International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 8 (2019), pp. 1165-1170

Management of a Smart Factory using Fog Computing

Jinsub Han[1], Insub Kim[2], Hyunmo Yu[3], Hoon Choi[4] *

[1,2,3,4]Department of Computer Science & Engineering,
Chungnam National University, Daejeon, Korea
ORCID: 0000-0002-4435-3997
[*]Corresponding Author

Abstract

As the IoT system consisting of sensor/actuator devices for monitoring industrial facility's safety becomes popular, the network traffic for sensor data is increasing rapidly. Moreover, the workload at the operation & management (OM) server due to these increased data cause another problem of slowing down the server's response, failing to process safety related data in real time. Fog computing can solve these problems. Fog nodes store data collected from sensors of IoT systems. The nodes inspect the data and report to the OM server of only abnormal data. Fog nodes also handle the requests from the OM server by sending specific stored data. This paper describes a fog node applied to a smart factory environment. A simple, prototype implementation showed the feasibility of the fog computing.

Keywords: IoT(Internet of Things), fog computing, smart factory, fog node, software platform

I. INTRODUCTION

As IoT spreads, it is expected that massive data will be produced from various devices. Cisco and other market research organizations projected to create a huge network of more than 50 billion devices in 2020 [1].

As the number and type of devices connected to IoT increases, the amount and type of data to be processed by the operational management server located in the cloud is increasing. Many experts point out the risks of cloud computing architecture where IoT systems are connected to and managed by operations management servers and warn that overload will occur within the next two years.

To solve this problem, fog computing has been proposed [2], which is a distributed computing architecture for efficiently processing large amounts of data generated by devices in the Internet environment.

Fog computing is a distributed processing that stores data and provides networking services at the fog nodes that exist between cloud servers and IoT devices (Figure 1). Fog computing is a method to solve network and server overload problem [3]. Fog nodes placed adjacent to the IoT device store and manage IoT data on behalf of the administration server or the operation & management server at a distance.

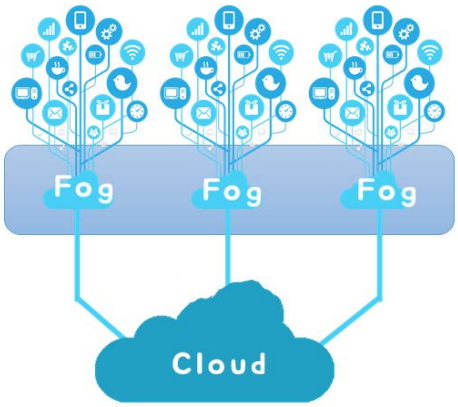

Figure 1: Fog Computing Architecture

There have been many researches and applications of fog computing in various fields such as IoT system monitoring [4], smart home [5], smart office [6], smart city [7], and robot industry [8].

This paper introduces Plant Eyes, a smart factory environment using fog computing. Plant Eyes is a fog node-based software platform that monitors the smart factory's work environment and helps efficient managing of smart factories.

Plant Eyes consists of IoT devices, fog nodes, and an operation & management(OM) server located in the Internet. The fog node receives and stores the sensor data received from the IoT device in the database. The fog node also analyzes the received data. If the node judge the data value is abnormal, it reports this situation to the OM server and controls the IoT devices by itself. Therefore, small fraction of sensor data is sent to the OM server. This reduces the network's data traffic and the server's workload. The OM Server allows administrators/users to view the status of the smart factory via the Web.

This paper is organized as follows. Chapter 2 describes the requirements and design issues of fog computing for a smart factory environment. Then implementation of a prototype Plant Eyes is followed; functions of IoT devices is describes in Chapter 4, and functions of the fog nodes, the OM server in Chapter 5 and Chapter 6 respectively. Chapter 7 mentions the verification of the prototype implementation.

II. FUNCTIONAL REQUIREMENTS OF THE PLANT EYES

Plant Eyes' role is to receive and intelligently process data of industrial facility IoT on behalf of an administrator in the OM server. The requirements are summarized as follows.

(1) Fog nodes should receive data related to environment and industrial facility conditions collected from sensors attached to industrial facilities via wired and wireless communication.

