Self-Controlled Production: Forecasting with Smart Sensor Technologies

Wolfgang Maass[1] and Shahd Zahran[2]

Chair in Business Administration, Faculty of Law and Economics, Saarland University
P.O. 15 11 50, 66041 Saarbrücken, Germany

Abstract. Sensor-based architectures are currently simple data collectors that feed-forward to central data processing units. With huge amounts of sensors data networks are threatened by congestion of low-quality data and noise. Therefore we present a Smart Sensor Architecture that enables semantic annotation of sensor data near the local environment of a sensor and processing of data by sophisticated algorithms. We show technologies by which the Smart Sensor Architecture is realized and discuss how the behavior of smart sensors can be changed at run-time. Our approach was deployed in a plant irrigation scenario. Initial results of the evaluation are presented.

Keywords: smart sensors, semantic annotations, forecast algorithms, low-cost hardware

1 Introduction

Sensor technologies are increasingly used in domains, such as retailing, production, and home automation. With 120 billion sensors sold in 2011 and estimating that each sensor issues 10 Byte per minute, sensor technologies already generate 560 PetaByte per year with approximately 1 MegaWatt electricity consumption as a lower bound. Some even estimate 1 trillion units of wireless sensors will be sold in 2022. Discounting the intrinsic fuzziness of these kinds of estimations, it is nonetheless clear that sensor technologies will generate enormous amounts of data, aka big data. In contrast to discrete transactional data, sensor technologies may also generate continuous data streams with large percentages of noise. The current system design, based on client-server approaches, assumes a central data center that receives and processes all sensor data. This will put a lot of strain on the networking systems and processing power of central data systems. Assuming that abovementioned estimates are lower bounds and that growth rates in data created by sensors and pure number of sensors deployed, sensor technologies will easily overcharge any data processing network in general and the Internet in particular.

Based on these assumptions we investigate smart sensor technologies by which software can be pushed at runtime to leaf nodes near to sensors depending on processing needs. By doing this data will be decentrally processed by software environments and only enriched data with assured quality is send to higher-order nodes and finally to central data processing systems. This

[1] wolfgang.maass@iss.uni-saarland.de
[2] zahran.shahd@gmail.com

approach resembles the approach taken by the ATLAS experiment for the Large Hadron Collider (LHC) at CERN where 300.000 MByte/s are reduced to 300 MByte/s by various event filters [1].

In this paper we describe the Smart Sensor Architecture that combines sensor technologies with flexible software approaches that even support software push at runtime. We demonstrate our approach and technologies by the plant irrigation domain. Our approach combines various technologies that are important for future Information Systems that leverage sensor technologies. Most of all we use semantic technologies for processing sensor data and analytical processing for computing forecasts on current and historical sensor data. Semantic technologies are based on logical reasoners while analytical processing is executed within a Javascript environment. The first, ie. the logical reasoner, supports adaptation at runtime by injecting logical axioms, rules, and logical facts. On the other hand Javascript based on nodeJS supports code injection at runtime. All these technologies are cutting edge but by integrating everything on integrated hardware systems, i.e. Raspberry PI, connected to various sensors. In our courses, students are introduced to these complex technologies by practical cases, such as the plant irrigation case discussed in the following.

2 Smart sensor architecture

The basic structure of our architecture is a leaf node that consists of three modules: 1) semantic computing module (SCM), 2) an object-oriented / functional computing module (FCM), and a 3) leaf node management module. Leaf nodes can exchange data with other leaf nodes so that a network of leaf nodes can emerge. Leaf nodes can also transfer processed data to master nodes for higher-order processing of integrated data from a large set of leaf nodes. For instance, we are testing master nodes that process integrated data by applying time series analysis based on map-reduce algorithms and key-value data representations [2] – in our case based on mongoDB .

