A New Internet of Things Architecture with Cross-Layer Communication

Alberto Messias da Costa Souza
Cruzeiro do Sul University
São Paulo, Brazil
Email: linuxstring@gmail.com

José Roberto de Almeida Amazonas
Escola Politécnica of the University of São Paulo - USP
São Paulo, Brazil
Email: jra@lcs.poli.usp.br

***Abstract*—This paper describes a new Internet of Things architecture with cross-layer communication. This architecture shows the importance of cross-layer communication between physical, middleware and application layer and explores this functionality to improve the decision make process. Our implementation extends the LinkSmart Middleware, aggregates pattern recognition services to the middleware and introduces a cross-layer communication structure and associate parameters. The cross-layer communication permits to modify the behaviour of the middleware and physical layers by means of control functionalities implemented in the application layer. In this paper, we validate the cross-layer communication in a new IoT architecture with pattern recognition services. We develop an application, that used a real database.**

Keywords-***Internet of Things; Cross-Layer Communication; IoT Middleware; Pattern Recognition.***

I. Introduction

The Internet of Things (IoT) refers to the next generation of the Internet [1], which interconnects trillions of nodes, represented by small ubiquitous devices, equipped with sensors, interconnected Web servers, supercomputers or clusters [2].

This revolution will not affect only the amount of information, but also their quality. Many small processors embedded in objects will be integrated in daily routines.

According to Smith [3], data management in IoT is a crucial aspect. Considering a world of interconnected objects which constantly exchange many kinds of information, the volume of generated data and involved processes makes the data management to become critical. New services to process and analyse the massive data generated by the communication between devices will be needed. These services will need to have open interfaces and will have to be able to provide a simple integration between many applications.

The pattern recognition mechanisms are implemented in the lower layers of the IoT model, namely the physical, middleware and services layers [4]; therefore, and these capabilities need a new architecture in which communication between lower layers and the application layer is enabled. This communication needs a contextualized control implemented in the application layer. In this paper, we introduce this new IoT architecture and its implementation emphasising the communication aspects between layers.

This paper is organised as follows: after this brief Introduction, Section II introduces IoT concepts and IoT middlewares. The proposed architecture and its implementation details are shown in Section III. Conclusions and future works are presented in Section IV.

II. Background

This section introduces the background definitions about IoT and IoT middlewares.

A. Internet of Things

As stated in [5], "the IoT is a global network infrastructure, linking physical and virtual objects through the exploitation of automatic identification, data capture and communication capabilities. This infrastructure includes the existing and evolving Internet and other network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent federated services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability, actuation and control ".

Figure 1 illustrates the inclusive model as proposed by the CASAGRAS EC funded project [5].

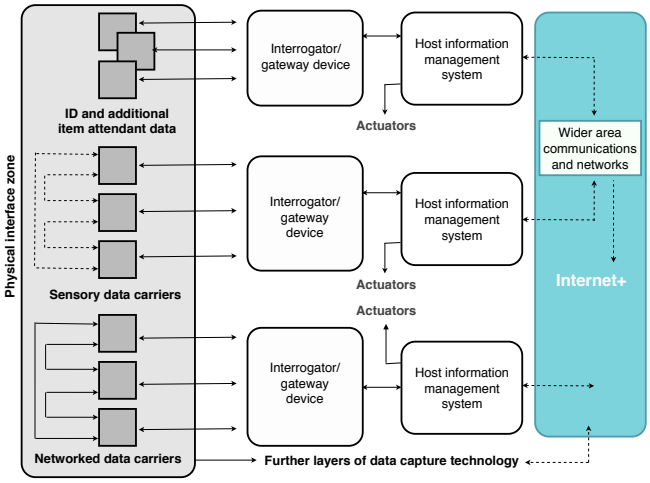

Figure 1. CASAGRAS inclusive model [6].

According to the CASAGRAS inclusive model, a real-world object has its identification ID and associated information stored on some kind of item-attendant data carrier as, for example, on a RFID [7] tag. It is important to realize that the identification technology is not restricted to RFID. Biometry and bar codes are other examples of ID technology that can be employed. The information is retrieved from the object by means of an interrogator that acts as a gateway device and sends it to be stored in a host management system. The Internet is used both to allow access to the retrieved information and to search for further information and associated applications and services. The end result is that an action will take place either displaying new information and/or acting upon the object and/or the environment [6]. The whole process is context-aware and the final action depends on the object itself and its present status in the current environment.