(2) A hardware specification of a fog node may be less powerful than an OM server. A lightweight database is used in a fog node for the case low capability hardware is used for a fog node.

(3) The data received from IoT devices should be analyzed in real time to determine whether the data belongs normal range or not. Upon receipt of the data representing some abnormal situation in the factory, the fog node should report to the OM server.

(4) In above case, the fog node also searches for methods of coping this situation and notify immediately facility workers and local managers.

(5) When the OM server requests data, for example, location information of a specific worker or temperature values of a site for last two hours, to the fog node, the fog node retrieves the data stored in the database that meet a specific condition and transmits it to the server.

In order to show the feasibility of applying fog nodes to smart factories satisfying the above requirements, we developed a prototype fog node as described in Section 3.

III. STRUCTURE OF A PROTOTYPE IMPLEMENTATION

The Plant Eyes system consists of IoT devices installed at the industrial facility, a fog node and an OM server (Figure 2).

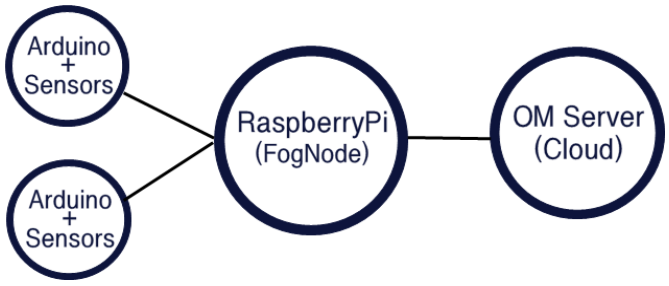

Figure 2: Fog system implementation diagram

Raspberrypi3B+ was used for the fog node, and a laptop PC was used for the OM server (<Table 1>) in the prototype implementation.

Python3 is used to implement the fog node, and JavaScript is used to implement the OM server. <Table 2> shows software environment of the prototype implementation.

Table 1: HW Environment of a Prototype Implementation

	Fog node	OM server
Hardware	Raspberrypi3 B+	Samsung laptop series 9
CPU	1.2GHz QUAD Core Broadcom BCM283 64bit ARMv7 Processor	Intel(R) Core(TM) i7-6500U 2.59GHz
RAM	1GB	8GB
OS	Raspbian 4.14	Windows10

Table 2: SW Environment of a Prototype Implementation

Implementation Environment	Python3, JavaScript
Interface Purpose	Providing smart factory information
Data format	JSON
Input Source and Destination	Arduino sensor, Fog node, OM server
OS	Window, Raspbian
Database	SQLite
Library	BLE, Bootstrap

Communication between the fog node and the IoT device uses a CoAP (Constrained Application Protocol) [9]. The OM server uses datagram socket to communicate with the fog node.

IV. IOT DEVICE IMPLEMENTATION

4.1 Data Collection using the Arduino Sensors

In this paper, Arduino Uno SMD R3 board is used. The main model of the Arduino board offer 14 digital I/O(Input/Output) pins. The 14 pins consist of six PWM (Pulse Width Modulation) pins, six analog input pins that can be mixed with digital I/O pins, and two pins that provide interrupt handling for event processing.

AVR functions as a base module of the ATmega chip mounted in Arduino and provides various libraries used for sensor control. This allows various sensors to be used in Arduino.

The programs that run on Arduino are written in the Integrated Development Environment. The program has a structure of setup and loop. Setup sets the baud rate and pin mode of the serial, and loop is the part where the user program is repeatedly executed. The completed program can be uploaded to the Arduino board via USB.

Arduino defines UART (Universal Async. Receiver/Transmitter) communication as "Serial" and sets the baud rate of the serial with the value set in the setup part by

the user. Arduino can acquire sensor values with UART communication, sensor library and analogRead function.

This feature makes it easy to control external electronic devices such as LEDs and motors.

4.2 Communication with Fog Node

Lightweight protocols for IoT networks include MQTT or CoAP. The Plant Eyes system uses the CoAP protocol, which is suitable for low-power, low-bandwidth and small-sized nodes. It has a feature that can easily convert and interoperate with the HTTP Web protocol because it conforms to the RESTful scheme. Figure 3 shows the request and response process of the CoAP.

