Sensors and Materials, Vol. 31, No. 6 (2019) 1815–1829
MYU Tokyo

S & M 1901

Interactive Task Assignment Model for Internet of Things Devices

Tse-Chuan Hsu,[1] Yao-Hong Tsai,[2*] Dong-Meau Chang,[3]
Fu-Yao Liu,[4] and Chih-Hung Chang[5]

[1]Centre for Creative Computing, Bath Spa University, England/Hsuan Chung University, Taiwan
[2]Department of Information Management, Hsuan Chung University, Hsinchu, Taiwan
[3]Longyan University, School of Information Engineering, Fujian, China
[4]Department of Tourism Management, Tourism School, Sanming University, Sanming, China
[5]College of Computing and Informatics, Providence University, Taichung, Taiwan

(Received January 9, 2019; accepted March 12, 2019)

Keywords: Internet of Things, design algorithm, network communication

In the application of the Internet of Things (IoT) technology, devices may passively receive signals for service or proactively provide data. The main problem in the communication mechanism for an IoT device is how to use the communication mechanism to trigger the wake-up device to exchange data. In the past, IoT devices were mainly operated at the remote end. After the user delivered the service demand task, a one-to-one device operation was possible through network communication. Owing to the changes in technology and demands today, the communication infrastructure has been transformed from a one-to-one approach to a one-to-many model, and even a many-to-many model has evolved. The operator no longer only performs a single operation on each device, so it is necessary to make the device itself gradually intelligent through the calculation of the communication design algorithm, which is a major technological breakthrough. In this study, we will show how to use data communication computing algorithms and communication systems to build intelligent architectures and proactively respond to services across different devices. In addition, using a wake-up detection system, it is possible to avoid the problem that the device responds after receiving the task and delays the delivery of the task result. Experimental results show the efficiency of the proposed method. The variance of real response time was limited within a small interval as the number of devices was increased.

1. Introduction

In network communication, the intelligent communication system has been used as a mature technical service for many years. It can be used to assist network communication packets to intelligently find routes for communication transmission. The collision detection technology of the communication packets and the traffic service quality management algorithm can smoothly transmit the digital data in the packets to the destination end, ensuring that the data transmission packets are not lost.

*Corresponding author: e-mail: yaohong.tsai@gmail.com
https://doi.org/10.18494/SAM.2019.2254

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

In the Internet of Things (IoT) technology, if a remote operation is required, the device or robot can assist the user in the task. In addition, when a large number of devices must operate at the same time, the agent can be used to centrally manage the operating system using a single platform to operate a large number of devices. Take the current smart home system as an example. When smart homes operate smart curtains, lights, TVs, refrigerators, and washing machines through a single remote control, they will be connected via the Internet. The user sets up the service serial script through a single remote control and rapidly links the entire communication service script through a push-button operating device. Therefore, the user's demand for multidevice operation can be further completed.

However, this mode is limited to the passive operation of the device, and it is impossible to perform a two-way active response operation. The data response operation between the detailed devices cannot often be obtained or responded to by the server for subsequent operations. At present, it has been developed in automation technology for many years, especially the IoT technology. However, in the IoT technology, there is no good communication design service architecture. In most IoT models, the active one-to-one operation of hardware device system services is mainly performed by humans. Particularly when service applications must be implemented through IoT, it is necessary to expand to a large number of device operations, such that it is impossible to operate at the same time.

Therefore, an intelligent computing task model algorithm is proposed in this study, which allows the automation of a single application architecture for management, corresponding to a unified communication platform, and the establishment of all object devices that can automatically communicate with the platform. When the demand side issues an operation command to the platform, and all the objects automatically communicate with the platform, the assigned task results are processed separately.

To solve the above problems, in this study, an innovative IoT service intermediary communication station that can provide fast communication between devices is proposed. Through the proposed MQTT server platform,(1,2) the design establishes a set of innovative wisdom logic based on the communication between the devices. This will let the device know what role it is assigned to and what it will do throughout the IoT environment. After the device receives the task dispatched by the MQTT server, it can remotely accept the update script and service job software. This will allow different devices to contact the platform to understand what they should do. At the same time, the installation kit required for the task is dispatched to help establish a new type of communication service architecture in the IoT environment.

In this study, the following two important breakthrough technologies were successfully proposed: 1. verifiable communication platform mechanisms and methods, including how to enable hardware devices to communicate with the platform, establish an innovative automated communication feature model, and flexibly exchange the communication model with the transfer method and exchange the data content (Response to the event service can be effectively carried out through the network) and 2. a technology platform for software management and a software design method to assign the task assignment work to the hardware device through the platform. By using decentralized objects for data processing and calculation, the mode can reduce the speed and performance of high-speed computers under centralized management.

Through the establishment of a new IoT platform, IoT and cloud fog architecture, and a system performance simulation test, we can accelerate the speed of distributed data processing and reduce network transmission costs.

In Sect. 2, we explain the current developments in related technologies. In Sect. 3, we provide a description of the research structure. In Sect. 4, we show experimental results. Conclusions are provided in Sect. 5.

2. Background

The basic three-layer architecture of the IoT system includes an application layer, a network layer, and a sensing layer. At present, the sensing layer has been developed for many different sensing components, including those for many daily necessities, such as smart bracelets, Bluetooth weight scales, and Bluetooth sphygmomanometers, all of which have reached the consumer goods commodity stage. In the network and application layers, devices are independent individuals, lacking an effective integrated management platform, including devices communicating with each other, and actively analyzing feedback suggestions between devices[4] as shown in Fig. 1.