B. Internet of Things middleware

As shown in [8], there are many middlewares, which are defined as software systems that provide an abstraction layer between the operating system and development applications environments in the context of pervasive computing, whose focus is to provide an useful and abstracted suite of procedures that can deal with the heterogeneity of devices and contexts of information.

For this work, it is of relevance the IoT middleware *Network Embedded System for Heterogeneous Physical Devices Middleware in the Distributed Architecture - (HYDRA)* created by the FP6 IST [9], which started in July 2007 and finished in December 2010.

As observed in [10], the first objective of the Hydra Project was the development of a software middleware based on the Service-Oriented Architecture (SOA), in which the communication occurs transparently between the lower layers.

The framework should support centralised and distributed architectures, security and trust, and model driven applications development. One of the framework's development premises was its applicability in current networks and novel network models with interconnected devices that operate with reduced computational power, energy and memory capacity.

The resulting product of this project was called LinkSmart middleware, a name that will be used to refer to the developed middleware from this point onward.

The elements of the LinkSmart middleware are placed between the application and physical layers. The physical layer is related to network communication resources, while the application layer contains modules related to the management of information flow, user interface, application logic and configuration details. Between the two layers is the LinkSmart middleware, consisting of three sub-layers, network, service and semantics, each responsible for specific functions and purposes [9].

The LinkSmart Middleware funcional structure is divided into two parts: (1) Application elements describe components deployed on hardware which is performance-wise capable of running the application that the solution-provider creates. This means these components are meant to be run on powerful machines; and (2) The device elements describe components deployed based on LinkSmart Middleware. These components are running on small devices which have limited resources [9].

III. Cross-layer communication in IoT architecture

This section describes the communication model of the new proposed IoT architecture with cross-layer communication. In our proposal, we insert pattern recognition services in the IoT architecture, specifically in physical and middleware layers. The focus of this paper is the communication model of the new IoT architecture and the cross-layer communication implemented between the physical, middleware and application layers.

To implement this new IoT architecture, we extended the LinkSmart middleware by creating a new pattern recognition services implementation and an abstraction of the following algorithms: outlier detection, values estimation and clustering. This solution is able to apply these algorithms to many kinds of environments and devices. New applications can retrieve contextualized information from the middleware rather than receiving raw data from either the physical or middleware layers.

Figure 2 shows the proposed change in the layer structure of the LinkSmart middleware.

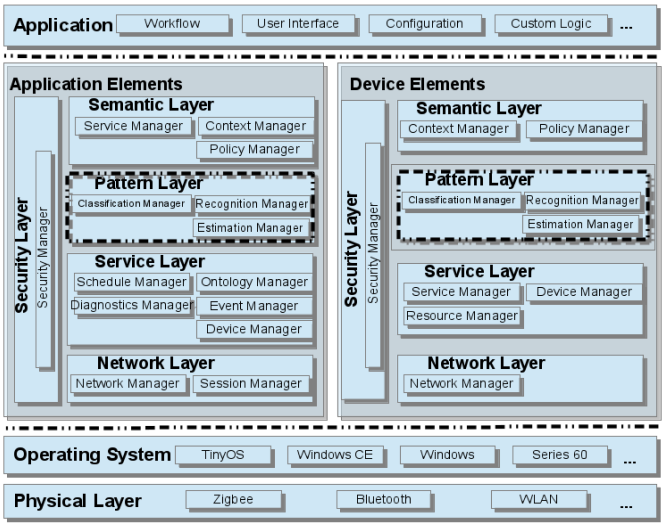

Figure 2. A new layer structure of the LinkSmart middleware, adapted from [9].

In Figure 2, it is shown a new box called *Pattern Layer*, highlighted by a dashed rectangle. This new layer has three managers: classification, recognition and estimation, which implement the pattern recognition functionalities.

The value estimation, behaviors recognition, classification and outlier detection algorithms [11] [12] contribute to network traffic minimisation in the IoT context, as the upper application layer will not receive raw data but pre-processed information by the LinkSmart middleware pattern services.

 ISBN: 978-1-61208-422-0

These algorithms have been implemented with a distributed processing data architecture using the Big Data technology [13] [14].

