Reflective Internet of Things Middleware-enabled a predictive real-time Waste Monitoring System

Vito Bellini, Tommaso Di Noia, Marina Mongiello, Francesco Nocera, Angelo Parchitelli, and Eugenio Di Sciascio

Department of Electrical & Information Engineering
Polytechnic University of Bari, Bari, Italy
{firstname.lastname}@poliba.it

Abstract. Nowadays, Urban Waste Collection process has become crucial to ensure cities' wealth and viability. The growth of urban centers and the rapid expansion of industry led to a revision of plans and waste collection routes to increase their efficiency and effectiveness. Traditional approaches for the Vehicle Routing Problem and the Waste Collection Problem are not taking into account some factors such as the huge amount of available information produced by the Internet of Things devices.
In this paper we propose a solution allowing an Internet of Things middleware to conform with reactive programming paradigm that hence will be more flexibile and adaptive for the Waste Management Problem by retrieving the best route planning using all real-time information available in a Smart City context. All gathered data stoke a predictive model used to estimate the best timing to optimize the waste collection, allowing the system to plan optimal routes accordingly to the state of bins across the city area.

Keywords: Reflective middleware, Internet of Things, Waste Management Problem, Deep Learning

1 Introduction and motivation

The Smart City paradigm has become one of the most important research topics around the globe. Many cities across the world promote Smart City models, which include a wide range of features for sustenance to provide municipalities with a better use of their resources using Internet of Things (IoT) sensor network to monitoring all city data, from environmental to bus tracking. In this model all data becomes useful information for day-by-day decision making. This processes driving all cities living and becoming fixed references for example for linking and upgrading infrastructures, technologies and services in transportation, energy grid and so on. Waste management organization and city administrations, in different location provide the challenge to an efficient and effective system to collect, dispose-off, and recycle the waste, with particular attention for the health

standards and environment friendliness. A typical application field for such initiatives is Waste Management (WS) including collection, transportation, and disposal of garbage and other waste products. This factors are negatively influenced by improper bin collection system, lack of information about collection schedule, inefficient route planning, insufficient resources, and other factors [1]. The realm of this paper is to design and develop an IoT-based and real time waste monitoring system. Modern technologies such as real-time and reflective IoT middleware, cloud system, Wi-Fi, GSM as well as ultrasonic ranger sensor are implemented. Offering a substantial way to optimize solid waste management increasing collection throughput and improve its sustainability. Moreover, the implemented system allows to predict the best route that solve the Vehicle Routing Problem and its constraints and for a long time the citizens' habits.

2 Background and related work

The Internet of Things (IoT) is one of the most influential technological shift we are facing in the last years. One of the challenges in building an IoT (and Services), Middleware is the way software will be developed and composed on the top of flexible infrastructures and integration architectures. Although the great interest in software composition and verification methods, when developing service and thing-based software systems many challenges are still unexplored. The strongest challenges still regard the way a smart IoT-based architecture is designed in order to make it robust with respect to the contextual changes it continually undergoes [2, 3].
In this paper we propose a reflective model whose aim is to inject context-awareness into an IoT middleware [4, 5]. The reflective extension allows a software system to dynamically change its logic without internal changes to the code. In our approach, the awareness of the surrounding context is encoded by means of a rule-based system which drives the dynamic behavior of the middleware. We refer to this previous work [6] for a detailed description of the approach, and for a Recap of Reflection paradigm.