IoT-based applications are included in various IoT applications such as smart cities, homes, and hospitals. Their outputs from data collection and analysis results help improve some unobserved environmental information. In IoT applications, a large number of hardware devices are required to receive data and perform network connection communication applications. The large number of incoming data streams generated by many IoT devices will cause severe network congestion and high overloading of the server. This results in communication delay and performance degradation. Some fields use IoT services, because the last network communication is as yet underdeveloped. The data could not be returned efficiently.[3]

On the basis of the three-layer structure, many related digital sensing elements have been developed in the current research of the sensing layer. This structure is used to receive data

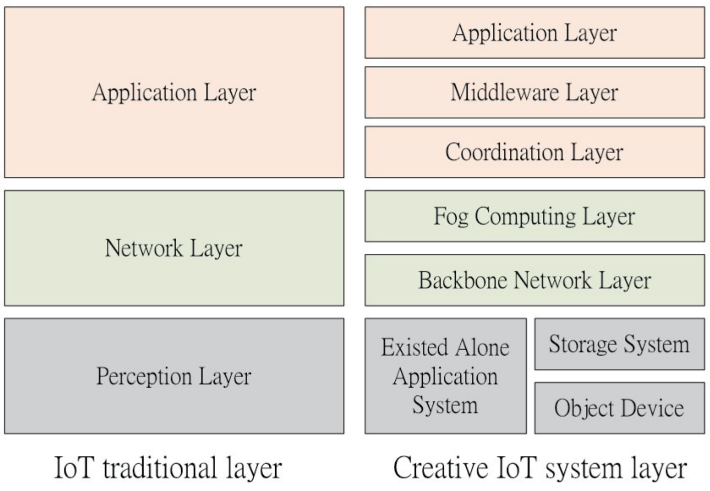


Fig. 1. (Color online) Three-layer IoT architecture.

related to the IoT environment. The communication and application of the network layer focus on how to make the data transmission from endpoint devices more flexible and efficient under the existing communication network. When the required detection event occurs, the value change information is transmitted to the server by using the network delivery algorithm, which facilitates the instant analysis and processing of the data.

Our research team published an improved transfer calculation algorithm for network communication processing and applied it to the IoT system. This algorithm is used to calculate the time of waiting for the response to the machine-to-machine (M2M) message, and the calculation result is used as a criterion.(5–7) When the server delivers a task to all external IoT devices, the endpoint device performs event processing and returns the event processing result after receiving the task. If the result response average is higher than the assumed standard time criterion, the frequency and number of times the device is required to return the data gradually decrease. After the successful application of the research, a creative IoT platform will wait for the data to be returned for system quantitative management and gradually reduce the cost of requesting a fast and immediate data transmission.

In this study, we continue the previous M2M result of the research team and develop an enhanced marker regression calculation algorithm. Through the OM2M model, the machine learns the environment, strengthens the correction in the communication logic design, dispatches the task and software service script, and provides a timer synchronization scheme to calculate the wait mark on the server side. When the endpoint device fails to give back the obtained task information after time expiration, the endpoint device is forced to return the currently analyzed result first and the computing service script is restarted. This calculation model can overcome the difficulty in collecting any subsequent data information when the endpoint device is overdue from the previous result.

3. System for M2M Fog Computing Design

The communication technology discussed in the IoT system has historically can only complete data transmission through the Internet, and there is still no effective system for information collection and application. When the cloud computing mode is performed, devices can be shared by dynamically configured calculations. If the IoT device can expand the I/O device to the sensing layer, the data detected by the device can be integrated with the cloud. This allows the sensing device at the fog end to complete information gathering and real-time operations. Therefore, it is possible to provide more types of changes in the timely dynamic management and intelligent analysis of various items in the IoT.(8) Guinard *et al.* combined cloud computing technology with IoT's EPCglobal Network technology architecture to build an EPC cloud and apply it to electronic goods security systems.(8) Connected via the cloud platform, the IoT system attempts to systematically implement the SaaS system through the cloud technology of the IoT device and successfully solves and overcomes the difficulties encountered in remote management.

In this study, we use the M2M communication structure developed by OSGi to construct the message server as a task dispatch server. All endpoint devices are end nodes. When the service we design is delivered to the endpoint through the message server, the device itself wakes up the data exchange and task assignment services. According to the order of assignment, the information of the task script and the result of the script execution are synchronized. When the device has program update or data exchange requirements, it can be dispatched by M2M, and its service scope can be more extensive. Therefore, in this study, we explore an IoT technology that can be used to improve the communication management of a large number of devices. All terminal IoT devices are built with the "Fog" platform computing model and provide a layer of fog between the terminal device and the cloud data center. As with the current construction of many memory card storage or small servers or routers, some data that do not need to be placed in the cloud are directly processed and stored at this layer. This can greatly reduce the computing and storage load of the cloud, improve efficiency, increase transmission rate, and reduce latency analysis time.

The conventional Internet protocol is designed for transmission modes that do not need to be permanently connected and have few connections, but have a large amount of data. For consistency, IoT is just the opposite and must maintain the connection status frequently, and the number of connections is extremely large, far exceeding the number that existing telecommunication equipment can handle, and the amount of data is relatively small. Therefore, under the state of different transmission modes, the devices used in telecommunications or the Internet in the past cannot support the needs of the IoT era. For example, when using IoT as an access control system for an office, you need a service that can continuously maintain network connectivity and communicate any time. Although it is not a conventional high-speed computing network environment, this system cannot tolerate a short-time network disconnection, because it will cause problems with the normal opening and closing of the office door.

3.1 Designed communication framework

The task of the network layer is not only to integrate and transmit data, but also to have a certain degree of information processing capability. It can handle emergencies immediately or respond to simpler system requirements on its own. It works like a human's "reflex nerve" and can respond to specific conditions on its own, without having to pass all the data to the brain (i.e., the IoT data center) for interpretation. In this manner, not only can the correct response be made in the first place, but also the computing burden of the overall network and data center can be reduced. For example, if a node on the power network detects that the device is abnormal and may be dangerous, it can stop the power supply by itself and send the related information up to the data center of the smart power network for subsequent processing.

From the past IoT device remote operation technology, although the device serial connection is integrated into a single platform system, and multiple device connection control is performed on a single platform, the device autonomous computing and communication capability is still lacking. The concept of device autonomic computing originates from the fact that when humans deal with an event, they will automatically perform a number of different tasks processed through the brain, so that the event can be completed rapidly after its initiation. For example, when a person wants to drink hot water, he takes a cup to the water dispenser to get it. If he is accidentally burned by the hot water, the brain's computing power will cause the hand to rapidly drop the cup and makes both hands stay away from the hot water, and the person looks for cold water to cool them.