In our implementation we used the following techniques: linear regression for values estimation [11], k-means algorithm for clustering [11] and contextualize for retrieved values from sensors and others devices. To detect outliers we used clustering distance [15] [16] [17].

The values estimation and outlier detection algorithms have also been implemented in the physical layer.

Figure 3 shows the implemented architecture.

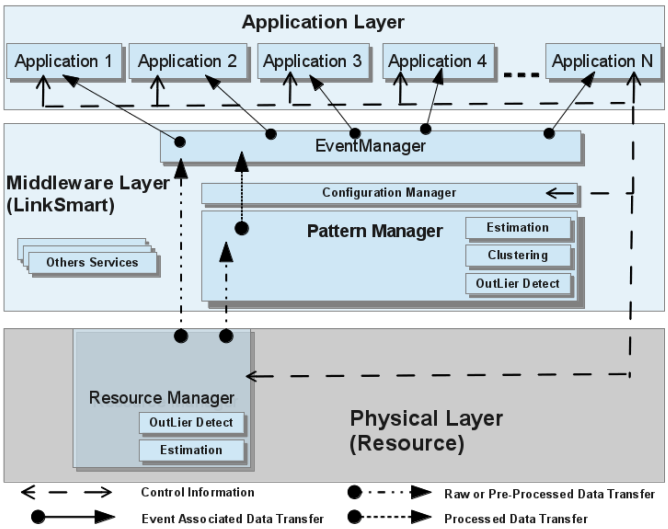

Figure 3. Implemented architecture

The most important aspect of this implementation represented by the Figure 3 is a new pattern recognition module inserted in the LinkSmart middleware. This implementation follows the IoT-A reference model [18], and has the following layers:

- Physical layer: hosts the resource layer represented by sensors and smart objects. The resource manager box represents the driver or software responsible to connect with the LinkSmart and to send raw data or pre-processed data by the value estimation and/or outlier detection algorithms. If data generated by a physical resource is processed in this layer, the resource manager informs the configuration manager in the LinkSmart middleware or directly the application. Both the configuration manager and the application can change parameters in the resource manager to disable the pre-processor so the resource manager proceeds to forward the raw data. Note that from this layer data can flow to the middleware or to the application layer.
- Middleware Layer: it is represented by the LinkSmart Middleware, modified in this implementation by the inclusion of the pattern recognition and configuration managers, the event manager which is quite important in this architecture, and others services that are less relevant in the present case. The pattern recognition manager implements three services: value estimation, clustering and outlier detection. The configuration manager enables the applications or the resource manager to configure parameters in the pattern recognition manager, defining when the algorithms are run or how much data must be stored, enabling or disabling pattern recognition services, cleaning the stored data. After processing data the pattern recognition services can send the contextualized information to the event manager which by its turn will detect and send the new events to its clients. The event manager tackles the scalability issue as it can be in charge of one application client or millions of them. The event manager can be directly accessed by the resource manager which in this case does not activate the pattern recognition services. This feature can be chosen by the application designer and later changed in the configuration manager.
  This layer implements the cross-layer communication as it can receive or send configuration parameters both from the resource manager and applications. However, the most relevant aspect of cross-layer communication is the fact that raw data can be open, processed and interpreted in this layer which otherwise should be a specific function of the application layer according to the ISO/OSI layer model [19].
- Application Layer: the application layer is represented by client applications and configuration applications. The client applications receive the events from the event manager, either raw data or processed information. The configuration application is responsible to configure parameters in the configuration manager or send parameters to the resource manager to control its behaviour or the pattern recognition services activation/de-activation. This bidirectional communication along the associated configuration feature represents the cross-layer communication.

The cross-layer communication is a requirement of this architecture and is implemented as previously described. The proposed architecture enables a fully distributed implementation as the information can be processed at each physical node, or at network access points, or at middleware service delivery nodes, or at application nodes. The cross-layer communication of our proposed architecture enables this distinctive IoT feature.

A. Implementation aspects

We created an object-oriented structured programming middleware implementation. Figure 4 illustrates the specific cross-layer communication structure.