Related work Unquestionably there also several efforts invested in waste management with IoT, in general, smart waste monitoring system consists of sensors and transmission medium, collecting different types of data regarding the waste detection level found inside a smart bin.
Developing predictive models for waste management systems could improve the efficiency and reduce costs as marked out by [7]. In this work, the authors present a simple predictive model that is used to estimate waste generation rates, useful to calculate how many bins to dispose in different city areas. They also found that being able to estimate correctly waste generation rates help engineers and planners to calculate type, size and location of bins and transportation routes as well.
Another work [8], tries to do forecasting of waste generation using regression models. In this work they found some variables for every type of garabe in order

to predict the amount of solid waste fractions. In particular they use number of residents, population age, urban life expectancy, total municipal solid waste as inputs for their regression model.
A smart city service[9] for real time waste monitoring and collection was designed and developed. The system consists in smart bins; each bin installed with Arduino Uno, ultrasonic sensor and Radio Frequency (RF) transmitter on the top of the container. When the container is full of waste, it sends signal to the control center which will have the level of waste in the containers and through GSM/GPRS, a message (SMS) will send to the mobile phone of the truck driver of which waste bin is full and need to be empty.

3 Smart Waste Management System Architecture

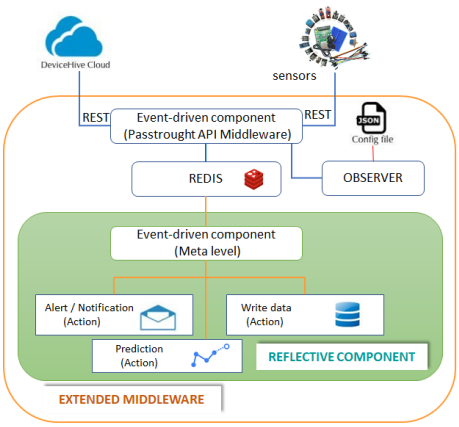


Fig. 1. Model of proposed architecture.

In this section we describe how we instantiated the proposed reflective model depicted in Fig. 1. Each component is described in Table 1.
Infrastructural components. Garbage container therefore must be supported by the following sensors in order to accomplish all the smart requirements: *NFC reader*, used to unlock the containers opening and keep tracks of the users usage; *Weight sensor*, to weigh the user wastes production; *Ultrasonic fill-level sensor*, capable to detect the fill-level for a container, it's positioned at container's top cover; *GPS sensor*, used for geo-location purposes; *Temperature sensor*, used to predict the deterioration rate for organic wastes based on external temperature and to detect vandalism such as burning the container; *Bluetooth low energy chip and antenna*, to communicate with others smart containers and garbage trucks; *Battery*, a long life-cycle battery to provide power to the communications chips.

Table 1. Instantiation of the model.

Component description	
The DeviceHive middleware	Among the various IoT middleware available in the market, we choose *DeviceHive* for several reasons: easy to install, a rich documentation and high integration with a wide range of programming languages and IoT protocols.
Redis as MOM	In our instantiation of the model we adopt Redis[1], a NoSQL DBMS, as MOM. Using Redis as Message Broker allows the system to translate a message from the messaging protocol of the producer to the recipient's messaging protocol. Through appropriate Publish/Subscribe directives, Redis implements the mechanism of Publish/Subscribe, whereby producers do not send messages directly to recipients; rather the messages are published and sorted into channels, without knowing who is really writing to the channel.
Message Bus	Instantiation of the *Message Bus* of our model is obtained using the Redis Message channel. To implement the Publish/Subscribe mechanism, publishers publish messages on the Redis channel, recipients subscribe to the channel and read information about the available services.
Observer	The Observer component observes rules extracted from the Rules Repository. The Observer (of the data flow) notifies the Message Broker about the arrival of new data as well as of the active rule and its status (activation or execution).
Event-driven component	We use Node JS as Event-driven component. It receives data about the sensor and the domain value of the sensor through a RESTful interface.
Meta Level	We use a Node JS process for the activation of the meta-level. In the instantiation it contains a simple call to a generic function, with the string identifying the methods and the classes of the base level. According to this mechanism, the logic of the program can be dynamically changed depending on the data got from the sensors and changes are transparent to the internal structure of the software. Within the message received by Redis the class name and the name of the function to call are specified in special strings. The data to be written is contained in a separate object, which will be the argument of the function.
Adapters and Actions	In order to determine which are the Actions available at runtime, a system for loading dynamic components within the same *Meta level* has been implemented. Two files for each *Action Levels* are available: a JSON file containing configuration data and a JavaScript file containing the class itself. Objects instantiated with the data in the JSON file will then be stored in an array, and referenced by the name of the same class, specified within each rule.

