EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

IoT-F2CDM-LB: IoT Based Fog-to-Cloud and Data-in-Motion Architectures with Load Balancing

Istabraq M. Al-Joboury[1] and Emad H. Al-Hemiary[2]

[1,2]Al-Nahrain University, College of Information Engineering, Department of Networks Engineering, Baghdad, Iraq
{[1]estabriq_94, [2]emad@coie-nahrain.edu.iq}

Abstract

The work in this paper tries to enhance the performance of IoT by modifying the Cloud based architecture in terms of storage, processing, and Load Balancing (LB). The assumption is as follows: In a single Fog server, high traffic coming from Things may cause packet loss which in turn affect the overall IoT performance. To overcome such a situation, LB on Fog layer is proposed and implemented practically using virtualization technology. The proposed IoT based Fog-To-Cloud and Data-in-Motion with LB (IoT-F2CDM-LB) architecture consists of two load balancers; HAProxy and Server Load Balancing (SLB), are used to distribute messages from Things to four virtualized Fog servers according to Least Connection technique. The implementation is carried out using VirtualBox and GNS3. Message Queue Telemetry Transport (MQTT) protocol is used to transfer messages between layers. Both load balancers result in packet loss reduction by 50%, lower response time and higher throughput.

Keywords: Internet of Things; Fog Computing; Cloud Computing; Load Balancing; Virtualization Technology.

Received on 07 November 2017, accepted on 23 December 2017, published on 23 January 2018

doi: 10.4108/eai.6-4-2018.155332

1. Introduction

The Internet of Things (IoT) has emerged with features of ubiquitous connectivity (anything, anywhere, and anytime). Things include actuators, sensors, or objects (smart watch, smart car, smart pen, smart chair, smart bag, etc.). With IoT, it is possible to turn anything to smart called smart X. These smart Things communicate with each other to achieve a complex and difficult goals without human intervention. IoT becomes more and more an interesting concept almost in all domains of life because of its features and capabilities. Things are low power, low bandwidth, low cost and low memory, and identified by Internet Protocol (IP) to be linked to the world through the Internet [1-2]. Cloud computing (CC) is a suitable way for storing and managing web services by applying "pay-as-you-go" model. CC isolates resources and services from customers in order to facilitate the operation of applications. However, it has an impact on real time applications because of possible high delay. CC has three different types, namely: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Services (PaaS) [3-4]. Fog computing (FC) has the same features of CC (virtualization, network, processing, and computing) and made by Cisco. These features are provided near to Things to solve the problem of high delay and is specialized for sensitive applications like healthcare [5]. Cisco said that there will be around 25 billion of Things, objects, and actuators by 2020. These Things generate an enormous number of messages. Governments and societies benefit from these messages to improve the quality-of-life of end users [6]. Messages from Things can be analyzed and processed using Data in Motion (DM) technique that is introduced by IBM to provide analyzing messages without to need storing it [7-8]. Virtualization is a technology used to reduce power, cost, space and complexity by making multiple virtual servers on a single physical server called Virtual Machines (VMs). It provides a middleware layer called Hypervisor to separate the hardware and software layers [9]. Due to high number of messages from Things, servers may crash when processing these

requests. Thence, Load Balancing (LB) can solve this problem by distributing the messages across multiple servers [10]. The main contribution of this paper is to propose IoT based Fog-to-Cloud and Data-in-Motion with LB (IoT-F2CDM-LB) architectures using virtualization technology with Message Queue Telemetry Transport (MQTT) protocol to reduce the packet loss and achieve the highest throughput. The rest of this paper is organized as follows: section 2 explains the MQTT protocol. Section 3 introduces LB techniques and load balancers. Section 4 lists literature review. Section 5 describes IoT-F2CDM-LB architectures. Section 6 and 7 lists the hardware and software, and discusses the design and implementation of proposed architectures respectively. Section 8 and 9 show the results with discussion. Finally, section 10 concludes this paper.

2. MQTT Protocol

MQTT protocol is an open source, application layer based on Transmission Control Protocol (TCP). It operates as publish/subscribe model with asynchronous connections. It has a low overhead (2 bytes header), comparable to client/server model. It is simple to implement and is recommended for smart applications like smart hospital, smart home, smart school, etc. MQTT improves performance of high delay and low bandwidth. The Broker part of the MQTT is needed to enable connections between clients (publishers and subscribers) [11]. Publishers send messages within a specific topic, then the broker distributes these messages to the subscribed clients. The topics of MQTT consist of levels sectioned by slash in order to clarify a specific topic among of all topics in the same network. Clients use wildcards such as: + (Plus symbol) represents all values in single level and only one topic. # (Hash symbol): represents all values in multiple levels [12].

The MQTT protocol has a retained message which means that broker saves the last value on each topic and sends to new subscribed clients in order to save messages from failures. MQTT has three levels of Quality of Service (QoS) to guarantee the message delivery [13]: QoS level 0, message is received at most once, without an acknowledgment; QoS level 1, message is received at least once, with an acknowledgment. Finally, QoS level 2 in which message is received at exactly once, and this level requires four-way handshake. Last Will and Testament (LWT) message is sent by broker to advertise all subscribers that there is something wrong on connections [14].

3. LB

LB is the process of enhancing performance by distributing traffic among a number of server pools. There are a few load balancers that support MQTT protocol. The researchers used High Availability Proxy (HAProxy) open sourced [15], Nginx Plus [16], and Elastic Beam [17] where license requirement is needed. This section explains two main load balancers used in this paper: HAProxy and Server Load Balancing (SLB) router.

The HAProxy is an open source software used to distribute load among multiple servers based on TCP. HAProxy consists of two lists: Back-end list that contains servers that receive messages from front-end list. This list can be defined by: IP address, port, and LB technique used. The other is the Front-end list that represents messages from clients. This list can be defined by: IP address and port of clients, and Access Control List (ACL). ACL is used to provide some rules to permit or deny messages arrive from clients to servers [18]. HAProxy uses health check to monitor the availability of back-end servers. Health check works by establishing TCP connection between the HAProxy and the back-end servers. If one of servers fails to process messages from the front-end, then HAProxy removes the server from the back-end list [19]. HAProxy provides a redundancy by adding two HAProxys and act as an active or passive model using a Keepalived to prevent the load balancer gets overwhelmed with huge number of messages because of single point of failure. Keepalives is a software used for routing messages, it works by creating a shared virtual IP address between two HAProxys. For example, clients use 10.0.0.3 as the destination IP address as shown in Figure 1 [20].