If the above actions are implemented through the IoT device, this series of actions must have a neural operation function on each end point, such as a network sensor on the finger area of the brain, to achieve this. Since the device can only receive commands in one direction and then provide services, the service information must be centrally managed. There has recently been a significant improvement in service performance management. However, if you want to make the function of each endpoint as the human finger in the service operation, the device should independently reflect the operation and a mechanism for the device to receive the command and process it immediately should then be designed.

In this experiment, the M2M architecture is implemented in the basic communication layer, and communication integration is performed through fog computing, so that task assignments are assigned to all device nodes. Just like the operation of the human hand and foot via the nervous system, all the scattered device systems can instantly return the results to the server, which can rapidly disperse the operation and reflect the state of neural reflection. Therefore, in this study, we continue the past experience, applying the feedback time calculation construction to the IoT cloud integration architecture, storing data through the cloud, and using fog for fast distributed computing. At the same time, the intermediary system is used to integrate the cloud system with IoT, and the solution can be provided in all the different devices. The device side can actively analyze and capture information and provide data information for distributed computing and storage management, and the cloud performs synchronous storage management of necessary data to accelerate IoT calculation speed. An example of a fog service with the IoT platform is shown in Fig. 2. On the left of Fig. 2, data collection and analysis services for all devices are gathered to the fog computing storage service and then transmitted to the cloud. Users deploy analytics to the IoT platform through the cloud, which are shown on the right of Fig. 2.

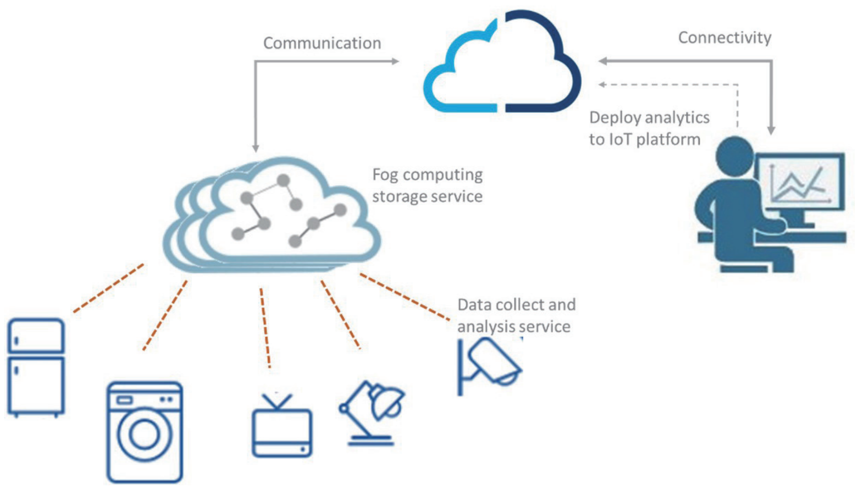
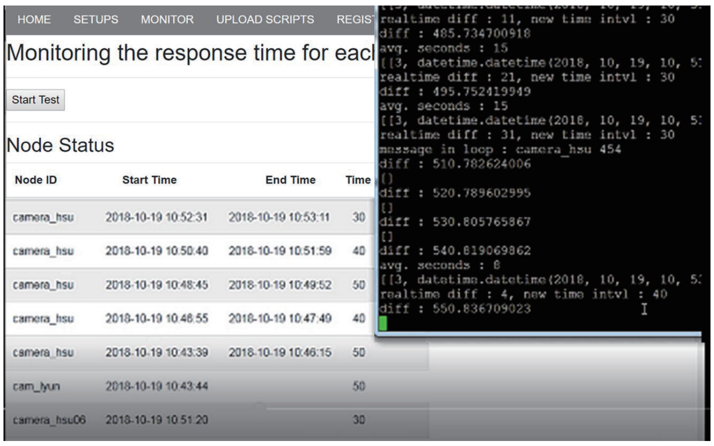

Communication
Connectivity
Deploy analytics to IoT platform
Fog computing storage service
Data collect and analysis service

Fig. 2. (Color online) Example of fog service with IoT platform.

3.2 IoT client device monitoring

The network state management information exchange service improves the service performance. If a large amount of data is required to be sent back in time, a more expensive network service is required in the state of the device service. For example, when the video recorded through the monitor system is transmitted back to the server, the required network performance cost is high, especially when the environment of the monitor system is, for example, a wider road monitoring field. Assuming that there are about eight monitors at each road intersection, all the video images are transmitted back to the centralized server management center through the dedicated network and then recorded by the management center computer. The network usage cost is very high. If the proposed research is combined with the decentralized calculation model to create a decentralized cloud monitoring calculation model, all devices can use the platform to send analysis tasks to all IoT devices that communicate with the platform and then through the IoT. The device performs calculations and the analysis results are transmitted back to the platform. This method will reduce the traffic usage of the network.

In addition to reducing the frequency of the flow operation service program, in this study, we further explore how to efficiently establish a response feedback communication module. When the server performs task assignment and waits for a return message, we add a calculation engine to calculate the time of message transmission and gradually postpone the delivery of the task time in response to a good response. In the service state model, data synchronization calculation is performed. This method successfully reduces the time wasted and cost caused by a large number of distributed devices simultaneously returning information. It has been found through research that after the algorithm is executed multiple times, a certain technical obstacle occurs. In a system environment where the performance is poor, the system waits for the first response for a very long time, and the device will not return the execution result because the packet transmission is missing. After the decentralized tasks sent by the system arrive at each node, most nodes complete the response, and some devices with poor performance still have no response after the completion of all tasks, as shown in Fig. 3. It is obvious that the lack of the algorithm should be improved for better performance.



