Toward a Real-Time Development and Deployment of IoTs Application for Smart Garden on OpenStack Cloud

Dang Huynh-Van, Khanh Tran-Quoc, and Quan Le-Trung(✉)

Department of Computer Networks, University of Information Technology, Viet Nam National University, Ho Chi Minh City, Vietnam
{13520180, 13520388}@gm.uit.edu.vn, quanlt@uit.edu.vn

Abstract. While lots of Internet of Things (IoTs) applications have been designed and implemented for many areas, especially for the ambient-assisted living domain, real-time development and deployment of those IoTs applications have still been an open issue. This paper focuses on such an open issue through our developed web-based IoTs application for a smart garden, be integrated into and managed via the OpenStack Infrastructure at University of Information Technology (UiT). An intelligent model applying fuzzy logic to process measured data for the smart garden, has also developed to give useful information on keeping plants growing, monitoring, and taking care of plants more easily and effectively. Load-balance network architecture on cloud is used to deal with the scalability, availability of the deployed system.

Keywords: Wireless sensor network · Smart model · Fuzzy logic controller Smart garden · TinyOS · Collection Tree Protocol

1 Introduction

Nowadays, Internet of Things (IoTs) is one of the newest research topics all over the world. Lots of research have been focused on such a domain to develop more creative products to enhance our life quality. IoTs applications appear in a lot of different domains [1], e.g., health care, industry, environment monitoring, weather forecast, transportation as well as agriculture. In smart agriculture field, there are lots of systems based on IoTs. These systems have been developed to solve many practical problems in the modern agriculture domain, e.g., environmental monitoring, soil parameters monitoring, automatically watering, etc. For example, in "Smart Rabbit Farm" [2], there is an IoT-based application to monitor the rabbit farm conditions with various sensors. On this system, the authors also used the cloud network infrastructure to store the collected data and after that visualizing these environmental parameters as well as useful notifications generated by basically comparing these values with the threshold. Some other system as in [3–5], environmental parameters or soil parameters were also concerned to create some useful IoTs-based applications. Most of these systems have been developed by using a single standalone server [3] or an application running on a smartphone [4] to store and visualize measured data. These solutions are only suitable for

Y. Chen and T. Q. Duong (Eds.): INISCOM 2017, LNICST 221, pp. 121–130, 2018.
https://doi.org/10.1007/978-3-319-74176-5_12

small-scale areas or some decorated houseplants while the *scalability* and *availability* are not the primary requirements. However, with the high demands of the modern life, to deploy a smart solution in a real situation, there are many challenges [6] we must consider, such as the real-time monitoring, the scalability and the availability.

In this paper, we present a Smart Garden IoTs-based system. *This system deals with above challenges by using a load-balance network architecture on the Open Stack cloud as well as applying the fuzzy logic controller to analyze measured data.* Our system not only helpful for the gardeners on keeping their plants growing well, but also suitable to deal with high demand for real-time monitoring.

The structure of this paper is organized as follows. Section 1 introduces the research topic and break down current challenges on IoTs-based in smart agriculture domain as well as our paper's contribution. Section 2 clarifies our system implementation. This Section includes fours sub-Sections, and each sub-Section presents a part of our core system modules. Section 3 is the results of our system working in a practical situation. Section 4 shows the related work in the current IoTs-based application for smart agriculture as well as wireless sensor devices and wireless sensor network architecture. Finally, Sect. 5 ends this paper with the conclusions and future work.

2 Implementation

In this Section, we clearly explain our system architecture by stating every part of the model. Figure 1 shows the overview of our system, there are three important modules: wireless sensor nodes (IoTs end devices), servers system, and web interface.