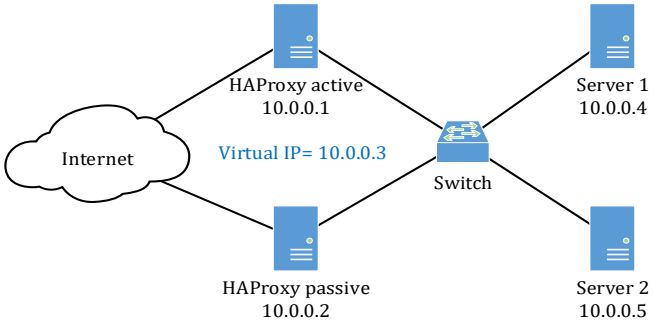

Figure 1. HAProxy with Keepalived (drawn from text [18])

The SLB router is an IOS image of Cisco router. Inside SLB router, a virtual server is defined to represent a list of real servers called server farm. SLB router redirects messages from clients to this virtual server, and then the virtual server redirects messages to one of the servers in the server farm. According to a specified LB technique [21], SLB can be in one of the two modes: Directed mode indicates that virtual server can be configured with any class of IP regardless of the class of the servers in the server farm. Therefore, this type needs to perform NAT in order to translate the IP address of virtual server to one of destination servers in server farm. For example, if the IP address of virtual server is 192.1.1.1, then the clients sends the messages to destination 192.1.1.1, and the SLB router receives these messages and redirects to one of IP address of servers using NAT protocol as shown in Figure 2 [22]. The second mode is dispatched. It indicates that IP address of virtual server is configured with servers in server farm. Each

server has its own real IP address plus virtual address of virtual server. This type needs the servers to be connected with the same Local Area Network (LAN) of SLB router. For this reason, it could not be capable in network with multiple routers as shown in Figure 3 [23]. SLB router provides a redundancy using (Hot Standby Redundancy Protocol) HSRP protocol between two SLB routers by creating a shared virtual IP address as shown in Figure 4 [24].

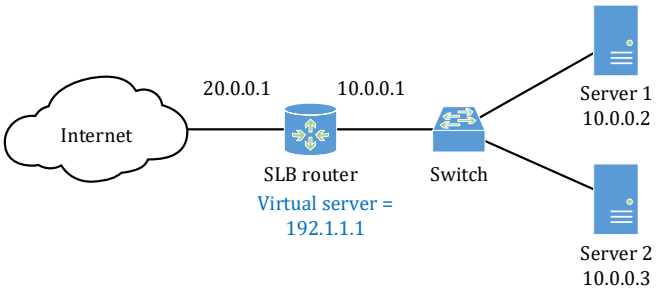

Figure 2. SLB router in directed mode (redrawn from [22])

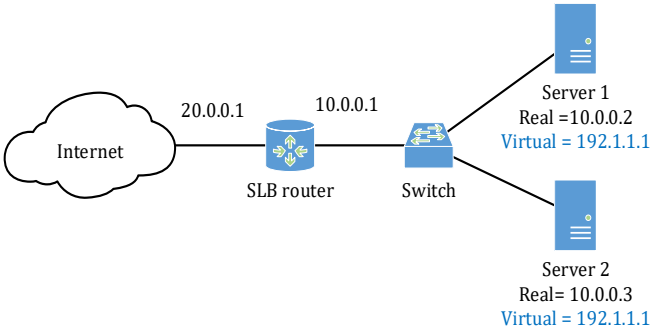

Figure 3. SLB router in dispatched mode (redrawn from [23])

These load balancers distribute messages from multiple clients to several servers simultaneously according to a specific technique to reduce the traffic on single server. LB techniques can be classified into several types [25]; the most common techniques are as follows:

1) Round Robin (RR) This technique distributes the traffic among the servers sequentially. For example, two servers receive messages from clients in sequence as shown in Figure 5.
2) Weighted Round Robin (WRR) is the same as RR; however, some servers have capabilities more than others. Therefore, weights can be added to the servers. For instance, if there are two servers: the first server receives 5 messages (because it is weighted to 5), the second server receives 1 message (because it is weighted to 1), and so on as shown in Figure 6.
3) Least Connection (LeastConn) technique chooses the server with the lowest number of active connections. For example, if there are three servers: Server 1 is processing 3 connections, server 2 is processing 15 connections, and server 3 is not processing any connections. Some servers have more overloads than others because clients stay connecting to servers much longer than other servers. So, server 3 receives three messages. Both servers 1 and 3 have now the same number of active connections. Then, load balancer performs RR on server 1 and 3 until number of active connections in both servers reach 15. Thence, load balancer performs RR on three servers 1, 2, and 3 and so on.