Monitoring the response time for eac

Node Status

Node ID	Start Time	End Time	Time
camera_hsu	2018-10-19 10:52:31	2018-10-19 10:53:11	30
camera_hsu	2018-10-19 10:50:40	2018-10-19 10:51:59	40
camera_hsu	2018-10-19 10:48:45	2018-10-19 10:49:52	50
camera_hsu	2018-10-19 10:48:55	2018-10-19 10:47:49	40
camera_hsu	2018-10-19 10:43:39	2018-10-19 10:46:15	50
cam_lyun	2018-10-19 10:43:44		50
camera_hsu06	2018-10-19 10:51:20		30

```
realtime diff : 11, new time intvl : 30
diff : 485.734700918
avg. seconds : 15
[[3, datetime.datetime(2018, 10, 19, 10, 5
realtime diff : 21, new time intvl : 30
diff : 495.752419949
avg. seconds : 15
[[3, datetime.datetime(2018, 10, 19, 10, 5
realtime diff : 31, new time intvl : 30
message in loop : camera_hsu 454
diff : 510.782624006
[]
diff : 520.789602995
[]
diff : 530.805765867
[]
diff : 540.819069862
avg. seconds : 8
[[3, datetime.datetime(2018, 10, 19, 10, 5
realtime diff : 4, new time intvl : 40
diff : 550.836709023
```

Fig. 3. (Color online) Camera_hsu receives the response but not cam_lyum.

Therefore, in this study, we design new calculation methods and adjust them. In addition to the fact that the devices with better performance should be calculated within the average value, a new set of events for time reduction calculation methods is added. When the system detects that the service response time is very long, the system cannot determine whether it can effectively contact the device. An S1 event is defined in the model. When the waiting time exceeds A s, the system sends an S1 event requesting communication detection. The IoT platform actively sends a confirmation service status detection message and waits for a response. If the service message response time is still more than A s, the system writes the service time back to the database for recording, displays the device abnormal signal in the platform, and records the delay frequency as N times. When the event occurs for the second time delay, the system initiates multiple times to send a message detection, performs $(N - 1) \times A$ for synchronous message detection calculation, and gradually detects the service status, and records the delay frequency. The flow chart of service detection with delay frequency is shown in Fig. 4. An example of the IoT backward algorithm with a delayed message is shown in Fig. 5 and the response time for IoT is shown in Table 1.

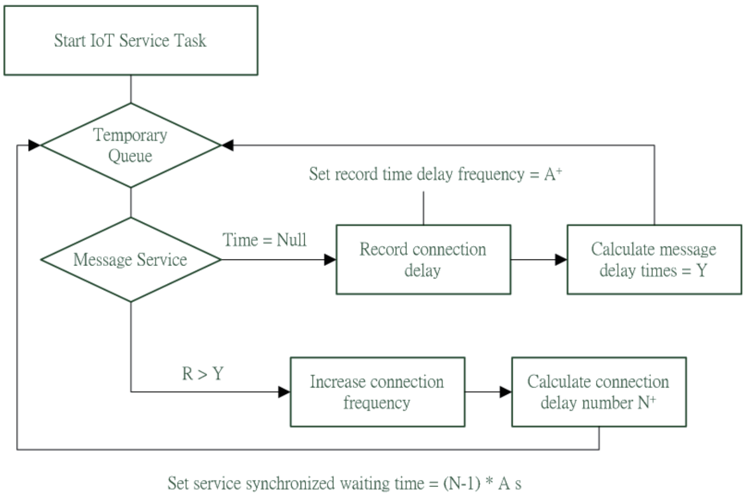

Start IoT Service Task
Temporary Queue
Set record time delay frequency = A+
Time = Null
Message Service
Record connection delay
Calculate message delay times = Y
R > Y
Increase connection frequency
Calculate connection delay number N+
Set service synchronized waiting time = (N-1) * A s

Fig. 4. Service detection with delay frequency.

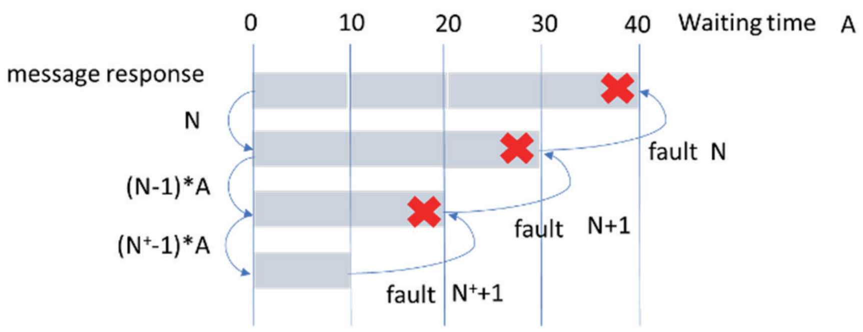


Fig. 5. (Color online) IoT backward algorithm with delayed message.

Table 1
The response time for IoT in Fig. 5.

Fog Device Timer	N	Z	Z+(N-1)xA s	N^+	A^+ = Next Check Timer	Y=Average Respones	R. Real Response > Y
1	Null	50	50-(Null)*10	0	50	50	53
2	1	50	50+(0-1)*10	-1	40	(53+58)/2=55.5	58
3	-1	40	40-(0-1)*10	-1	30	(53+58+62)/3=57.6	62
4	-1	30	30-(0-1)*10	-1	20	(53+58+62+65)/4=59.5	65
N	(~+1)	…	…		…		…

* if N=Null N+=1 A+=A
else if (N-1)*A=0 N+=(N+1)

By adding a new backward calculation model, when the average feedback time is lower than the set target value, the system reduces the time in seconds required for the next startup to request the device. Assuming that the device performance processing status is poor, in order to avoid a very long waiting time, the processing request event return frequency is increased. To reduce and avoid the poor processing speed of the device, all IoT devices transmit data at the same time.

In this study, we summarize the different monitoring construction environments into one-to-many and one-to-one situations, as shown in Fig. 6. In a one-to-many environment, only one monitoring program (thread) is used throughout the system to monitor all applications. The advantage of this architecture is that the monitoring program consumes less system resources and has less impact on the application. However, as the number of applications that must be monitored increases, the accuracy and immediacy of monitoring will be relatively low. An overworked monitoring program cannot collect and provide immediate monitoring information. In addition, a one-to-many architecture system requires constant switching and exchange of information between monitoring programs and different monitoring targets, which also greatly increases the workload of the system. However, in a one-to-one architecture, each application is monitored by a monitoring program; in this case, the immediacy and accuracy of monitoring can be guaranteed, but a large number of monitoring programs will also cause the system to spend numerous calculation resources. Figure 6 shows the one-to-many and one-to-one threads.

