus' flexibility has allowed us to respond quickly to user feedback and update or expand the system's capabilities without interrupting existing services. Future work will involve larger deployments of the system at additional locations, and the expansion of the types of data the system handles.

6 References

[1] RFC 2396: Uniform Resource Identifiers: Generic Syntax, 1998. `http://www.w3.org/TR/xmlschema-2/#RFC2396`.

[2] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In *Mobile Data Management (MDM)*, 2007.

[3] B. Firner, P. Jadhav, Y. Zhang, R. Howard, W. Trappe, and E. Fenson. Towards Continuous Asset Tracking: Low-Power Communication and Fail-Safe Presence Assurance. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, June 2009.

[4] C. D. Gershenfeld N, Krikorian R. The Internet of Things. *Scientific American*, 291(4):76–81, Oct 2004.