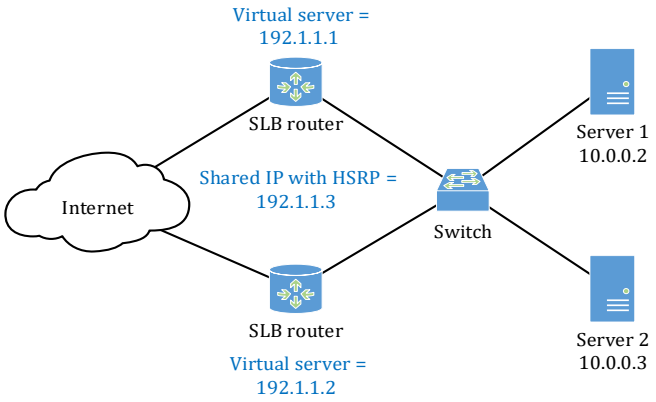

Figure 4. Two SLB routers with HSRP protocol (redrawn from [24])

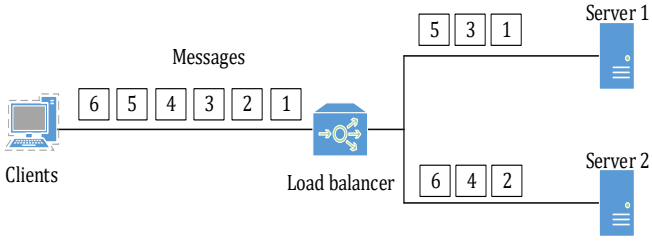

Figure 5. RR Technique (redrawn from [25])

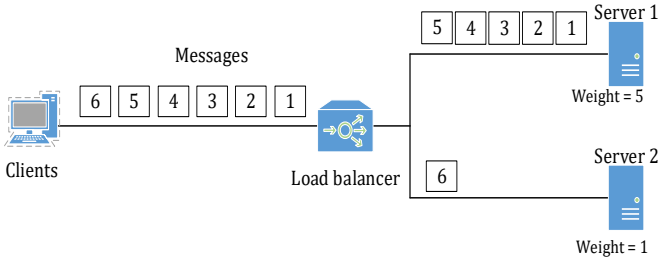

Figure 6. WRR Technique (redrawn from [25])

4. Literature survey

There are few works have been proposed the integration between Fog and Cloud or simply (F2C) as in [26], the authors introduce F2C architecture and its advantages with

main challenges and implement IoT based F2C and IoT based Cloud architectures using Tareador simulation tool to provide the performance comparison between them in terms of execution time and speedup. The results show IoT based F2C is better than IoT based Cloud because it is reduced the execution time. However, the authors have not considered IoT protocols in the architecture and not implemented the proposed architectures practically. But, the authors in [27] implement F2C computing on real application for chronic obstructive pulmonary disease (COPD) patients and the results show that the F2C improves the quality of life of patients. None of two previous papers have considered the delay on their proposed, while the authors in [28], propose distributed service allocation strategy for both resource offering and service requirements based on F2C computing in order to reduce delay of service allocation and decrease traffic load on Cloud. Then, the same authors in [29] minimize the delay of the services and provide the capacity requirement. In addition, the queuing theory are considered in F2C concept as in [30], the optimal workload allocation is proposed based F2C architecture to reduce power consumption and delay. The results show that F2C is better performance that Cloud. We notice that the previous researchers have not been tested and implemented F2C with IoT protocols. Some authors try to propose an opposite path from Cloud to Fog (C2F) as in [31], the authors propose C2F architecture for monitoring healthcare network and smart homes. The results show C2F provides the better service to Things. Finally, the authors in [32] propose IoT architecture based on Cloud to combine MQTT and Hypertext Transfer Protocol (HTTP) protocols and to distribute traffic among virtual servers using HAProxy. The performance evaluation of protocols is presented in terms of number of clients and Central Processing Unit (CPU) cores. The results show the MQTT protocol has better performance than HTTP. These authors have considered protocols, however the architecture is based on Cloud and not based on F2C.

This paper proposes a new IoT architectures based F2C with LB and virtualization using different load balancers, namely: SLB and HAProxy. Up to our knowledge, SLB has not been used previously by researchers in IoT applications. Also, previous researchers have been used HAProxy on Cloud layer not on Fog layer.

5. IoT-F2CDM architectures with LB

Messages are handled by a single Fog server in the traditional network, which might become loaded on heavy incoming messages. This condition will lead the Fog server to act slowly and results in packet loss. Consequently, packet loss may lead to wrong decision-making. To solve this situation, multiple server arrangements for LB can share processing power and prevent packet loss. For experimental purposes and due to limited number of available physical servers, virtualization technology is used to create the required number of servers for testing. The proposed IoT-F2CDM-LB architectures consider five layers: Things, gateway, Fog, Cloud, and monitoring with HAProxy and SLB. Messages from Things are distributed according to LeastConn technique across multiple servers to mitigate the packet loss. When a new publisher tries to connect, this technique translates message from publisher to server which has the least number of connections. This technique is used when servers have the same capabilities and is suitable for protocols of long sessions like MQTT protocol. Several servers are virtualized on single Fog server using type 2 to reduce the cost, power, and space. These virtual servers receive messages from load balancer and send to Cloud for permanent storage using DM technique.

The main hardware and software tools used in the proposal architectures as shown in Table 1 and Table 2 respectively. These software perform several tasks such as writing specific codes, simulate networks, emulates sensors, and monitoring messages.

6. Network design and implementation

This section discusses the design and implementation of IoT-F2CDM architectures over two sites: Site A (located at Al-Nahrain University, College of Information Engineering), and site B (located at the Ministry of Higher Education (MOHE), Department of Research and Development (RRD)). The layers of proposed architectures are as follows:

6.1. Things layer

In this paper, Things layer consists of one real heartbeat rate sensor and several virtual sensors from Tsung. Heartbeat rate sensor is programmed with C/C++ language. The real sensor is placed on patient's body to capture the heartbeat rate messages in real time. While, virtual sensors are emulated in Tsung tool to generate a high traffic using XML programming languages to match the real sensor. Tsung is used to reduce the cost, power, complexity and space of real sensors. Things publish messages using MQTT protocol with QoS level 0 and 1.

6.2. Gateway layer

Messages from Things layer are transferred to the upper layers through gateway layer located at site A. Mikrotik AP transmits messages from sensors to Fog layer. Cisco switch mediates the Things and Fog layer, and connects several hardware together. Cisco router is used to forward messages from Fog to Cloud layer through the Internet. Open Shortest Path (OSPF) protocol is configured on router to provide routing mechanisms.

6.3. Fog layer

This layer receives messages from Things layer to provide a temporal storage and distributes messages to four Fog servers using LB techniques, Then selects the important messages using a DM technique and sends to Cloud layer.