Each leaf node is assigned to at least one physical object, identified by an ID. This allows building reference between data and physical origins. A leaf node receives data from various connected sensors that provide data about the object status or its context. The processing modules of a leaf node, ie. SCM and FCM, take dedicated tasks, assigned by the leaf node management module. The SCM processes all incoming data and creates RDF representations for each data point [3] and uses in particular the expressivity of OWL [4] that enables logical reasoning by DL reasoners [5]. Thus, it is explicitly represented that each data point is associated to a particular object ID. Furthermore RDF-based data representations can be processed by quality assurance procedures executed by the logical reasoned – in our case we use the Pellet reasoner [5]. This approach resembles an approach sketched by [6] recently elaborated by [7].

The FCM accesses the data store via a SPARQL endpoint [8] provided by the SCM via a JSON intermediate data representation. The FCM can carry a wide variety of processing capabilities. In our case, FCM is used for analytical processing, and in particular processing forecasts based on historic data. Results are send back to the SCM and stored in the RDF repository.

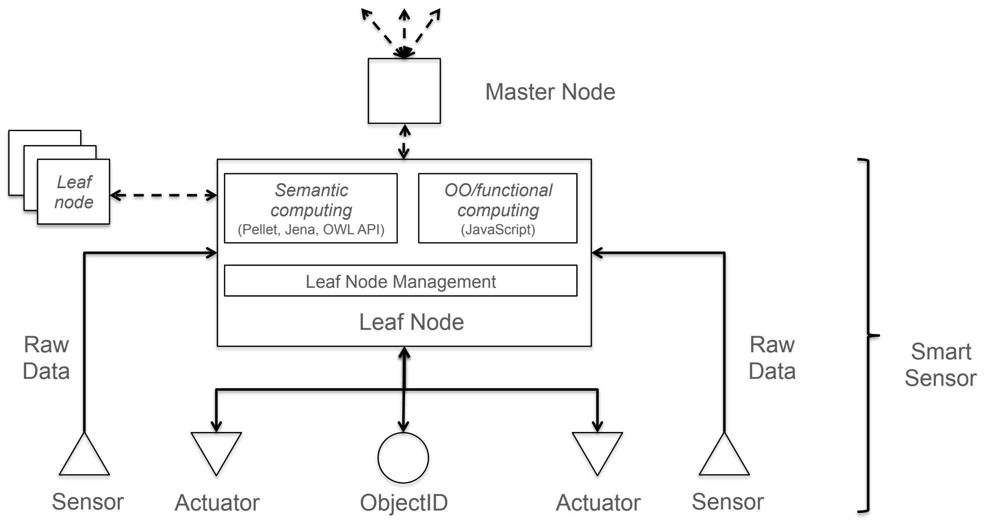

Figure 1: Smart sensor architecture

Finally, each leaf node can create behavior by triggering actuators according to processing results.

3 Example HappyPlant

3.1 Technical realization of the HappyPlant smart sensor model

We have deployed the smart sensor model with a plant irrigation setting. In one pot we had three sunflowers controlled by a HappyPlant Smart Sensor System (HP3S) while the control setting consisted of three sunflowers and was irrigated by a professional botanist. HP3S was implemented on a Raspberry PI board connected to a PhidrgetBoard with three sensors for a) soil moisture, b) temperature, and c) humidity (cf. Figure 2). Additionally we automatically took pictures of both plant systems for cross-checking plant health status by botanists. The SCM was implemented by an OWL API as the ontology manager on OWL/RDF data representations. Additionally we used a Jena 2.0 framework for posing SPARQL queries and translating results into JSON data representations that can be digested by the FCM. The FCM was realized in JavaScript based on a nodeJS platform.