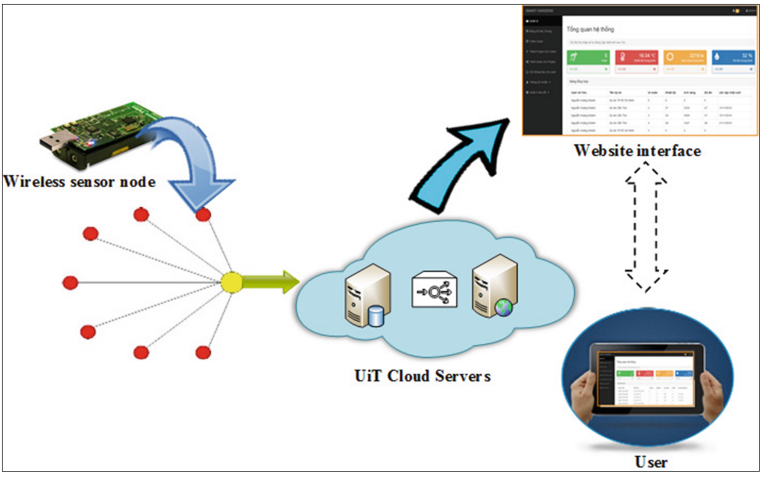

Fig. 1. Overview of IoTs application system architecture over UiT cloud servers

2.1 Programming Wireless Sensor Devices

In our system, we used Telosb Mote which was developed by UC Berkeley. It is an open source platform includes all the essentials for lab studies such as USB programming capability, an IEEE 802.15.4 radio with integrated onboard antenna, TI Micro Controller MSP430 with 10 kB. It also has optional sensor suit include: temperature, light and humidity sensor. The Telosb platform run TinyOS 1.1.10 or higher as well as Contiki – which are current most popular open source operating system for wireless sensor devices. In our implementation, we used TinyOS 2.1.2.

For the purpose of this study, we write a program in nesC – a programming language for creating TinyOS applications [7]. This program is loaded into sensor nodes to collect environment data as temperature, light, and humidity then sends these data to root node. SensorCollection program has three important files: module file (SensorCollecionC.nc), configuration file (SensorCollectionAppC.nc) and Makefile. The module file lists all interfaces that we will use in our application and define the program implementation. The configuration file is responsible for assembling other components together, connecting interfaces used by components to interfaces provided by others. The Makefile is used to compile the application.

In the SensorCollection application, we used Collection Tree Protocol (CTP) provided by *ColectionC* component in TinyOS as a collection routing protocol for data transmission. CTP is a distance vector protocol designed for sensors network and used in research, teaching and commercial products. CTP is a tree-based collection protocol. Some number of nodes in network work as a tree root. Other nodes create a set of routing tree to these roots. CTP is address-free so that it does not send a packet to a specific root; instead, it chooses a root by choosing next hop – called its parent node. CTP uses a value called expected transmission (ETX) as its routing gradient (similarly to metric value in other routing protocol). Root's ETX is 0 and other node's ETX are the ETX of its parents plus the ETX of its link to its parents. When choosing a route for transmitting data, CTP chooses the path which has lowest ETX value. An implementation of CTP is stored in tos/lib/net/ctp directory of TinyOS.

2.2 Building Server System

Our server system is developed based on OpenStack Cloud of the University of Information Technology. The model systems include two web servers, two database servers, two load balancers. All of them run Centos version 6.6. The full topology architecture is shown in Fig. 2.

By using double-quantity servers, our system guarantees if one of web server dies then all request of users will automatically redirect to another that enhance the scalability and availability of the deployed system. **Web server:** On the web server Apache is installed (Apache is as well-known open source web server). After completing installation Apache, we continue installing PHP and PHP's module as a server site language to develop our website. To make the content on the directory/var/www of two web servers automatically synchronized, we use unison solution. Unison is a file synchronization tool, which synchronizes manner Master-Slave. For example, when files on server 1 modified, it also happened in server 2. **Database server:** About the