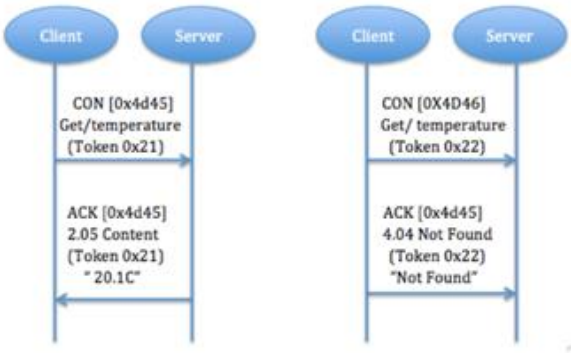

Figure 3: CoAP request / response

A CoAP message is transmitted asynchronously in a datagram transport layer protocol. The Arduino board used in this prototype does not support Ethernet for UDP communication. Arduino, an IoT system, is connected to the Raspberrypi3, a fog node, via Ethernet shield (Figure 4).

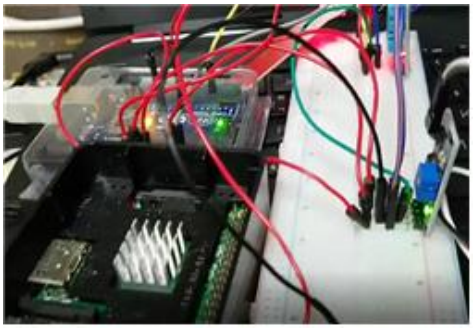

Figure 4: Connection of Arduino and Raspberrypi3

The CoAP protocol works in client-server model. The server and client are configured using Python's Txthings which is implemented based on the Twisted framework. The Twisted framework has most of standard protocols as libraries available on the Internet, and it provides protocol classes and Deferred, Callback, Error Back, Callback Chain, and Reactor. The server creates a resource tree, manages resources with init, render_GET and render_PUT for each resource, and processes messages received from the fog nodes.

4.3 Control of the IoT Device

In order to demonstrate the control message received by the Fog Node, LED and DC motor are connected to Arduino as shown below (Figure 5).

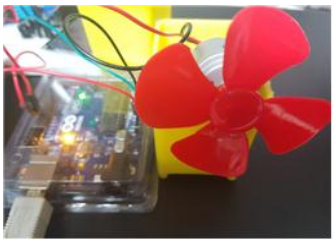

Figure 5: Environment connected with DC motor propeller

The CoAP protocol's put message is used for controlling the Arduino. The server checks the payload of the put message, encodes the character according to the payload, and then writes to the serial and sends it to the Arduino. In Arduino, it checks if Serial.available is true for write. If it is true, encoded letters are classified. If the payload value of the put message is 'motorON', the DC motor connected to the Arduino is activated (Left of Figure 6).

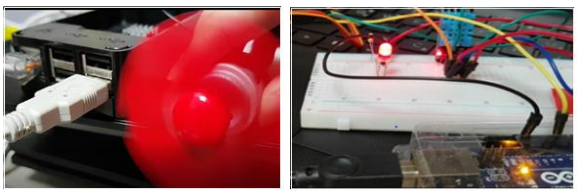

Figure 6: Operation of DC motor propeller and LED light

The LED also lights when the payload value of the put message is 'ledOn' (Right of Figure 7), and turns off the LED when 'ledOff'.

V. IMPLEMENTATION OF THE FOG NODE

5.1 Communication with Arduino

We configured the client using Twsited's Call back, Error back, defer, and reactor. The fog node sends a CoAP Get Request message and identifies the server's resources through request.opt.uri_path. It accesses the IP address and the CoAP port of Raspberrypi3 using the remote, and receives a response from the server through the Call Back.

We configured the endpoint, protocol, and client before running, and connected the port to the reactor, and then ran it to cycle through the loop periodically. When the agent is running through the reactor, it sends a Get message to the CoAP server via the requestResource in the class, requesting the sensor data value. It shows the sensor data received through the PrintResponse.

5.2 Data Management

Fog node receives data from hundreds of sensors of temperatures, humidity, CO_2, and noxious gas concentrations

in IoT devices of the smart factory. The fog node temporarily put received data in main memory, then checks abnormality in the data. Only in case that the data are evaluated abnormal, the fog node report the server. The node periodically stores the data in the local storage.