Figure 4 shows the class diagram that represents the developed object-oriented programming structure. There are two classes that represent the cross-layer parameters: (1) the interface *CrossLayerParameter* that creates a main structure of the cross-layer parameters. It can be seen the contract of abstract methods *setLayer* that is responsible to define

 ISBN: 978-1-61208-422-0

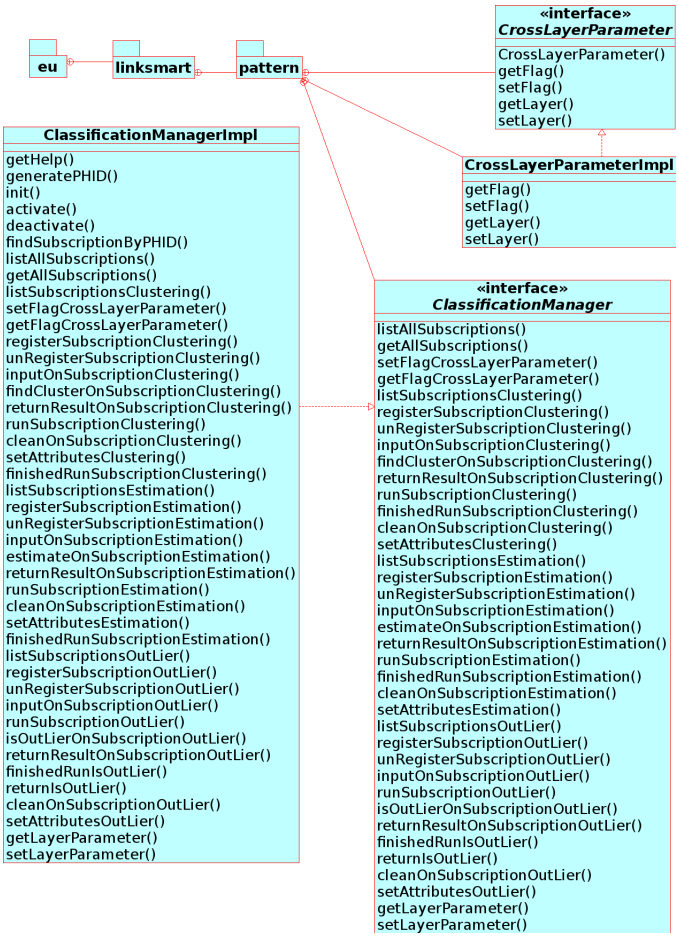

Figure 4. Class diagram of the cross-layer parameters and service implementation in the pattern recognition module.

a layer of either processed or not processed data, *getLayer* that is responsible to return the defined layer, *setFlag* that is responsible to define a boolean flag to inform if the pattern service is active or not in this layer, and *getFlag* that is responsible to return the state of the flag. The class *CrossLayerParameterImpl* is the actual implementation of the methods defined in the interface *CrossLayerParameter*.

The classes *ClassificationManager* and *ClassificationManagerImpl* define the structure and implementation of the pattern recognition services inserted in the LinkSmart Middleware, but they are not the focus of this paper.

New applications to use this structure and services implemented in the LinkSmart Middleware need to define parameters of the middleware or physical layers. Anytime, during the execution of the application, the parameters and behaviors of the physical or middleware layers can be changed. The driver program responsible to connect devices and the LinkSmart middleware must use and interpret those parameters and change the devices' behaviour.

B. A testbed implementation

This section describes the developed testbed implementation, with a resource manager and a test client application to demonstrate the functionality of the proposed cross-layer communication.

Figure 5 shows the web page with the LinkSmart status. It can be seen that the service, highlighted by a red line, called by *ClassificationManagerImpl* has started and was registered in the middleware with Hardware Identification (HID) *0.0.0.8650460202121146535*.

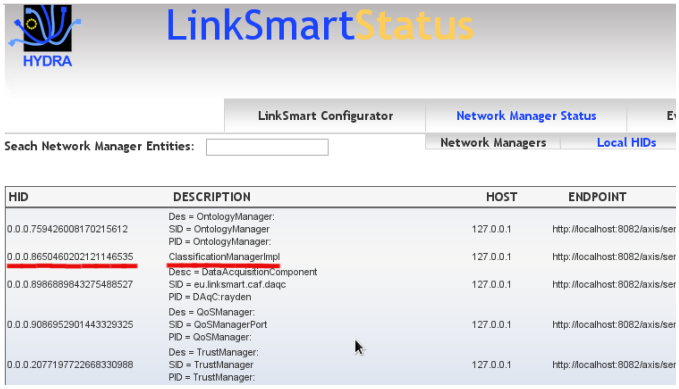

Figure 5. The LinkSmart middleware webpage with ClassificationManager services started.