Therefore, multiple executions are synchronized. When the IoT device monitors the performance, the delay request feedback response time decreases, and the analysis and calculation event processing frequency also decrease. Conversely, in a device with poor performance, the frequency gradually increases, the feedback time for data analysis shortens, and the analysis of the feedback result accelerates.

4. Experimental Results

In this study, the MQTT server was set up in the Amazon EC2 server to establish a communication intermediary platform and simultaneously construct a creative service platform to simulate the component operation behavior performed by the operator when updating the service event. To meet the communication needs, the IoT device was constructed on a Raspberry Pi to conduct experiments. The Raspberry Pi model B+ V1.2 communication version was used with the Ubuntu operating system, and the M2M service agent is established in the Ubuntu environment. Figure 7 shows the platform of the proposed fog system.

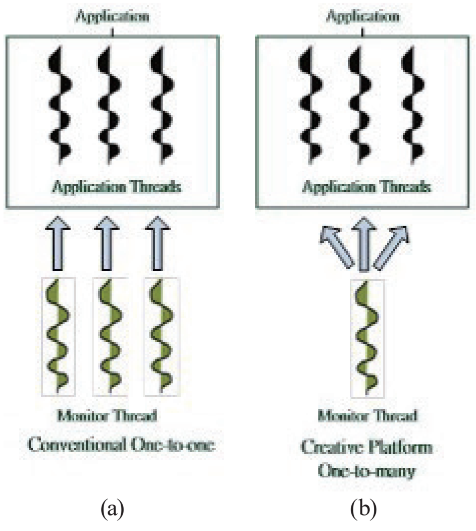


Fig. 6 (Color online) (a) One-to-one and (b) one-to many threads.

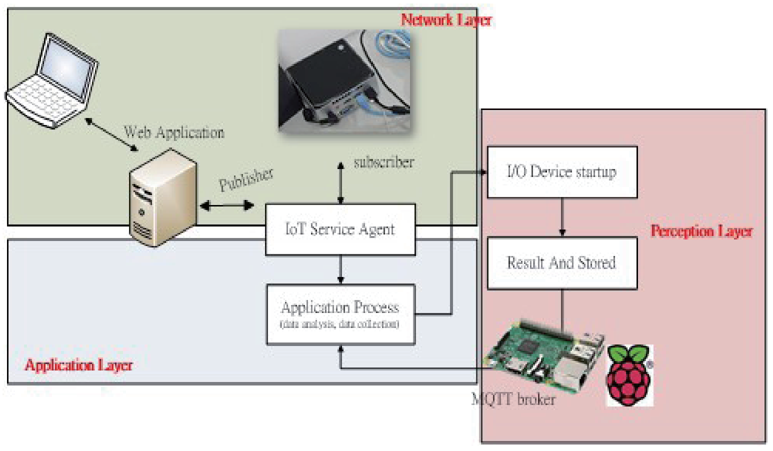


Fig. 7. (Color online) Platform of the proposed fog system.

After the device starts the service, the system will actively communicate with the MQTT server and assist the task assignment through the message server. We simulate that the device is required to take photos and perform digital image analysis operations to analyze whether someone is in the image. When someone enters the image, the system actively records and waits for the delivery to be sent to the MQTT server, simulating the establishment of a security alert protection network. In this study, several Raspberry Pis were built to simulate the fog unit. Each device is an independent computing entity. In addition to collecting information from the sensing device, it also performs image analysis and calculation, reducing the cost of transmission from all image events back to the system and the new platform. Figure 8 shows the experimental devices used.

In this study, several image indicators were assumed. First, we ran a single-machine test to calculate the average time in seconds to analyze ten digital images in the Raspberry Pi analysis.

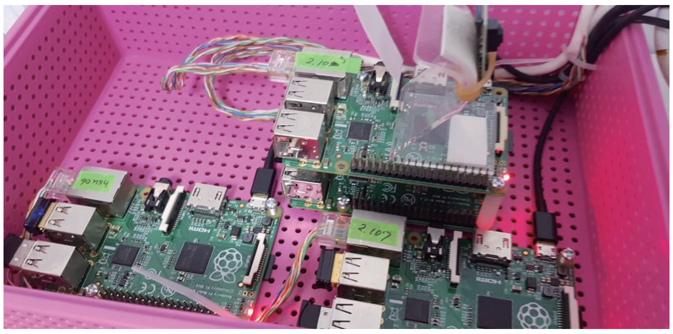
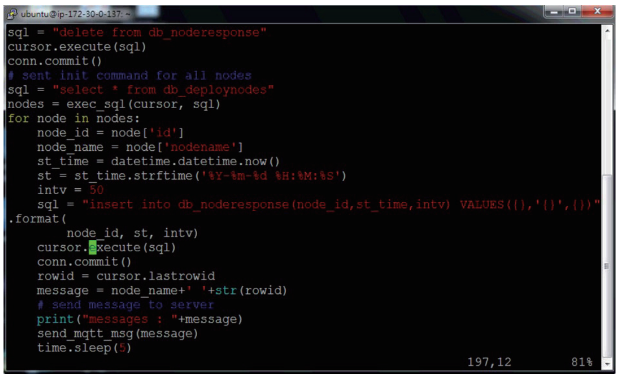
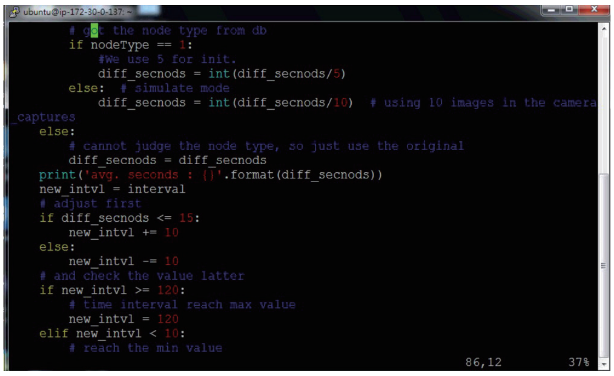


Fig. 8. (Color online) Experimental devices.