Reflective behavior. The system automatically performs actions according to the values received by the sensors via a matching of the set of formal rules. A configuration file contains the definition of the rules on the data and the actions to take if the rule is fired. The Observer (of the data flow) notifies the Node JS Component that forwards the extracted active rule to the Message Broker. The active rule and its status together with the information about sensors and domain value is published on the *Message Bus*. On the message channel information about the arrival of new data is published. The Message Broker works as a through for data flow and the active rule that are forwarded to the Reflective part of our component. The consumers subscribed to the Message Broker are notified about the message. The Message Broker forwards the whole message made up of sensor, domain value and the rule (extracted from the config file) to the Rule Engine that executes it. Reflection is applied thanks to the signing of the consumer component of the Redis channel. This component reads the published messages that contain specific information on the method to be called, the class of which the function is a member and the topics. The *Meta Level* enables the actions of the reflective component. The activations are managed via a plugin system. Each plugin is self-consistent and controls an appropriate device in Plug&Play controller. In this first instantiation, we provide the following actions: *(i)* Alert to Fire Department; *(ii)* Alert to Environmental Protection Office; *(iii)* Notification to a *Route Decision System Planner*. The alert event is a notification, like as an e-mail or a push notification, to a single department or more, when an event happens. The performed events are derived from high-level conditions, sensors value, properties and context.

Data transfer. Data are gathered from bins to a public vehicle every time they are nearby the containers and transferred using the standard Bluetooth Low Energy[2] (BLE) to the middleaware. This form is an extension of Social Network and it allows to applying the concept of social networking to IoT object ensuring to establish relationships with other objects [10]. We use as metric the social importance of the public transport (autobus, police car, taxi, garbage truck,...). This particular metric is static and allows to take advantage of Wi-Fi public link of the public vehicle.
For example, the bins A wants to communicate our data status in a particular time slot. A citizen bus pass through near the bins in a Bluethoot range action. The bins send a packet data via Bluethoot to a bus and this forward data to the middleaware. The timeslot transmission is set for 1 hour expired that the packet is rejected by the middleware. For this reason we use the timestamp field. Another check is performed: an unique identifier– sending with the packet – allows to block a possible duplicate packet.

We can analyze the exchange of information between the various components involved. Follow an example of message:

```
{ "device":" B1-L1", "uid":1323232, "w_sensor":30, "temp_sensor":16,
"hum_sensor":58, "fill_sensor":20, "co2_sensor":107,
"timestamp":"1516096813" }
```

The simple data packet, in JSON format, show all the variables transfer from bin to the middleware. Based on the value of sensors and the retrieved rule the *Meta Level* enables the actions of the reflective component.

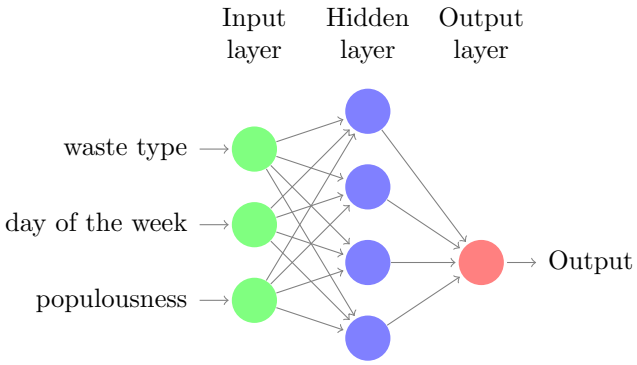
Route Decision System Planner. In order to select the best routes for garbage trucks, a predictive model is required to estimate which bins will be full. On that estimations we are able to plan routes selecting those one having the higher number of bins ready to be gathered [11]. After, we use a Vehicle Routing Problem, a generalized of a "Travelling salesman problem"that allows to calculate the best route, in particular the shortest possible route, passing through each node. In our case, as you know, the node is a particular bins that must be cleaned up. This component is responsible to make analysis over data collected from different sources and different measures. It helps to find out when the container will be filled and when it is the best time to schedule its processing by the garbage truck. To achieve this, a deep learning method is applied on the data in order to build a regression model that predicts in how many hours the container will be filled. We choose to use a deep learning model because artificial neural networks are much faster to train respect to other methods like SVM, where their training time gets harder on very large datasets. Moreover neural networks can easly handle high dimensionality features space and this let us to leverange on features taken across different kinds of sources without any particolar feature-engineering process.