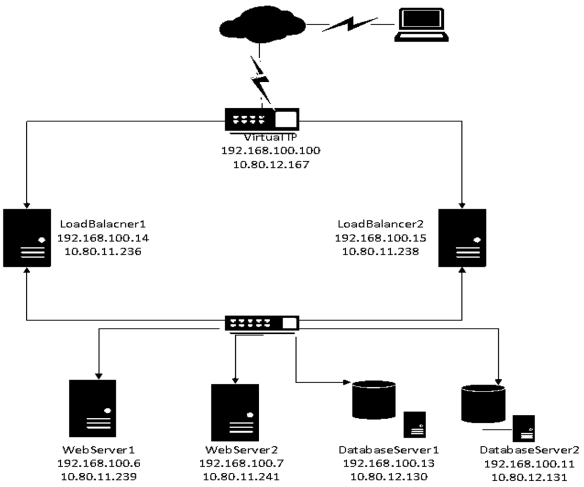

Fig. 2. The server system architecture

database, we use MySQL as an open-source relational database management system. Also for the simple management of tracking activities, we use phpMyAdmin - The open source software written in Php is used for administering the Mysql server through a web browser. Because we use two databases to ensure real-time store collected data from sensor nodes, we must configure MySQL Replication to make one of them become the master database, and other become slave database. **Load balancer:** The load balancer is a transfer and control information system, which ensure secure web server and database server. We use Keep alive to make a Virtual IP, after that, we install HAproxy - a well-known open source software for TCP/HTTP load balancer, it helps handle the incoming packets to different servers to help our server system not be overloaded.

2.3 Developing Web Application

After we set up our server system, we build a web application to analyze and visualize collected sensors data. This application is designed for two different using objects: administrators and normal users. The administrators are the garden owner, and normal users are gardeners who work in the garden and use this system for looking after their plants easily and more efficiently. We have created a database that includes four primary tables named client, project, node, and tree for storing corresponding data. *Client:* this table stores user information like user id, username, password, user role, and some contact information (phone number, address…) *Project:* storing information related to the project that will be assigned to a particular user defined in the client table. *Node:* this table stores wireless sensor node information, including node id, project id, tree id, time update new data and sensors data (temperature, light, humidity). *Tree:* designed for storing information of the plants in our garden. These data will be used as standard data in fuzzy logic controller; we will clearly explain in next section. After we

complete the database, we implement our visualization system by creating and website for two usage purpose as we said above: administrators and normal users. The administrators have full privileges to manage their gardens. They can visualize the garden overview status as well as control all devices, gardeners and their post on the website. While normal users – the gardeners have limited features, the only can view the garden that administrator assigned them. This user also can manage the sensors nodes used for their areas and read the posts created by administrators to get useful information related to their plants.

2.4 Applying Fuzzy Logic Controller for Data Processing

One of the most important part in our web-based application is data processing mechanism. In order to provide helpful information to gardeners, the system must analyze collected raw sensors data before it displays alerts or reports on web interface. Figure 3 shows the data flow processing:

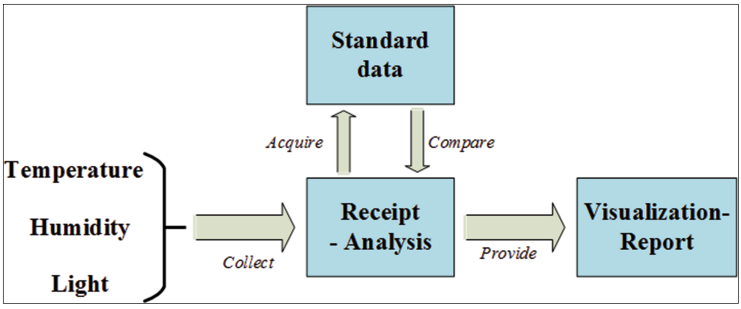

Fig. 3. Collected data processing flow.

As we show in this figure, raw data from sensor nodes are sent to server system for storing and then analyzed by the reasoning engine to create helpful information displayed on website. In *Receipt and Analysis* block, our system handles these data in different controller engine depend on their types. In particular, temperature data will be considered in temperature fuzzy logic controller as we show below.