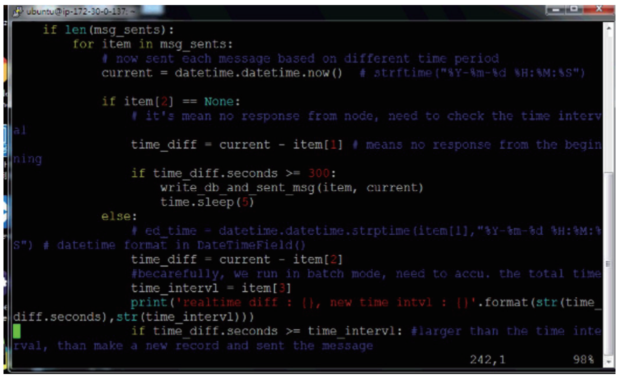
```
sql = "delete from db_noderesponse"
cursor.execute(sql)
conn.commit()
# sent init command for all nodes
sql = "select * from db_deploynodes"
nodes = exec_sql(cursor, sql)
for node in nodes:
    node_id = node['id']
    node_name = node['nodename']
    st_time = datetime.datetime.now()
    st = st_time.strftime('%Y-%m-%d %H:%M:%S')
    intv = 50
    sql = "insert into db_noderesponse(node_id,st_time,intv) VALUES({},'{}',{})"
.format(
        node_id, st, intv)
    cursor.execute(sql)
    conn.commit()
    rowid = cursor.lastrowid
    message = node_name+' '+str(rowid)
    # send message to server
    print("messages : "+message)
    send_mqtt_msg(message)
    time.sleep(5)
                                                              197,12        81%
```

Fig. 9. (Color online) Setting threshold to 50 s.



```
        # got the node type from db
        if nodeType == 1:
            #We use 5 for init.
            diff_secnods = int(diff_secnods/5)
        else:  # simulate mode
            diff_secnods = int(diff_secnods/10)  # using 10 images in the camera
_captures
    else:
        # cannot judge the node type, so just use the original
        diff_secnods = diff_secnods
    print('avg. seconds : {}'.format(diff_secnods))
    new_intvl = interval
    # adjust first
    if diff_secnods <= 15:
        new_intvl += 10
    else:
        new_intvl -= 10
    # and check the value latter
    if new_intvl >= 120:
        # time interval reach max value
        new_intvl = 120
    elif new_intvl < 10:
        # reach the min value
                                                              86,12         37%
```

Fig. 10. (Color online) Gradually increasing/decreasing the returning time.



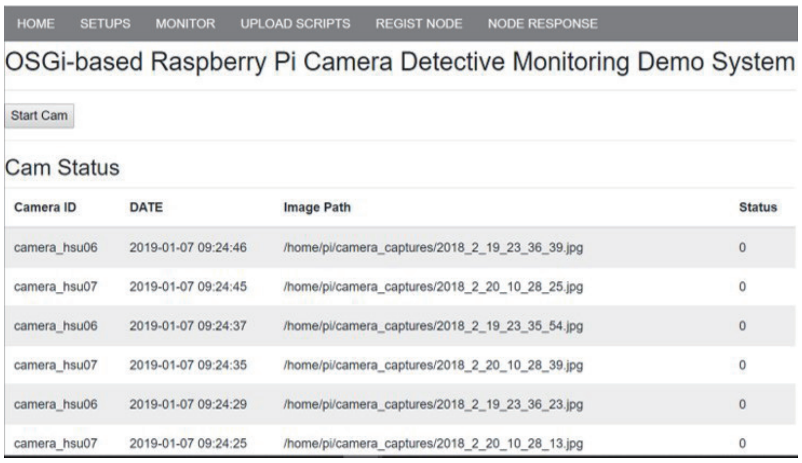
```
    if len(msg_sents):
        for item in msg_sents:
            # now sent each message based on different time period
            current = datetime.datetime.now()  # strftime("%Y-%m-%d %H:%M:%S")

            if item[2] == None:
                # it's mean no response from node, need to check the time interv
al

                time_diff = current - item[1] # means no response from the begin
ning

                if time_diff.seconds >= 300:
                    write_db_and_sent_msg(item, current)
                    time.sleep(5)
            else:
                # ed_time = datetime.datetime.strptime(item[1],"%Y-%m-%d %H:%M:%
S") # datetime format in DateTimeField()
                time_diff = current - item[2]
                #becarefully, we run in batch mode, need to accu. the total time
                time_intervl = item[3]
                print('realtime diff : {}, new time intvl : {}'.format(str(time_
diff.seconds),str(time_intervl)))
                if time_diff.seconds >= time_intervl: #larger than the time inte
rval, than make a new record and sent the message
                                                              242,1         98%
```

Fig. 11. (Color online) Wake up process for delaying 300 s.

After the initial experiment, the average time for a single device to calculate the image was about 50 s. We selected this value as the threshold. When the device synchronization response time was less than 50 s, the return time gradually increased to 50 s. When the response time was longer than 50 s, the request time was reduced by 10 s to send the synchronization message. The corresponding processes for threshold setting and increasing/decreasing the return time and the waking up process are shown in Figs. 9, 10, and 11, respectively.

Experimental results of data analysis, computations, and response time are shown in Figs. 12, 13, and 14, respectively. Images were transmitted within the system to test the delay time and the response of each device. Using the test data, the monitoring system reorganized the task assignments and got the balance of delay time among the devices. The test platform for task assignments was the MQTT server.

From the above experimental results, the system event processing performance is found to affect the MQTT server by making the server send the task time again in the dynamic average calculation time. In addition, we observe that when someone is detected in the image and the event image is returned, the response time of the performance transmission increases. Through the established timeout startup mechanism in this experiment, the device sends the next MQTT event when it responds in time, wakes up the device to start the service, and reduces the transmission result due to the change. The experimental conditions are set to perform the calculation analysis on five sets of devices and perform conventional calculation analysis, and the experimental results are shown in Table 2.