The fog node has a function of searching the data stored in the node's database in response to the OM server's request. The server may request the fog node to send the data of all sensors during a given time span or requests data of a specific sensor of a certain time. The fog node accesses its local data and transmits the corresponding data to the OM server.

5.3 Autonomous Device Control

Temperature management in smart factory is important for the safety of workers. It is also closely related to the life of machine, which is essential for running smart factory. Therefore, a smart factory may be equipped with an air conditioner, a heater, and other thermostats. Since the proper temperature of smart factory varies from each zone(area) or each time period, it must be managed separately.

Plant Eyes' fog node is able to handle IoT devices real-time based on the data generated by sensors in smart factory. The fog node learns the normal temperature profile for each zone based on data collected for a long time period. The fog node was implemented to read values from the thermostat sensor and controls an air conditioner or a heater accordingly.

The fog node may also control some IoT device in response to the OM server's request. The server may request the fog node turn on or off a specific IoT device in the factory.

This is flexible and efficient because each fog node installed in each zone of smart factory finds abnormal temperature and handles this situation quickly without human support. The node does not transmit normal data from IoT sensors to the OM server, but informs the situation of the fog node. If the air conditioner is activated, it sends to the OM server only the information that the air conditioner has been turned on.

5.4 Quick Respond to Risky Situation

In the event of an accident or device malfunction in the smart factory, the fog node detects it and notifies the administrator. It is important to take quick action in a risky situation. In the fog computing structure, it is possible for a fog node to react and cope more quickly than the remote OM server. When the fog node detects a dangerous situation or data abnormality, the fog node first controls appropriate IoT devices, without having to get approval from the OM server. The fog node reduces chance to lead to significant damage in emergency situations thanks to rapid measures and response.

5.5 Diagnosis of Emergency using OpenAPI

When the fog node recognizes a dangerous situation during the process of received data as shown in Figure 8, the fog node first sends an alarm to the workers. Then the node refers to the public web site such as [11] provided by the government to find out a countermeasure for the situation.

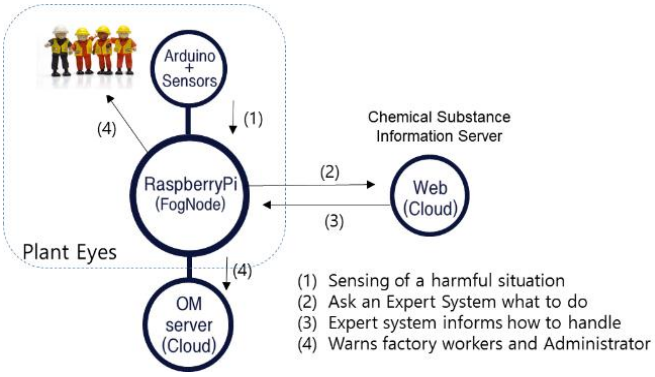

Figure 7: Risk situation response process

When a toxic gas or a dangerous chemical is spilled, it is difficult for a fog node or the OM server to find out what kind of chemical substance it is and how to cope with it. Therefore, the fog node better refer to an outside expert system and inform the site manager how to deal with this dangerous substance. The factory facility may be controlled according to a guideline provided by the expert system. Then the node reports to the OM server (Figure 7).

5.6 Data Communication with the OM Server

In this study, we used DDS(Data Distribution Service) as the communication middleware between the fog nodes and the OM server [13]. DDS is suitable for exchanging large amount of data between communication nodes in real time in distributed computing environment. (Figure 8).

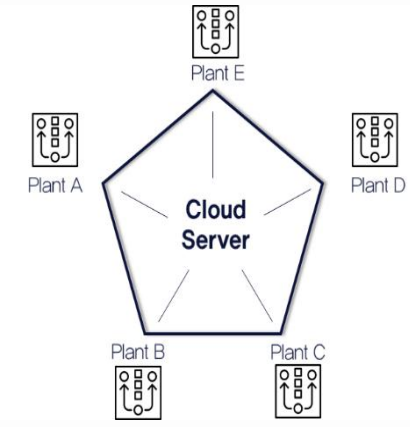

Figure 8: Communication with fog nodes to Operations Management Server

The fog nodes exchange data with the OM server in a DDS topic defined by the fog management application.