Table 1. Hardware used in IoT-F2CDM-LB architectures

Hardware	Type	Specifications
heartbeat pulse rate sensor [33]	Real	Runs over 5 volts
NodeMCU (ESP8266-12E) [34]	Real	Programmed with C/C++ language
HP Pavilion Laptop	Real	Tsung tool is installed on a PC (HP Pavilion) that works as a Linux server (3-core) with Ubuntu 64-bit server version 14.04.5 Long Term Support (LTS) OS, 4 GByte of Dynamic Random Access Memory (DRAM), 500 GByte of permeant storage. Tsung is required to be installed on independent hardware and not on a virtual machine (VM)
HP ProLiant 380 G7 (Fog server)	Virtual (IaaS)	A VM is setup using VirtualBox on Linux UBUNTU server 64-bit installed on HP ProLiant 380 G7 with 16-core server with 32 GB DRAM, 500 GB of permanent storage. The VM Fog server is equipped with single Core processor, 4 GB DRAM and 100 GB dynamic allocated of permanent storage
HP ProLiant 380 G8 (Cloud server)	Virtual (IaaS)	A VM is setup using VirtualBox on Linux UBUNTU server 64-bit installed an HP ProLiant 380 G8 with 16-cores and 32 GB DRAM and 500 GB of permanent storage. The VM Cloud server is equipped with single Core processor, 4 GB DRAM and 100 GB dynamic allocated permanent storage
Mikrotik Access Point (AP)	Real	IEEE802.11n
Tenda AP	Real	IEEE802.11n
Cisco switch catalyst 2924	Real	Cisco switch connects different devices in the same LAN
Cisco router 2621	Real	Cisco router connects different networks together

VMs are created to act as virtual Fog servers in order to create the required number of servers and apply LB. Four Fog VM servers and two HAProxys are virtualized with MQTT QoS level 0 and 1 using VirtualBox as shown in Figure 7. All VMs are configured to operate Apache2, PHP5, MySQL, PHP-Mosquitto, DM technique, and network configuration.

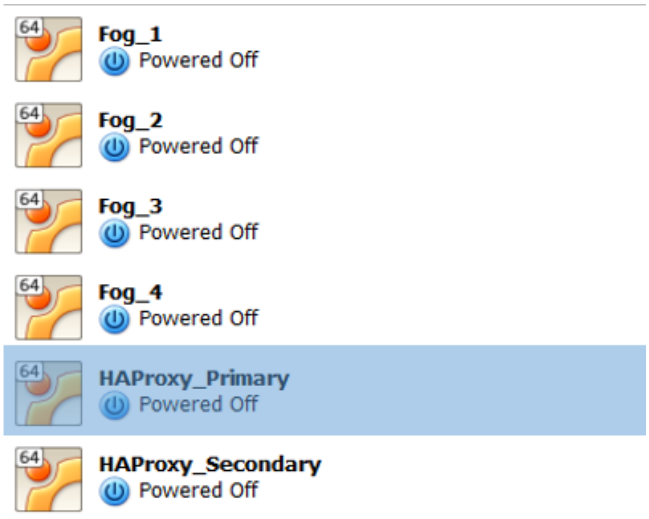

Figure 7. VMs in VirtualBox

A. HAProxy

It is used to distribute a huge number of messages based on TCP to four virtual Fog servers. These servers are setup to bridged setting in VirtualBox. The IP address of these servers are configured in HAProxy backend list. While, the IP address and ports of clients (Things) are configured in HAProxy frontedend list. Messages from clients are transferred to load balancer using MQTT with QoS of MQTT level 0 and 1. Then, these messages are separated to four virtual servers according to LeastConn fashion. Each virtual Fog server has a DM procedure to select the important messages, and then send to Cloud layer. Due to a single point of failure, the Fog layer provides availability in terms of servers. Two load balancers as an active/passive model is configured to the architecture using the Keepalived mechanism. Keepalive is used for monitoring the connections between the frontend and load balancer. Re-route the traffic to secondary load balancer if an interrupt is detected as shown in Figure 8. MQTT protocol is used for communication in this method. The previous research has been implemented HAProxy on Cloud layer, while this proposed architecture based F2C implements HAProxy on Fog layer and provides redundancy.

B. SLB router

SLB router is used in this proposed architecture as shown in Figure 9 because it is easy to employ and easy to maintain

Table 2. Supported software used in IoT-F2CDM-LB architectures

Software	Descriptions
Graphical Network Simulator-3 (GNS3) version 2.0.2 [35]	GNS3 is an emulation tool made by Cisco to visualize a virtual environment of networks. Also, it can connect GNS3 to the real world. GNS3 is based on daynamips that is a program to emulate an actual Cisco IOS routers series
VirtualBox [36] and VMware workstation [37]	VirtualBox and VMware are type 2 virtualization software used to create multiple VMs with different OSs in single physical PC or server as a Guest OS. VirtualBox has some differences than VMware. VirtualBox is more friendly, free, has low overhead, and consumes less power than VMware. However, VMware is a close source with licenses requirement, has the ability to drag and drop between the host OS and Guest OS
Tsung version 1.6.0 [38]	Tsung is an open source tool with General Public License version 2 (GPLv2) developed by Erlang and provides multi protocols related to proposal works; MQTT and HTTP and emulates millions of sensors in Poisson process. Up to our knowledge, there are two traffic generators that support IoT protocols JMeter [39] and Tsung
Arduino Integrated Development Environment (IDE) version 1.6.12 [40]	Arduino IDE is an open source tool written in Java used to write, debug, and upload code to MCUs platforms
Mosquitto broker version 1.4.10 [41]	Mosquitto broker is selected because it is widely used by researchers and written in C/C++ language and supports all level of QoS levels of MQTT. However, there are 26 types of MQTT brokers (for further information, the reader can look into [42])
MQTT_spy version 0.5.4 [43]	Mqtt_spy is a Java version 8 based tool used for monitoring, debugging, and troubleshooting on MQTT topics and payload. The main features of Mqtt_spy, includes: connect to several servers simultaneously, automatic reconnection when the connection is interrupt, support security such as TLS/SSL, provides graph, and can filter or summarize messages arrived from server
Wireshark version 2.4.0 [44]	Wireshark is a network analysis tool used to capture traffic between client and server and analyze different network protocols encapsulation
Putty version 0.70 [45]	Putty is a client side terminal emulator and based on Secure Shell (SSH). It allows to remotely access another host with authentication through Internet. Clients can use CLI when login to that host. Also, Putty allows multi-clients login
Processing Development Environments (PDE) version 3.2.1 [46]	PDE is a program used to process Java programming language and create applications with charts and sound

the servers (configuration is centralized). The IP addresses of servers are unknow to the outside network, thus enhances the security. Up to our knowledge, this type of load balancer is not used previously by researchers on IoT architecture. GNS3 is used to emulate the proposed virtualized Fog network as shown in Figure 10, where two SLB routers are installed and configured. Two SLB routers are connected to Cisco switch.