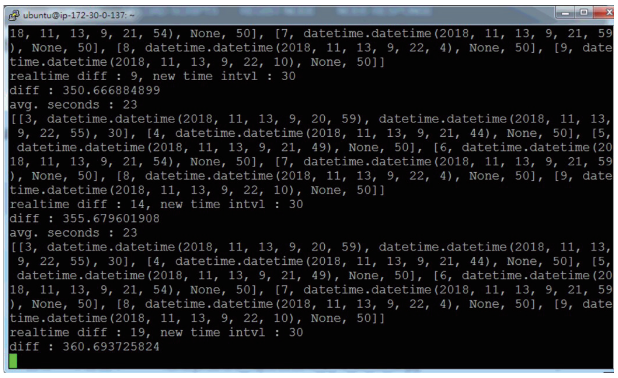


OSGi-based Raspberry Pi Camera Detective Monitoring Demo System

Cam Status

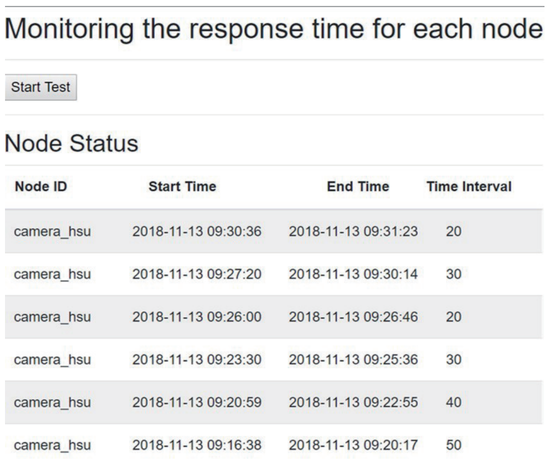
Camera ID	DATE	Image Path	Status
camera_hsu06	2019-01-07 09:24:46	/home/pi/camera_captures/2018_2_19_23_36_39.jpg	0
camera_hsu07	2019-01-07 09:24:45	/home/pi/camera_captures/2018_2_20_10_28_25.jpg	0
camera_hsu06	2019-01-07 09:24:37	/home/pi/camera_captures/2018_2_19_23_35_54.jpg	0
camera_hsu07	2019-01-07 09:24:35	/home/pi/camera_captures/2018_2_20_10_28_39.jpg	0
camera_hsu06	2019-01-07 09:24:29	/home/pi/camera_captures/2018_2_19_23_36_23.jpg	0
camera_hsu07	2019-01-07 09:24:25	/home/pi/camera_captures/2018_2_20_10_28_13.jpg	0

Fig. 12. Data analysis results of IoT devices.



```
18, 11, 13, 9, 21, 54), None, 50], [7, datetime.datetime(2018, 11, 13, 9, 21, 59
), None, 50], [8, datetime.datetime(2018, 11, 13, 9, 22, 4), None, 50], [9, date
time.datetime(2018, 11, 13, 9, 22, 10), None, 50]]
realtime diff : 9, new time intvl : 30
diff : 350.666884899
avg. seconds : 23
[[3, datetime.datetime(2018, 11, 13, 9, 20, 59), datetime.datetime(2018, 11, 13,
 9, 22, 55), 30], [4, datetime.datetime(2018, 11, 13, 9, 21, 44), None, 50], [5,
 datetime.datetime(2018, 11, 13, 9, 21, 49), None, 50], [6, datetime.datetime(20
18, 11, 13, 9, 21, 54), None, 50], [7, datetime.datetime(2018, 11, 13, 9, 21, 59
), None, 50], [8, datetime.datetime(2018, 11, 13, 9, 22, 4), None, 50], [9, date
time.datetime(2018, 11, 13, 9, 22, 10), None, 50]]
realtime diff : 14, new time intvl : 30
diff : 355.679601908
avg. seconds : 23
[[3, datetime.datetime(2018, 11, 13, 9, 20, 59), datetime.datetime(2018, 11, 13,
 9, 22, 55), 30], [4, datetime.datetime(2018, 11, 13, 9, 21, 44), None, 50], [5,
 datetime.datetime(2018, 11, 13, 9, 21, 49), None, 50], [6, datetime.datetime(20
18, 11, 13, 9, 21, 54), None, 50], [7, datetime.datetime(2018, 11, 13, 9, 21, 59
), None, 50], [8, datetime.datetime(2018, 11, 13, 9, 22, 4), None, 50], [9, date
time.datetime(2018, 11, 13, 9, 22, 10), None, 50]]
realtime diff : 19, new time intvl : 30
diff : 360.693725824
```

Fig. 13. (Color online) Results of comparison of real computations.



Monitoring the response time for each node

Node Status

Node ID	Start Time	End Time	Time Interval
camera_hsu	2018-11-13 09:30:36	2018-11-13 09:31:23	20
camera_hsu	2018-11-13 09:27:20	2018-11-13 09:30:14	30
camera_hsu	2018-11-13 09:26:00	2018-11-13 09:26:46	20
camera_hsu	2018-11-13 09:23:30	2018-11-13 09:25:36	30
camera_hsu	2018-11-13 09:20:59	2018-11-13 09:22:55	40
camera_hsu	2018-11-13 09:16:38	2018-11-13 09:20:17	50

Fig. 14. Response time of task assignment.

Table 2
Processing time for conventional calculation analysis on five sets of devices.