VI. IMPLEMENTATION OF THE OM SERVER

6.1 Requesting Stored Data to Fog Nodes

Figure 9 shows a web screen of the OM server.

Figure 9: Operations Management Web

Human administrator can click a zone of the factory to request sensor data real-time. A request message including the smart factory ID and the zone ID is sent to the fog node. The fog node parses the request message and replies sensor data of the corresponding zone to the OM server.

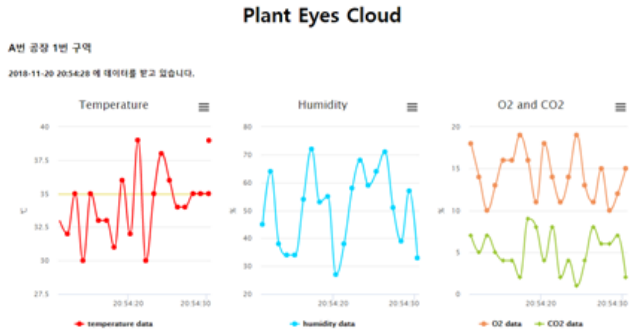

Figure 10: Real-time graph of sensor data installed in a specific area

The response data consists of sensor ID with its value such as temperature, humidity, carbon dioxide, oxygen concentration, and date and time of data update. They are drawn in the form of graph in Figure 10.

6.2 Request for Hardware Control

The OM server displays each zone with the names of installed IoT devices and their operational state. Administrator can click the on/off icon of a device to turn it on/off. Then the OM server sends a request message including the factory ID, zone ID, device name and command(on/off) to the fog node. The fog node parses the requested data to turn on or off the corresponding IoT device in the zone. Upon successful completion of the control, the fog node replies on/off state change back to the OM server. On the OM server, the new state is displayed on the monitor.

6.3 Display and Notification of Risk Data

When a fog node detects a risky situation based on the sensed data from IoT devices, and notifies it to the OM server, the OM server emits a warning sound with a warning screen to inform a human administrator.

VII. PERFORMANCE MEASUREMENT

A simple performance measurement has been carried out in order to show the usability of fog computing. The time for the fog node to receive, evaluate abnormalities of the sensor data and control a IoT device is measured. This latency time is compared with the latency time of OM server doing the same jobs. We made the temperature rise on purpose and measured the delay using the 'time' function from time point the fog node receives the temperature until the on or off command is sent out by the fog node. The same delay is measured at the OM server.

It is reasonable to think that the computing power of the fog node hardware is lower than the OM server's. Therefore, we used raspberrypi3 (B+) for the fog node and used a PC (i7) for the OM server (Figure 11).

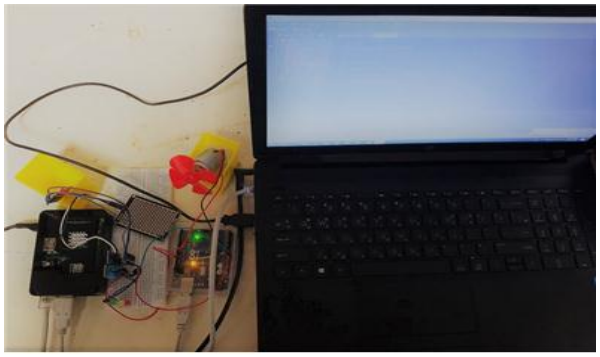

Figure 11: Prototype Implementation

Delays were measured by using the time function from time when the sensor data enters the fog node and the data is stored, processed until a command to control the air conditioner is sent to the sensor. These delays are processing time of the fog node and denote the response time to the sensor. The same delay times were measured without the fog node, i.e., when the sensor and the fan are connected directly to the OM server. These delays denote the processing time of the OM server.

Response time of the fog node is the time measured from when the sensor data were received till the time when the command to control the air conditioner is sent out. As mentioned in Section 1, fog nodes are usually located near to the IoT sites. It is obvious that the distance of the network link between a fog node and the devices is much shorter than the distance from the fog node to the OM server. Data

transmission delay in the network is proportional to the network distance. It is reasonable to assume that the network delay between the devices and the fog node is very small compared with the delay between the device and the OM server. Therefore, we assumed in this experiment that the network delay between IoT devices and the fog node is 0, and considered three cases where the OM server is 2, 5, and 7 seconds away from the IoT devices.