On the other terminal, two SLB routers are connected to real hardware of the IoT architecture (Gateway layer) through cloud. The cloud symbol inside GNS3 is not the intended CC but it represents the interface between the network inside GNS3 and the real world. The Cisco switch is connected to four Fog servers that are installed and setup to host settings in VirtualBox and is connected to Internet through cloud.

Two SLB routers are configured in directed mode in this proposed architecture. IP address of virtual Fog servers, port, and LB technique are defined in server farm of SLB routers. If one of servers is off or connection between SLB and server is interrupted, then the SLB waits for a settled time (set to 60 second - an adequate time for detecting failures) and the failed server is removed from the server farm. Due to SLB routers are in directed mode, NAT protocol is configured to map IP address of load balancer to one of virtual Fog servers according to LeastConn. In addition, OSPF routing protocol is configured in both SLB routers for forwarding messages to Cloud layer. The SLB routers act as active or passive using HSRP protocol to provide redundancy by creating a shared virtual IP address between them. The Things put this shared IP address in their destination field.

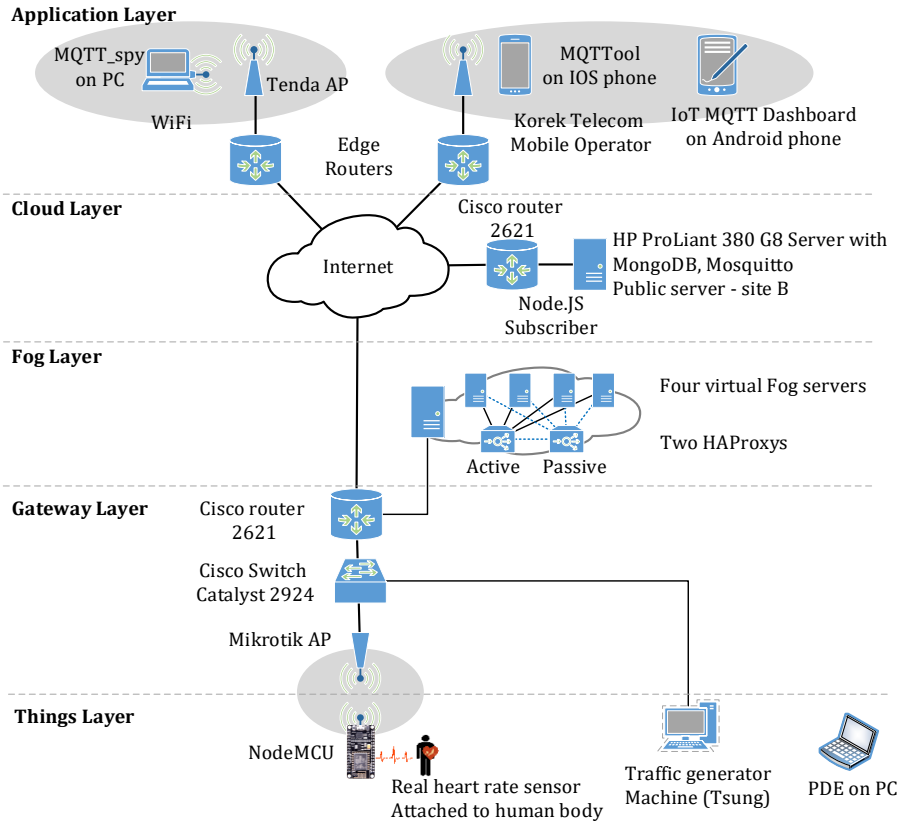

Figure 8. Proposed IoT-F2CDM-LB architecture with HAProxy load balancer

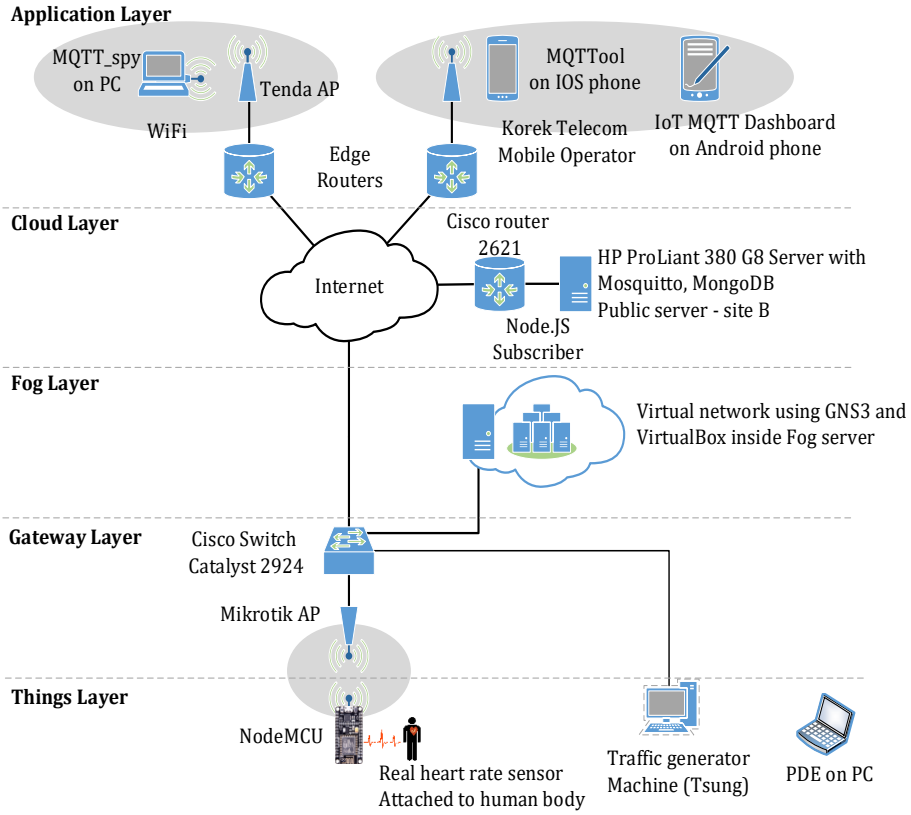

Figure 9. Proposed IoT-F2CDM-LB architecture with SLB load balancer

Messages from Things are transferred to load balancer (HAProxy or SLB) using MQTT with QoS of level 0 and 1. Then, these messages are sent to four virtual servers according to LeastConn technique. Each server selects important messages using DM technique and sends them to Cloud layer. DM technique is proposed to implement in Fog layer. The authors adopt the idea from DM in IOx Cisco router. This router consists of the traditional IOS image of Cisco router and Red-Hat OS for computing and storage. Cisco implemented DM with HTTP protocol. In this paper, proposed DM subscribe messages from Things using Python API with the help of PHP-Mosquitto broker. These messages are stored in MySQL database, and are filtered in real time to select important messages.