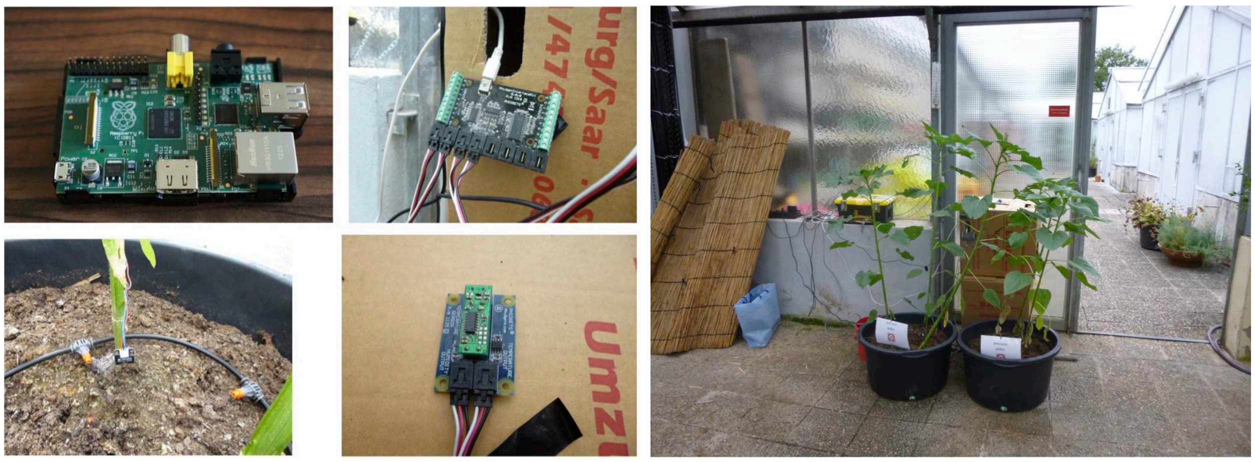

Figure 2: Hardware components

3.2 Data representation and semantic annotations

The SCM stored sensor data in an OWL representation according to a basic ontology (cf. Figure 3). By doing this, semantically annotated sensor data can be exchanged with other smart sensors that might use different ontological representations. With transfer ontologies, smart sensors have a means for integrating and leveraging data from other leaf nodes (e.g., [9]).

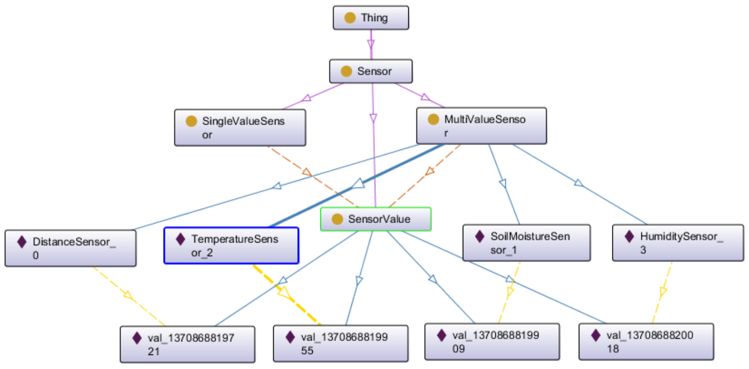

Figure 3: Sensor data ontology

3.2 Forecasting with sensor data

We used the FCM for implementing a forecasting algorithm with exponential smoothing with seasonal adjustment [10]. Seasonal adjustment was used for modeling different characteristics of day and night shifts while exponential smoothing supported dampening extreme values, in particular large temperature peaks. In the following we sketch the key elements of the forecasting algorithm.

- $I_t = \frac{A_t}{\bar{A}}$ with A_t as current entry sensor actual value, A average value of all actual sensors' values in the first season.

- $S_t = \alpha \frac{A_t}{I_{t-L}}$ with S_t smoothed value of the current entry, α smooting factor, and I_{t-L} seasonal index of the current entry.
- $I_t = \gamma \frac{A_t}{S_t} + (1-\gamma)I_{t-L}$ with γ smoothing factor and finally
- $F_{t+1} = (S_t)(I_{t-L+1})$ with F_{t+1} the forecast value for water supply for time period $t+1$.

The main actuator was water supply for the plants. To control the water supply, we developed a simple infrared-based sensor system that measured the amount of water and stopped when a particular amount was given.

5 Evaluation

We conducted a first feasibility study of the HP3S over two weeks. The hardware platform was very stable while the moisture sensor showed lack of precision (cf. Figure 4). Temperature and humidity sensors showed a good fit with the domain. Also the water supply sensor based on infrared was not precise enough. Therefore we replaced it in the second prototype by a digital scale with a much higher precision.