Fog Device Timer	N	Z	Z+(N+1)xA s	N^+	A^+= Next Check Timer	Y=Average Responses	Real Response
1	Null	50	50	Null	50	41	41
2	1	50	50+(Null+1)*10	1	60	(41+45)/2=43	45
3	2	60	60+(1+1)*10	2	70	(41+43+53)/3=46	53
4	1	70	--	1	60	(41+43+53+46)/4=45.75	46
5	2	60	60+(1+1)*10	2	70	(41+43+53+46+38)/5=44.2	38
6	3	70	70+(2+1)*10	3	90	(41+43+53+46+38+42)/6=43.8	42
7	4	90	90+(3+1)*10	4	110	(41+43+53+46+38+42+55)/7=45.42	55

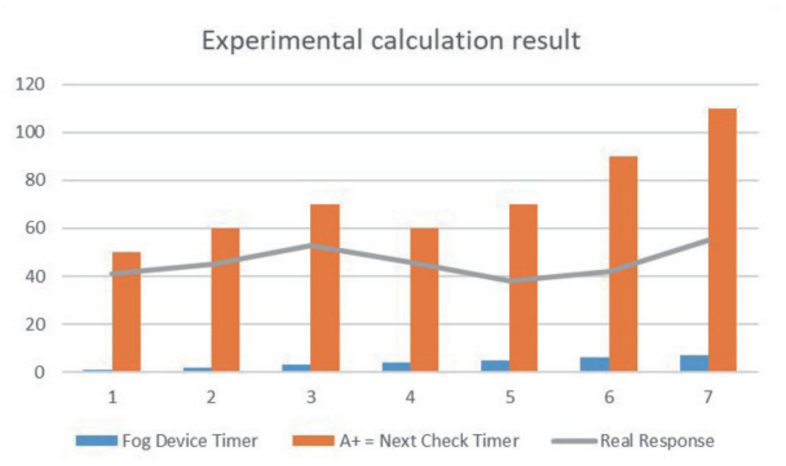


Fig. 16. (Color online) Reaction time for dynamic data transmission.

From the experimental data, it can be found that under the limited control state, the backhaul timer can be established through this study, which can gradually reduce the number of dynamic hardware requirements. Adjusting the event processing backhaul performance scattered in different fog devices can gradually postpone the time required to return the data. At the same time, the data processing response time does not change owing to the delay or increase in the frequency of detection responses. Figure 16 shows the reaction time for dynamic data transmission. It is observed that the variance of real response time is limited at an interval of 20 s.

5. Conclusions and Future Work

In this study, there are two main contributions that are proposed to effectively improve the performance of the entire system by the integration of fog operations and cloud, using IoT to synchronize multiple devices.

1. The dynamic adjustment algorithm for response time is applied to the event processing system for IoT devices.
2. For a multidevice synchronization operation, a mechanism with an improved dynamic operation transmission mode is proposed.

In this experiment, the proposed calculation method is used to increase the frequency of detection events and effectively correct the response time under different conditions and prevent the loss of a packet due to a long reaction time, which makes it impossible to detect the event processing situation. At the same time, with the combined operational advantages of the fog devices, the dispatched analysis and calculation events are delivered to the fog devices. In this study, we combine device computing performance with network performance. Research on the performance improvement calculation models for high-performance network environments and poor network environments will continue in the future. It includes performing calculations at IoT endpoints, establishing a network performance audit mechanism, improving the M2M communication model in the IoT computing service environment, and continuously integrating the results of cloud research.

References

1 D.-H. Park, H.-C. Bang, C. S. Pyo, and S.-J. Kang: 2014 IEEE World Forum on Internet of Things (WF-IoT) (IEEE 2014). https://doi.org/10.1109/WF-IoT.2014.6803125
2 S. Wang, Y. Hou, F. Gao, and X. Ji: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT) (IEEE 2016). https://doi.org/10.1109/WF-IoT.2016.7845396
3 M. Ishino, Y. Koizumi, and T. Hasegawa: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT) (IEEE 2015) 553–558.
4 P. Yadav and S. Vishwakarma: 2018 3rd Int. Conf. Internet of Things: Smart Innovation and Usages (IoT-SIU) (IEEE 2016). https://doi.org/10.1109/iot-siu.2018.8519920
5 M. Sreeram and M. Sreeja: 2017 Int. Conf. I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) (IEEE 2017) 486–491. https://doi.org/10.1109/I-SMAC.2017.8058398
6 I.-R. Chen, J. Guo, and F. Bao: IEEE Trans. Serv. Comput. **9** (2016) 3. https://doi.org/10.1109/TSC.2014.2365797
7 A. S. Petrenko, S. A. Petrenko, K. A. Makoveichuk, and P. V. Chetyrbok: 2018 IEEE Conf. Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (IEEE 2018). https://doi.org/10.1109/EIConRus.2018.8317246
8 D. Guinard, C. Floerkemeier, and S. Sarma: WoT 11 2nd Int. Workshop on Web of Things (ACM 2011). https://doi.org/10.1145/1993966.1993979

About the Authors

Tse-Chuan Hsu received his M.S. degree in computer engineering from Tunghai University, Taichung, Taiwan. He is pursuing his Ph.D. degree at the BathSpa University Centre for Creative Computing and he is also a lecturer in the Department of Information Management, Hsuan Chuang University, Hsinchu City, Taiwan. His research interests include Internet of Things, software engineering, and cloud computing data analysis systems.

Yao-Hong Tsai received his M.S. and Ph.D. degrees in information management from the National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, R.O.C., in 1994 and 1998, respectively. He was a researcher at the Advanced Technology Center, Information and Communications Research Laboratories, Industrial Technology Research Institute (ITRI), Hsinchu. He is currently an associate professor at the Department of Information Management, Hsuan Chuang University, Hsinchu. His current research interests include image processing, pattern recognition, personal communication systems, and cloud computing.

Dong-Meau Chang received his B.S. degree in chemical engineering from Tatung University, Taiwan, in 1988, and his M.S. and Ph.D. degrees from National Taiwan University and Tatung University, Taiwan, in 1990 and 1996, respectively. Since 2018, he has been a professor at Longyan University. His research interests are in IoT and Big Data Analysis.

Fu-Yao Liu received her B.S. degree in healthcare administration from Tajen University, Taiwan, in 2006 and her M.S. and Ph.D. degrees from the Aletheia University and National Ping-Tung University of Science and Technology, Taiwan, in 2009 and 2016, respectively. Since 2018, she has been a professor at Sanming University. Her research interests are in Smart Tourism and IoT.

Chih-Hung Chang received his Ph.D. degree in computer science from Feng Chia University in 2004. Currently, he is an associate professor at Providence University. He is a member of the IEEE Computer Society. His research interests include software engineering, cloud service, and big data. He has published more than 140 papers in journals, book chapters, and conference proceedings. He served as local arrangement chair for COMPSAC 2015 and program chair for TANET 2017. (ch.chang@gm.pu.edu.tw)