The raw data used by the resource manager are from the Guildford's facility that is a member of the European Commission funded Smart Santander Project [20].

The retrieved data were inserted in the Mysql [21] database and a class to simulate the resource manager was created. The resource manager provides temperature and light intensity values from a single sensor node, designated as *node25*.

To illustrate the proposed cross-layer communication, two applications have been created: (1) ResourceManager representing the physical layer and that is responsible to retrieve the data from the Mysql database and to send them to the Pattern Recognition Manager in the LinkSmart middleware; and (2) the client application that is responsible to control the Pattern Recognition Manager or the Middleware and Physical layers behaviors.

Figure 6 illustrates the execution of the developed resource manager application to represent the physical layer. This application has two services implemented: the value estimation algorithm to estimate a value if the real sample is missing and this value is relevant to the application; the outlier detection algorithm if an erroneous data represents a problem to the application as proposed in [4].

Other relevant aspect of this application is its capability to send and receive parameters both from the Middleware and Application layers. Note that the application starts the connection with the Middleware layer, informs the status of the services and begins to send instances with temperature and light intensity values of the environment. In a certain instant of time, the Physical layer receives a message to change the

 ISBN: 978-1-61208-422-0

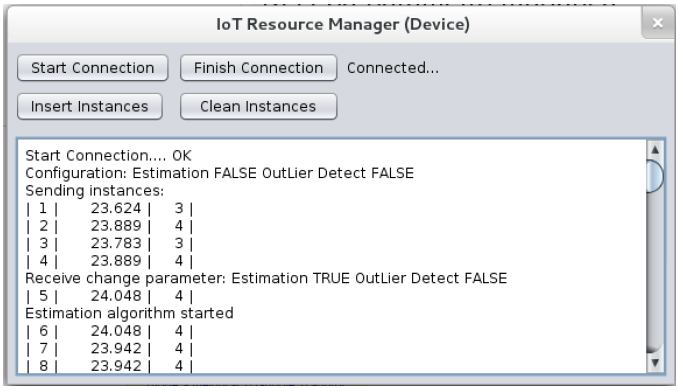

Figure 6. Resource Manager application representing the physical layer.

estimation parameter and to start this service. This execution example illustrates how the cross-layer communication enables the physical layer's behaviour modification in execution time.

Figure 7 illustrates the execution of the client control application developed for the testbed.

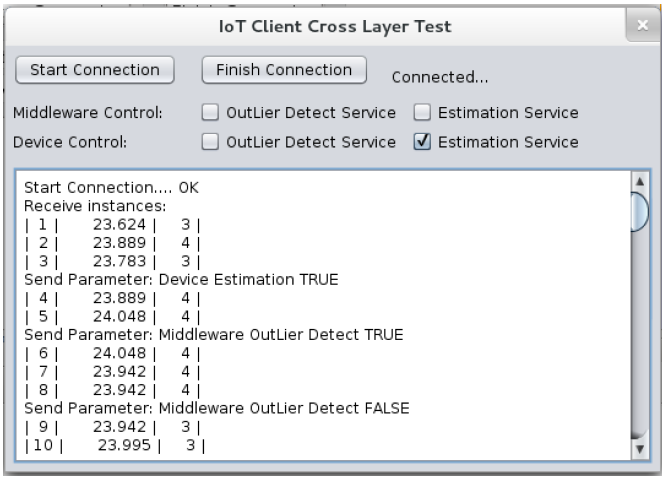

Figure 7. The client control application developed for the testbed.

This application has two functions: (1) it illustrates the client receiving data from the Middleware layer; (2) it implements the control function of the behaviors of the Physical and Middleware layers. In the application's graphical interface, it is shown the control of the Physical and Middleware layers: the application sends a message to start or stop the values estimation or outlier detection services in any of the layers using the developed cross-layer communication structure. The user can enable or disable the checkboxes in the interface.