Three messages (maximum, average, and minimum) are selected every one hour from database using PHP5 and MySQLi within topic (/sensor/MAXIMUMvalue/), (within the topic /sensor/AVERAGEvalue/), and (within topic /sensor/MINIMUMvalue/) respectively. These three messages are sent to Cloud layer, while the rest are deleted as shown in Figure 11. This proposed DM technique are implemented with MQTT protocol with QoS level 0 and 1.

6.4. Cloud layer

This layer receives three messages (maximum, average, and minimum) using Node.JS with the help of Mosquito broker and located at site B. These messages are stored permanently in near real time in MongoDB. Physician or patient's family can monitor the history of the patient through Cloud layer.

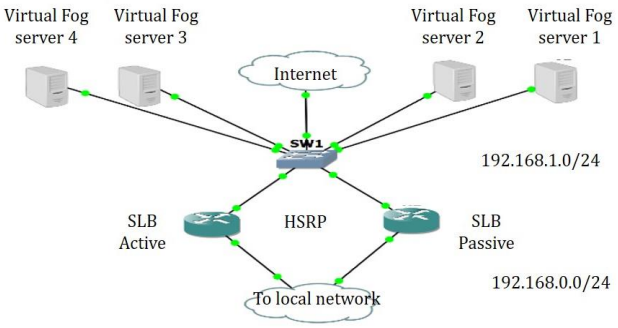

Figure 10. Virtual network using GNS3 and VirtualBox inside Fog server

6.5. Monitoring layer

Local and global monitoring tools are provided using smart phones, tablets, and PCs. PDE can monitor messages directly from sensors locally located at site A using TCP protocol. IoT MQTT Dashboard and MQTTool are installed and configured on Android and IPhone devices respectively to monitor messages using MQTT protocol form Things layer. MQTT_spy is installed and configured on VM using VMware to monitor messages from servers and debug brokers.

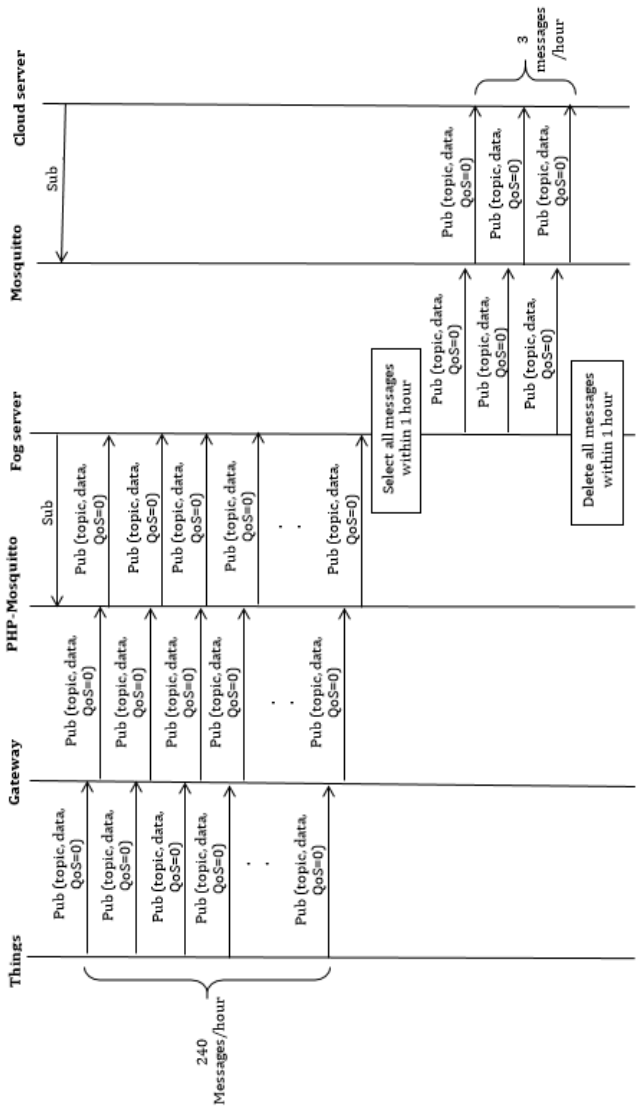

Figure 11. Sequence diagram IoT-F2CDM-LB architectures

7. Practical testing and results

This section presents the results of proposed IoT-F2CDM-LB architectures with explanations. These results are extracted from transferred messages from Things layer to Fog layer located at site A using MQTT protocol with QoS level 0 and 1.