The model is trained with data ($\mathbb{X}^{nxm}$, $\mathbb{Y}^{nx1}$) collected over the time. $\mathbb{X}$ contains n vectors of m features and $\mathbb{Y}$ contains n scalars that corresponds to

[2] https://www.bluetooth.com/specifications/bluetooth-core-specification

the number of hours since last time the container is processed by the garbage truck because it's filled up of waste.

Input features are: *type*: the type of waste (organic, plastic, paper, iron); *day of the week*; *populousness*: discrete variable indicating high, medium and low populousness areas.



Such a model should be re-trained periodically as new data arrives, in order to keep it update and to make it provide accurate predictions that reflect changes in time of the citizen's behaviour.

After this process we apply a particular Vehicle Routing Problem solve algorithm. The road network can be described using a graph where the arcs are roads and vertices are junctions between roads. Like a normal city, the arcs may be directed or undirected due to the possible presence of one way streets or not. We choose a Ant Colony Optimization (ACO) technique [12, 13], for solve this particular problem. This is a meta-heuristic technique where a set of ants are used to solve this optimization problem. Based on real world, the ant is able to find the shortest path between a food source and the nest, thanks to is secreted by pheromone. The principal step to compute the optimal path for VRP is a route construction, based on graph and the pheromone trail update. For our problem, we run an ACO for maximum n nodes that compose a cluster. This is a simple situation and in [14] we found a particular algorithm that show the effectiveness of this method.

4 Experimental field

We conducted experiments doing a controlled simulation to generate data of urban waste production for a city. We made some assumptions on bins filling rate depending on day and the populousness of an area. In particular we hypothesized that during the weekend the garbage production is higher respect other days of the week and we assigned a different bins filling rate to different areas of the city. The data are simulated using a particular set of coefficient that regulate the filling equation. In detail, we use the below equation:

$$h = p * d * m * k$$

The result of equation is the count of hours after which the bins is completely full for a particular waste type. We modelled for variables: p: represent the population area, divided in low density, medium density and high density; d: represent the day of week; m: represent the month; k: represent the normalization coefficient. We choose a discrete coefficient that modelling the production of particular waste. For example: in a high density area ($p = 1$) on Sunday ($d = 1$) in January ($m = 1, 3$) with static constant fixed to $k = 14$ the organic bins must be emptied after 18 hours. Following this equation we simulated 6300 data for 2 bins to train our neural network model designed for a regression task, made of 3 hidden layers by 50 neurons each. Categorical variables such as day of the week, populousness of the area and waste type are one hot encoded, for example $wastetype = \{1, 2\}$ is converted in a vector (0, 1) for type 1 and (1, 0) for type 2. These one hot encoded vector for each variables are concatenated in a unique input vector of length $floor(log_2(n) + 1)$ where n is the sum of the cardinality of each set of values that represent a categorical variable. The model we trained is able to predict the number of hours needed for a bin to be filled. To evaluate our model we used the hold-out 80/20 protocol. It consists in randomly splitting the entire dataset in two parts: train-set and test-set. The former contains 80% of examples from dataset and it is used only to train the model. The latter is used for evaluations. The evaluation metric we chose is Mean Squared Error. In our experiments we found that the average error for each bin is in the range of of 7 to 10 hours. We can assert that the model is underfitting the data and to address this issue more discriminating variables should be taken into account.
Moreover, we performed intensive tests on the IoT middleware hereby proposed targeted at evaluating non-functional requirements, more specifically scalability and quality of service (QoS).