Temperature fuzzy logic controller: To analyze temperature collected data value, we propose a simple fuzzy logic model as explained below (Fig. 4):

Fig. 4. Temperature fuzzy logic controller model

Input: $T_{node} - T_{setpoint}$ is the difference between the collected data from sensor node the standard value set before (the best suitable temperature recommended for the development of specific plant which we are taking care of). In this model, we defined five input level named: *NB* (negative big), *NM* (negative medium), *Z* (Zero), *PM* (positive medium) and *PB* (positive big). *Output*: Action is the necessary activities that gardener should follow to keep their plants growing well. To create the make sense output, we called five output value as *HeatHi* (gardener should doing something to increase temperature with a "high" level), similarly other values is *HeatLo* (heating with lower level), *Nothing* (doing nothing), *VenLo* (ventilating or slowing down temperature with low level) and the last in *VenHi* (we should slow down temperature steadily).

- *Membership functions:* The membership function associated with the input and output control variables are shown in the following figures (Figs. 5 and 6):

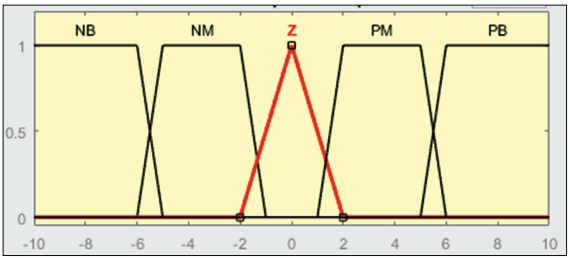

Fig. 5. The membership function of the input variables ($T_{node} - T_{setpoint}$)

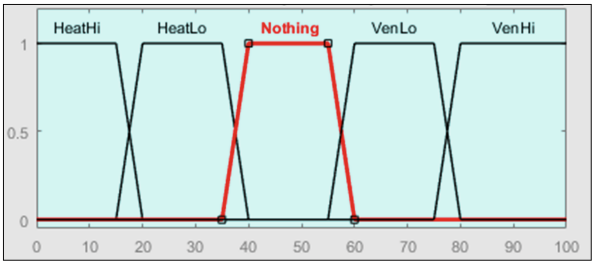

Fig. 6. The membership function for output control variables - Action

- *Inference rules:* The inference rules is shown on Table 1 based on Mandani rules composition: **IF** $(T_{node} - T_{setpoint})$ **is** (NB, NM, Z, PM, PB) **THEN** *Action* **is** (HeatHi, HeatLo, Nothing, VenLo, VenHi).

The Humidity and Light Controller are also similar with the Temperature Controler.

Table 1. The inference rules of temperature fuzzy logic controller

Rule	IF	$T_{node} - T_{setpoint}$	THEN	Action
1	IF	Negative big (NB)	THEN	Heating (High)
2	IF	Negative medium (NM)	THEN	Heating (Low)
3	IF	Zero (Z)	THEN	Nothing
4	IF	Positive medium (PM)	THEN	Ventilation (Low)
5	IF	Positive big (PB)	THEN	Ventilation (High)

3 Result and Evaluation

Using Mathlab to simulate those fuzzy reasoning model, we get result as shown in following figures (Fig. 7):

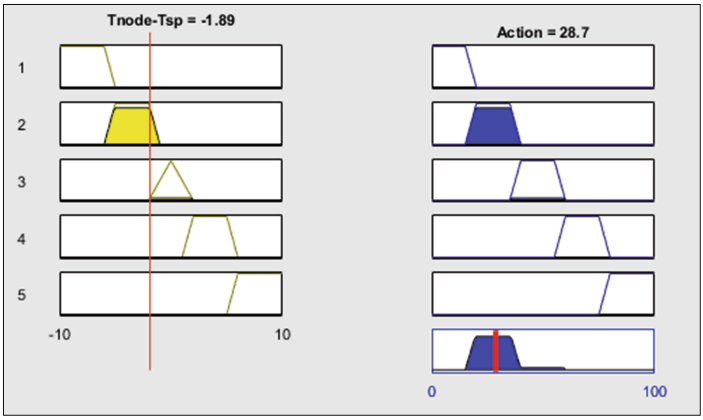

Fig. 7. An example of temperature fuzzy rule implementation.