7.1. Average throughput

Throughput is the number of messages per second that the destination server can handle. Two methods are used to find the throughput in this paper: Tsung captures messages when it runs. Throughput is computed using the following formula in Wireshark [47]:

$$Throughput = \frac{Size\ of\ successful\ messages\ in\ bits}{Transmission\ time\ in\ seconds} \quad (1)$$

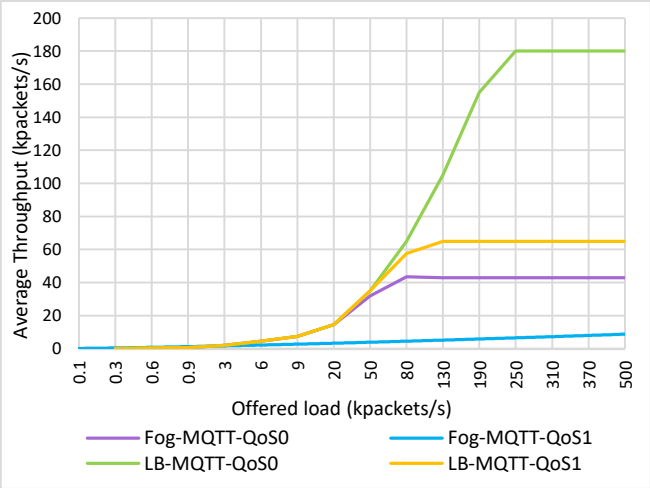

Figure 12. Average throughput of IoT-F2CDM-LB architectures

The average throughput is specified and calculated by averaging the collected values. Figure 12 show the average throughput in packets/s for both load balancers (HAProxy and SLB) with MQTT QoS level 0 and level 1. As the figure show, the use of LB increases Fog network throughput in both MQTT QoS level 0 and 1. Throughput is affected by the destination server capabilities to handle messages, and distance from destination server. Throughput decreases when average response time increases. The results reach the saturation level for all cases because of limited bandwidth.

7.2. Average packet loss

Packet loss is the number of packets fail to reach the destination server when they travel through network. Wireshark is used to calculate packet loss using the following equation [48]:

$$Packet\ Loss = Offered\ Load\ (Packets\ send\ by\ Things) - Packets\ Received\ by\ destination \quad (2)$$

Then, the average packet loss is calculated by averaging the collected values. Figure 13 show the average throughput in packets/s for both load balancers (HAProxy and SLB) with MQTT QoS level 0 and level 1. The results show that the use of LB reduces packet loss to its minimum value (half in QoS 0 and 1 times in QoS 1). This result comes from the fact that using LB will distribute the traffic over four virtual Fog servers, thus all messages arrive safe and sound.

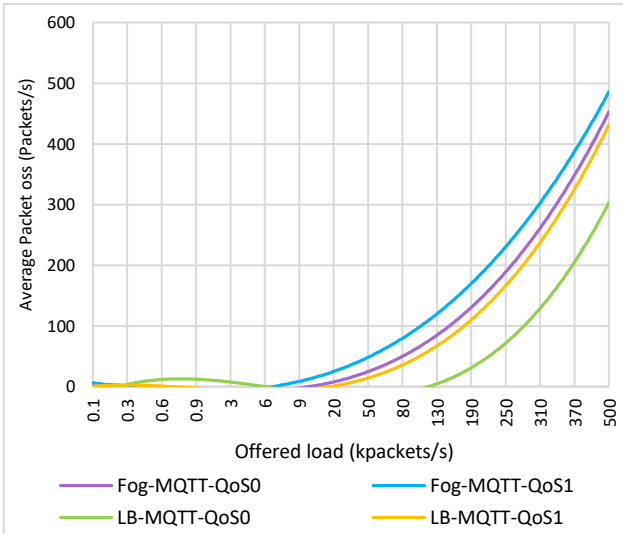

Figure 13. Average packet loss of IoT-F2CDM-LB architectures

7.3. Average delay

The delay is the time taken of message transferred from sender to destination server. It is captured using WireShark in timestamp. Figures 14 show the average delay in Fog layer with MQTT protocol with LB. This figure shows that messages is a little impacted in LB. and MQTT QoS level 1 is higher than MQTT QoS level 0 by factor 1 because the latter one has an ACK in application layer. Delay occurs because packet arrival rate to link exceeds output link capacity and packets queue in routers, messages must wait for turn.

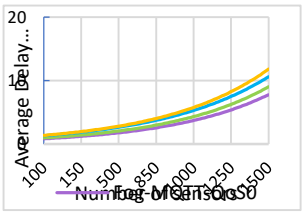

Figure 14. Average delay of IoT-F2CDM-LB architectures

7.4. Average Round-Trip Time (RTT)

RTT is the total time required for a message to travel from a sender in the Things layer to a destination server in the Fog or Cloud layers and returned back to the sender. Tsung collects messages transferred from sender to destination. Also, RTT is measured in Wireshark in terms

of Timestamps at application layer. RTT can be calculated using the following formula [49]:

$$RTT = (T\_4 - T\_1) - (T\_3 - T\_2) \quad (3)$$

In (3), T_1 is the timestamp the sender imitates a request message targeted to the destination server, T_2 is the timestamp the server receives the request message, T_3 is the timestamp the server respond to the request message after processing delay, and finally T_4 represents the timestamp the sender receives the response message. Since the RTT is affected by the link status between the sender and receiver, the average RTT is specified and calculated by averaging the collected values. Figure 15 show that MQTT QoS level 1 is higher than QoS level 0 by factor 2 because of QoS 0 does not have an ACK, and LB in both levels of QoS is affected by factor 1.