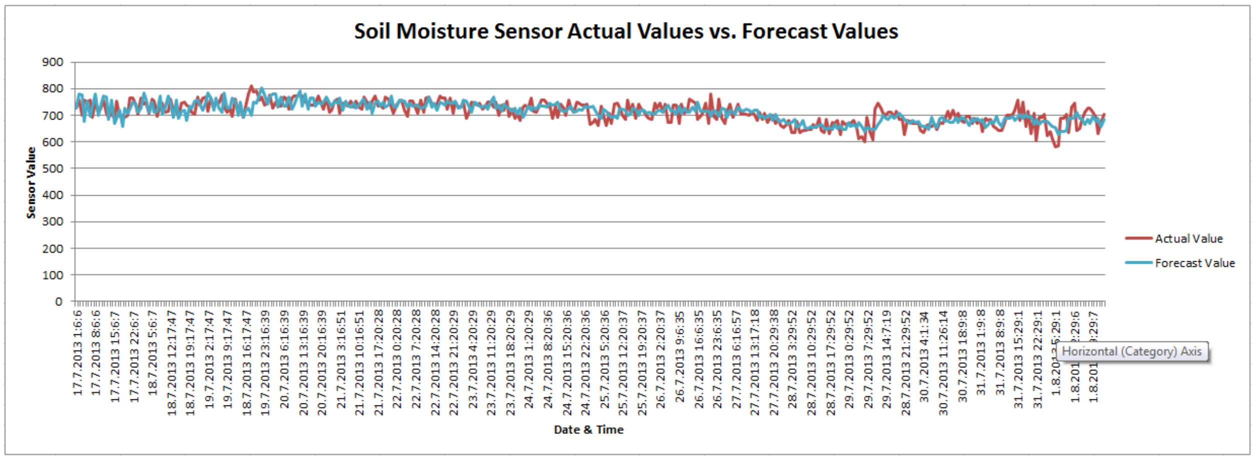

Figure 4: Soil moisture time series

More important, we found that the implementation of the leaf node model based on Raspberry PI [11] and PhidgetBoard (http://www.phidgets.com), worked very well with all components that implemented the SCM and FCM. Even though that SCM and FCM demand a lot of resources and processing power, the Raspberry PI board was slow compared to a notebook but fast enough for all our data processing tasks. The OWL API stored and retrieved OWL data representations very successful and the FCM implementation by JavaScript on nodeJS worked fine as well. Therefore we were able to show a first version of a smart sensor that integrates semantic processing and functional programming capabilities on a simple, low-cost, and wirelessly connected hardware realization. We even tested power supply by a small solar-panel that allowed to operate this system for about a day.

For the second iteration we replaced the soil moisture sensor by a high-precision sensor. First evaluations show better results.

4 Conclusion

We have presented a smart sensor architecture consisting of a SCM and FCM. Both modules can be reconfigured at run-time by leveraging special features of semantic data processing and of JavaScript runtime environments. We have described an implementation of the Smart Sensor Architecture by the HP3S for the plant irrigation domain. First evaluations should deficits in precisions of sensor technologies but in general we were able to validate our approach. We almost achieved real-time data processing and analytical support for self-controlled irrigation by the plant.

References

1. Dos Anjos, A., Armstrong, S., Baines, J.T.M., al., e.: Muon Identification with the Event Filter of the ATLAS Experiment at CERN LHC's. (2005)
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM 51, 107-113 (2008)
3. McBride, B.: The Resource Description Framwork (RDF) and its Vocabulary Description Language RDFS. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 51-66. Springer, Berlin (2004)
4. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.A.: OWL Web Ontology Language Reference. (2004)
5. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. http://www.mindswap.org/papers/PelletJWS.pdf (2006)
6. Sheth, A., Henson, C., Sahoo, S.S.: Semantic sensor web. Internet Computing, IEEE 12, 78-83 (2008)
7. Calbimonte, J.-P., Jeung, H., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. International Journal on Semantic Web and Information Systems (IJSWIS) 8, 43-63 (2012)
8. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C (2007)
9. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Berlin (2007)
10. Fitzsimmons, J.A., Fitzsimmons, M.J.: Service management: Operations, strategy, and information technology. (2004)
11. Upton, E., Halfacree, G.: Raspberry Pi User Guide. John Wiley & Sons (2012)