Table 3 shows the response times. On the case of the fog node, it took about 3 seconds. Because the fog node hardware used in this prototype implementation is a low-end raspberrypi3, the response times are rather long. High-end hardware of actual system will provide much shorter response time.

Table 3: Response times of fog node and OM server (initial condition: air conditioner OFF)

Mean Response Time of Fog Node	Mean Response time of OM server		
	Network delay	Server processing time	Sum
3.10s (air con. ON)	2s	1.02s (air con. ON)	3.02s
	5s	1.02s	6.02s
	7s	1.02s	8.02s

The OM server has been experimented under the same condition. The OM server operated in 1 second when the air conditioner is turned on, much faster than the fog node. Including the transmission delay, total response times become longer than the fog nodes'.

The actual values of Table 3 may not be significant because we used assumed values of network delay. But we can argue that if decent hardware, like the one used in a personal computer, is used for the fog node, the processing time at the fog node may be similar to that of the OM server. Then response times depend on the network propagation delay. Since the OM server is far from the IoT devices, the response times of the OM server become longer than the fog node's response time.

There is another factor, though we do not show it by experiment, that the fog node's response time becomes faster than the OM server's. If not for fog nodes, all the traffic will concentrate to the OM server, resulting in high workload at the server. High workload makes the server respond slow.

In summary, fog nodes can monitor and control the IoT devices much faster than the OM server does.

VIII. CONCLUSION

Fog nodes store operational data collected from sensors of industrial facilities, smart factories, and so on. The nodes inspect the data and report to the OM server of finding abnormal data. They contribute to prevent unnecessary data from being transmitted to the OM server through this monitoring process. Fog nodes also handle the requests from the OM server by sending specific stored data.

This paper described Plant Eyes, a smart factory environment using fog computing. Plant Eyes consists of IoT devices, fog nodes, and an OM server located on the Internet. A simple, prototype implementation showed the feasibility of the fog nodes. Experimental results show that the fog computing method of processing data at the source using the fog nodes is very efficient. Therefore, fog nodes are expected to provide reliable data management for industrial facilities, smart factories distributed nationwide.

ACKNOWLEDGEMENT

This work was supported by research fund of Chungnam National University.

REFERENCES

[1] Zaigham Mahmood, "Fog computing: Concepts, Frameworks and Technologies",p3 ,2018.

[2] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog computing and Its Role in the Internet of Things", Proceedings of the first edition of the MCC workshop on mobile cloud computing, August 2012.

[3] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks", Telecommunication Networks and Applications Conference (ATNAC), 2014 Australasian. IEEE, pp. 117-122, 2014.

[4] G. Yoon, D. Choi, J. Lee, H. Choi, "Manage-ment of IoT Sensor Data using a Fog Computing Node", Journal of Sensors, Feb. 2019.

[5] S. Yi, Z. Hao, Z. Qin and Q. Li, "Fog Computing: Platform and Applications," 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), pp. 73-78, 2015.

[6] J. Li, J. Jin, D. Yuan, M. Palaniswami and K. Moessner, "EHOPES: Data-centered Fog platform for smart living," 2015 International Telecommunication Networks and Applications Conference (ITNAC), pp. 308-313, 2015.

[7] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, Q. Yang, "A Hierarchical Distributed Fog computing Architecture for Big Data Analysis in Smart Cities", Proceedings of the ASE BigData & Social Informatics 2015, October 2015.

[8] Gartner, "Fanuc. Cisco, Rockwell, PFN Push 'Edge and Fog Computing' Into Factory", https://www.gartner.com/doc/3290826/fanuc-cisco-rockwell-pfn-push.

[9] Xi Chen, "Constrained Application Protocol for Internet of Things", https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap, 2014.

[10] Object Management Group. OMG, "The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol", version 2.1, 2010.

[11] Open Data Portal, Ministry of Environment _ Chemical Substance Information, https://www.data.go.kr/dataset/15029194/openapi.do.