In the illustrated example of the application execution, one can see that the program established the connection with the LinkSmart and started to receive instances with sample values. Next the user selected the checkbox to start the values estimation service in the physical layer: the application sent this parameter to the Middleware layer which forwarded the new value to the Physical layer. Then, the user marked the checkbox to start the outlier detection service in the Middleware Layer stopping it after a while, changing behaviors in both layers.

Figure 8 illustrates the LOG file generated with the execution of the ClassificationManager services implemented in the LinkSmart.

```
2015-01-09 21:35:50,378 INFO  - ClassificationManager Services Started
2015-01-09 21:35:51,129 INFO  - ClassificationManager Service OutLierDetection Started
2015-01-09 21:35:51,835 INFO  - ClassificationManager Service Estimation Started
2015-01-09 21:35:52,411 INFO  - ClassificationManager Service Clustering  Started
2015-01-09 21:35:56,984 INFO  - ResourceManager (IoTResourceManager.class)
                                 PHID 9835478513654
2015-01-09 21:35:59,129 INFO  - Client (ClientIoTCrossLayerTest.class) PHID 6584236574189
                                 Subscriber PHID 9835478513654
2015-01-09 21:36:02,455 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:02,991 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:03,393 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:04,856 INFO  - Client (ClientIoTCrossLayerTest.class) Receive
                                 ChangeParameter: Device Estimation TRUE
2015-01-09 21:36:05,384 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:07,032 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:08,775 INFO  - Client (ClientIoTCrossLayerTest.class) Receive
                                 ChangeParameter: Middleware OutLier Detect TRUE
2015-01-09 21:36:09,562 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:10,135 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:11,221 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:13,341 INFO  - Client (ClientIoTCrossLayerTest.class) Receive
                                 ChangeParameter: Middleware OutLier Detect FALSE
2015-01-09 21:36:14,378 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:15,824 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:16,258 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:17,874 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:18,561 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:19,992 INFO  - PHID 9835478513654 Receive instance...send to subscribers
2015-01-09 21:36:20,324 INFO  - PHID 9835478513654 Receive instance...send to subscribers
```

Figure 8. ClassificationManager LOG.

In Figure 8, one can see the part of the LOG file that refers to the execution of the client control application of the testbed.

This log file shows some lines referring to the initialisation of the *ClassificationManager* and its pattern recognition services, followed by the IoTResourceManager (Physical layer) and ClientIoTCrossLayerTest (Application layer) actions. The ResourceManager sends instances to the LinkSmart and the client application sends messages to the Physical layer with the estimation start parameter. Then it can be seen the messages to change services in the Middleware layer, i.e., to start and after a while to stop the outlier detection service.

The behaviors change of the Physical and Middleware layers, required by the Application layer, are also shown.

IV. Conclusions and Future Work

In this work, we have proposed, implemented and tested a cross-layer communication structure for an IoT architecture. The structure runs integrated with the novel IoT architecture implemented by extending the LinkSmart middleware with the following pattern recognition services: outlier detection, values estimation and clustering in the Middleware layer, and implementing the outlier detection and values estimation services in the Physical layer as well.

The proposed structure and its corresponding implementation allows the Application layer to change the behaviors of the lower layers in the IoT model, specifically the Physical and Middleware layers. The Application layer can enable or disable these services.

The developed object-oriented programming structure introduces scalability in the parameters handling and pattern recognition services aggregated in the LinkSmart middleware.

 ISBN: 978-1-61208-422-0

This framework addresses scalability, contextualisation and flexibility enabling a huge number of different kinds of devices to acquire environment context awareness. The information provided by a single light sensor, for example, can be read by various applications without any interference on each other. The raw data is processed only once in the Physical or Middleware layers, so different applications may be simpler and receive the filtered information, without the need to process the original raw data. This approach reduces the network traffic and the overall energy consumption.

The testbed implementation validated the proposed cross-layer communication integrated with the IoT architecture using real data from the Smart Santander Project. The execution shows that the IoT architecture implementation, the LinkSmart middleware extended with the pattern recognition services and the new communication model work with real data.

As future work, we will validate the novel IoT architecture with pattern recognition services, cross-layer communication and novel tariff systems in a real telecommunication network with real users and scalable applications.