5 Conclusion and future work

In this work, we propose an innovative solution for real time waste bins monitoring system based on IoT has been designed and implemented. The solution is based on a reflective extension of an IoT middleware which makes possible the design of a software resulting completely configurable and adaptable to different operating environments. The proposed system enables to automatically perform actions according to the sensor values received by triggering a set of rules able to better describe environmental data w.r.t. to actions to be performed. The developed solution system can be used to reduce valued human resources like human effort, time and cost and improve the sustainability of the solid waste operations and obtaining green environment.

References

1. M. Caniato, M. Vaccari, C. Visvanathan, and C. Zurbrügg, "Using social network and stakeholder analysis to help evaluate infectious waste management: A step towards a holistic assessment," *Waste Management*, vol. 34, no. 5, pp. 938–951, 2014.

2. F. Nocera, "Fuzzy ontology-driven web-based framework for supporting architectural design: student research abstract," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 2016, pp. 1361–1362.
3. T. Di Noia, M. Mongiello, F. Nocera, and U. Straccia, "A fuzzy ontology-based approach for tool-supported decision making in architectural design," *Knowledge and Information Systems*, pp. 1–30.
4. T. Di Noia, M. Mongiello, F. Nocera, and E. Di Sciascio, "Ontology-based reflective iot middleware-enabled agriculture decision support system."
5. F. Nocera, T. Di Noia, M. Mongiello, and E. Di Sciascio, "Semantic iot middleware-enabled mobile complex event processing for integrated pest management," in *In Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pp. 610–617.
6. M. Mongiello, T. di Noia, F. Nocera, E. di Sciascio, and A. Parchitelli, "Context-aware design of reflective middleware in the internet of everything," in *Federation of International Conferences on Software Technologies: Applications and Foundations*. Springer, 2016, pp. 423–435.
7. Z. Sakawi and S. Gerrard, "The development of predictive model for waste generation rates in malaysia," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 5, no. 5, pp. 1774–1780, 2013.
8. C. Ghinea, E. N. Drgoi, E.-D. Comni, M. Gavrilescu, T. Cmpean, S. Curteanu, and M. Gavrilescu, "Forecasting municipal solid waste generation using prognostic tools and regression analysis," *Journal of Environmental Management*, vol. 182, pp. 80 – 93, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0301479716304637
9. M. S. Omar, M. A. Anjum, S. A. Hassan, H. Pervaiz, and Q. Niv, "Performance analysis of hybrid 5g cellular networks exploiting mmwave capabilities in suburban areas," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.
10. F. Nocera and A. Parchitelli, "An adaptive formal metamodel for semantic complex event processing-driven social internet of things network," in *Current Trends in Web Engineering*, I. Garrigós and M. Wimmer, Eds. Cham: Springer International Publishing, 2018, pp. 7–18.
11. A. Parchitelli, F. Nocera, G. Iacobellis, M. Mongiello, T. D. Noia, and E. D. Sciascio, "A pre-process clustering methods for the waste collection problem," in *2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Sept 2017, pp. 242–247.
12. X. Wang, T.-M. Choi, H. Liu, and X. Yue, "Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3132–3141, 2016.
13. J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced engineering informatics*, vol. 18, no. 1, pp. 41–48, 2004.
14. M. Reed, A. Yiannakou, and R. Evering, "An ant colony algorithm for the multi-compartment vehicle routing problem," *Applied Soft Computing*, vol. 15, pp. 169–176, 2014.