This figure demonstrates the temperature fuzzy controller with a specific input value and the corresponding output action values by applying the interference rules that we defined in above section. As the figure shows, when the collected temperature value is smaller than the set point (–2 units) we have the output which towards to the second partition (should reduce the garden temperature slightly).

By using a PHP-based fuzzy logic library, we have embedded this reasoning engine to our web-based system. We also present the signal level from the *action* output as human-readable alerts so that the gardeners can easily know what they should do next for their plants growing well. Some sample alerts from our system is shown in Fig. 8.

As the figure shows, we define three levels for alert displaying, they are red, yellow and green. The green color means collected data is in a safe zone, which is the proper condition for plants growing. The yellow color informs received data is slightly higher than our expected value while the red color is dangerous, we must focus on this area because of the extremely difference between values from the sensor node and our standard values (Fig. 8).

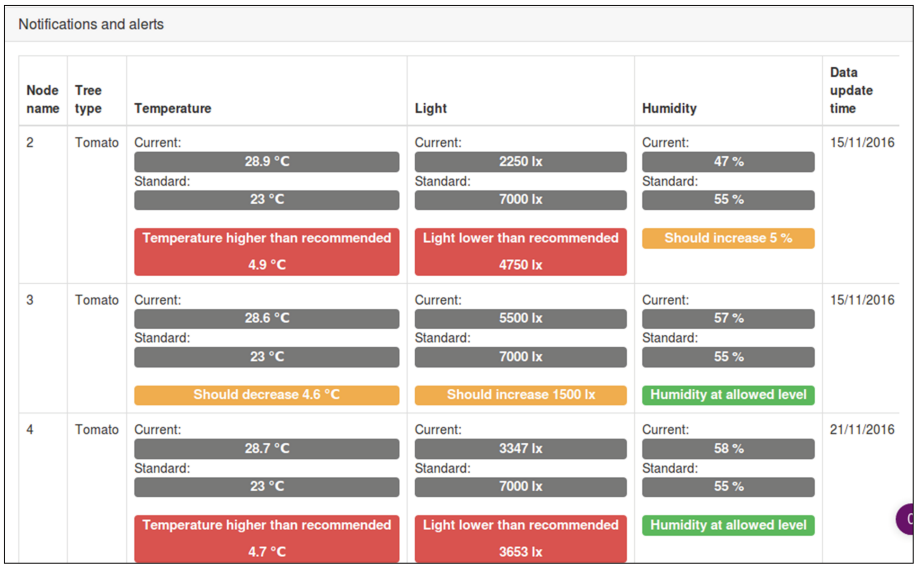

Fig. 8. A screen shot of our web-app visualization when operating in a practical experience. (Color figure online)

To verify if our reasoning model is working well, we have run the fuzzy logic controller with some sample input value on Matlab as well as on our web application. Table 2 shows the result of our experiment with the temperature controller:

Table 2. The experiment result from the temperature controller with some different input value

Input values	MatLab output	Web-app output	Web's notification
−8	8.56	8.33	*Red:* Temperature lower than recommended 7°C
−4	27.5	27.65	*Yellow:* Should increase 3°C
1	47.5	47.61	*Green:* Temperature at allowed level
4	67.5	67.65	*Yellow:* Should decrease 3°C
8	89	89.6	*Red:* Temperature higher than recommended 7°C

As the table shows, the output value from simulation on Matlab is similar with our fuzzy controller implemented on the web-app.

4 Related Work

IoT-based application. In the environmental monitoring field, applied for smart agriculture, there are many systems based on Internet of Things [2, 4]. In [3], a private cloud was developed for the use in precision agriculture and ecological monitoring.