ACKNOWLEDGEMENTS

We acknowledge the ICT- 2009-257992 (SmartSantander) and the REDUCE project grant EP/I000232/1 under the Digital Economy Programme run by Research Councils UK that supported the development and deployment of the SmartCampus testbed.

REFERENCES

[1] C. Pfister, *Getting Started with the Internet of Things*, 1st ed. O'Reilly Media, Inc., 2011.

[2] A. Jammes, J. Cooper, K. Jeffery, and G. Saake, "Research directions in database architectures for internet of things: A communication of the first international workshop on database architectures for the internet of things (dait 2009)," 2009, pp. 225–233.

[3] I. Smith, *The Internet of Things 2012: New Horizons*. CASAGRAS2, 2012, retrieved: July, 2014. [Online]. Available: http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf

[4] A. M. Souza and J. R. Amazonas, "A novel smart home application using an internet of things middleware," in *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2013 European Conference on*, 2013, pp. 1–7.

[5] E. F. P. CASAGRAS, "Casagras final report: Rfid and the inclusive model for the internet of things," 2009.

[6] J. R. d. A. Amazonas, "Network virtualization and cloud computing: Iot enabling thecnologies," *Casagras2 Academic Seminar*, September 2011, retrieved: November, 2013. [Online]. Available: http://www.casagras2.com.br/downloads/day2/2-Jose_Roberto_de_Almeida_Amazonas-Network'_Virtualization_andar_Cloud_Computing_IoT_enabling_echnologies.pdf

[7] B. C. Hardgrave, J. Aloysius, and S. Goyal, "Does rfid improve inventory accuracy? a preliminary analysis," *International Journal of RF Technologies: Research and Applications*, vol. 1, no. 1, pp. 44–56, 2009. [Online]. Available: http://dx.doi.org/10.1080/17545730802338333

[8] K. E. Kjaer, "A survey of context-aware middleware," 2005, retrieved: January, 2015. [Online]. Available: http://hydramiddleware.eu/hydra_papers/A_Survey_of_Context-aware_Middleware.pdf

[9] M. Sarnovsky, P. Kostelink, P. Butka, J. Hreno, and D. Lackova, "First demonstrator of hydra middleware architecture for building automation," June 2005, retrieved: January, 2015. [Online]. Available: http://www.hydramiddleware.eu/downloads.php?cat_id=2&download_id=29

[10] H. Project, "Hydra project overview," June 2007, retrieved: January, 2015. [Online]. Available: http://www.hydramiddleware.eu/articles.php?article_id=68

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*, 2nd ed. Wiley-Interscience, November 2000.

[12] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*, 4th ed. Academic Press, 2008.

[13] H. Sun and P. Heller, "Oracle information architecture: An architect s guide to big data," in *An Oracle White Paper in Enterprise Architecture*, 2012.

[14] D. Tracey and C. Sreenan, "A holistic architecture for the internet of things, sensing services and big data," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013, pp. 546–553.

[15] R. Pamula, J. Deka, and S. Nandi, "An outlier detection method based on clustering," in *Emerging Applications of Information Technology (EAIT), 2011 Second International Conference on*, 2011, pp. 253–256.

[16] D. Lei, Q. Zhu, J. Chen, H. Lin, and P. Yang, "Automatic k-means clustering algorithm for outlier detection," in *Information Engineering and Applications*, ser. Lecture Notes in Electrical Engineering, R. Zhu and Y. Ma, Eds. Springer London, 2012, vol. 154, pp. 363–372.

[17] A. M. Souza and J. R. Amazonas, "An outlier detect algorithm using big data processing and internet of things architecture," *Procedia Computer Science*, vol. 52, no. 0, pp. 1010 – 1015, 201, the 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915008959

[18] W. Joachim and S. Walewski, "Internet of things architecture iot-a," *Deliverable D1.4 - Converged architectural reference model for the IoT v2.0*, 2012.

[19] A. Tanenbaum, *Computer Networks*, 4th ed. Upper Saddle River; NJ: Prentice Hall, 2003.

[20] M. Nati, A. Gluhak, H. Abangar, and W. Headley, "Smartcampus: A user-centric testbed for internet of things experimentation," in *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*, 2013, pp. 1–6.

[21] S. Tahaghoghi and H. Williams, *Learning MySQL*. O'Reilly Media, 2006.

 ISBN: 978-1-61208-422-0