In that system, the authors used many open source tools: Linux, LAMP stack, PHP programming language and Laravel framework as well as lots of sensor nodes based on Arduino, Raspberry Pi and Libelium Plug and Sense. The system has just developed and tested using a single-core stand-alone server so that not guarantee for availability of the deployed system. Another real monitoring system based on IoT [5], which used some wireless sensor devices as Raspberri Pi, soil moisture sensor, temperature and water level sensor as well as AWS IoT technology for implementing an automatically take care of houseplants. In generally, the IoT-based system for smart agriculture is a new research topic and still attracting many researchers and investors to develop more intelligent systems for solving the practical problems and enhance our life quality. These systems must deal with some challenges as scalability, availability as well as security. In that trends, cloud computing is an emerging technology necessary for considering and applying on future IoT-based systems [6]. **Wireless sensor node.** Depend on the using purposes, there are many types of wireless sensor nodes. Each node consists of many optimized modules for saving energy, low-power transceiver by using modern wireless standard as Zigbee, Bluetooth, 6LowPan and sensor array for monitoring environmental parameters. The sensor array includes pH sensor, temperature sensor, humidity sensor, electromagnetic wave detector, photo-detector, gas sensor and other sensor for some factors from the air as CO2, SO2… Telosb, Micaz, Mica2, Arduino, RassberryPi are some of the most popular integrated wireless sensors node, which used widely in research and development IoT application. **Network topology**. A wireless sensor network is a collection of many nodes organized for working in a cooperative network. This network is well designed for lower-power consumption and resource friendly operation. To archive that challenges, many kinds of traditional network topologies were optimized for wireless sensor network as Bus, Tree, Start, Mesh, Ring, Cellular, and Grid [8]. Each of these types has different characteristics and compatible for using in a particular situation. For example, bus topology is easy to install but network congestion and single path transmission. In the start topology, there is a central node works as a sink node, other nodes in the network communicate to each other by using the route through the central node. If the central node has a problem, it will affect for all the network. This topology is compatible for application work as server – client model. In our system, we use tree topology to implement our sensor network.

5 Conclusion and Future Work

Our system is a simple model which implements an IoTs-based application for Smart Garden. By using wireless sensor network under the Collection Tree Protocol, the wireless devices collect basic environmental parameters (temperature, light, humidity) for analyzing through the fuzzy logic controller to give helpful information to the gardeners. Our system also deals with real-time monitoring, scalability and availability by implementing on Open Stack cloud computing. In the future, to improve our system more practically and efficiently we should continue to develop the reasoning engine to be more intelligent as well as upgrade the website more useful by adding some convenient features. We also should research and learn more about the automatic

technology system so that we can combine our system with a modern actuator system in order to generate a full stack "Smart Garden" architecture as a smart solution for the modern agriculture.

Acknowledgement. This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2016-26-01/HĐ-KHCN.

References

1. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
2. Yang, J., Guo, B., Wang, Z.: A study on the rabbit farm environmental monitoring system based on Internet of Things. Adv. Sci. Technol. Lett. **121**, 178–182 (2016)
3. Bajceta, M., Sekulic, P., Krstajic, B., Djukanovic, S., Popovic, T.: A private IoT cloud platform for precision agriculture and ecological monitoring. In: International Conference on Electrical, Electronic and Computing Engineering (2016)
4. Na, A., Isaac, W., Varshney, S., Khan, E.: An IoT based system for remote monitoring of soil characteristics In: 2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect Your Worlds, pp. 316–320 (2016)
5. Kuruva, H., Sravani, B.: Remote plant watering and monitoring system based on IOT. Int. J. Technol. Res. Eng. **4**, 668–671 (2016)
6. Mehta, A., Patel, S.: IOT based smart agriculture research opportunities and challenges. Int. J. Technol. Res. Eng. **4**, 541–543 (2016)
7. Gay, D., Levis, P., Behren, R.V., Welsh, M., Brewer, E., Culler, D.: The nesC language: a holistic approach to networked embedded systems. In: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, vol. 38, pp. 1–11 (2003)
8. Sharma, D., Verma, S., Sharma, K.: Network topologies in wireless sensor networks: a review. IJECT **4**, 93–97 (2013)