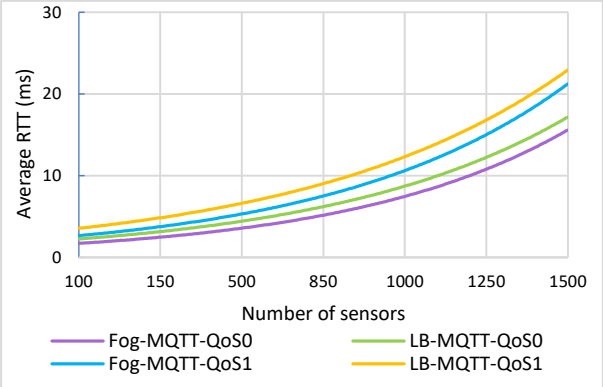

Figure 15. Average RTT of IoT-F2CDM-LB architectures

7.5. Average response time

The Response time is the elapsed time taken by a webserver to respond to a request initiated from a web client. This response time is measured using two tools: Tsung and Wireshark. Response time of HTTP is computed using the following commands in Wireshark: HTTP.request.method=="GET" || HTTP.response.code = 200. Then, the response time can be found in "time since request" in HTTP header. While the response time of MQTT QoS level 0 can be found by applying "MQTT" in the filter of Wireshark, then from the PUB message the value of response time is extracted. Finally, the same step for MQTT QoS level 1, expect that the value of the response time is seen in PUBACK. The average response time is specified and calculated by averaging the collected values. Figure 16 show the average response time of QoS level 1 is higher than QoS level 0 by factor 2 because QoS 1 have an ACK in transport and application layers.

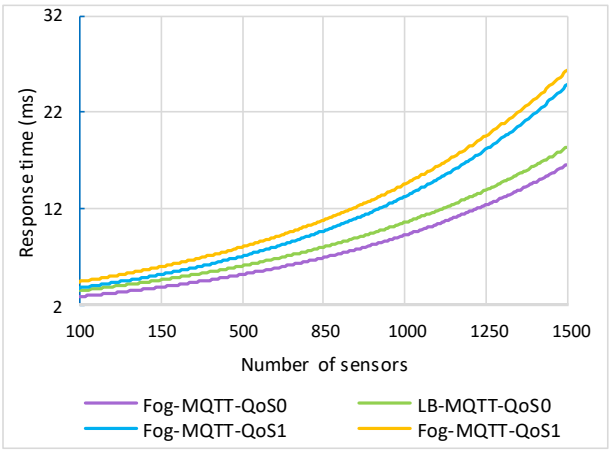

Figure 16. Average Response Time of IoT-F2CDM-LB architectures

7.6. HAProxy monitoring

Health check is configured on HAProxy to monitor the front and backend lists. It works by establishing a TCP connection between HAProxy and the backend servers in order to know if the servers listen to IP address and port or not. Figure 17 presents the traffic during four hours when four messages arrive every minute to HAProxy, then these messages are distributed across servers according to the LeastConn technique.

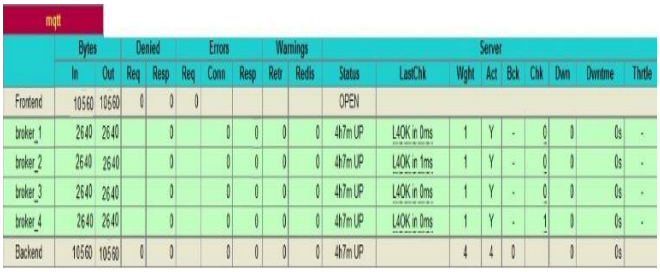

Figure 17. HAProxy monitoring

7.7. SLB monitoring

Traffic from Things can be monitored in SLB router as shown in Figure 18. The IP address of virtual load balancers (the first one is active, and the second one is passive) and number of transactions in active SLB are shown in the figure. Active SLB receives twenty-one messages for 5 minutes and distributes them to four virtual Fog servers according to LeastConn technique.

7.8. DM technique

This subsection presents the three messages (maximum, average, and minimum) arrived at Cloud layer from each four virtualized Fog servers as shown in Figure 19.

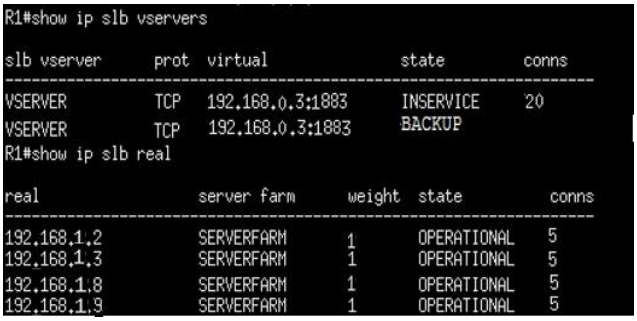

Figure 18. SLB monitoring

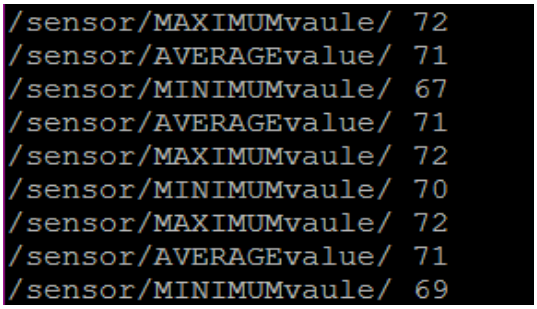

Figure 19. DM technique

8. Discussion

The IoT-F2CDM-LB architecture proposes the DM technique with QoS level 0. Also, the Things with QoS level 1 and level 2 can use DM technique without modification in the programming script. In this IoT-F2CDM-LB architecture, two different database structures: MySQL in Fog layer and MongoDB in Cloud layer. The reason for using MySQL in the Fog layer is to allow distributed design and handle complex transactions (multi-row transaction). On the other hand, MongoDB is more scalable to store messages and therefore installed in the Cloud layer. However, both databases can be used in both layers.

Two brokers are used, namely: Mosquitto and PHP-Mosquitto. The main difference between them is that the PHP-Mosquitto is compatible with PHP5 scripts and MySQL database. This chapter uses two different methods for LB, namely: HAProxy and SLB router. SLB router reduces the number of physical hardware used since everything is installed inside the router itself. The number of virtual Fog servers could be extended to more than four when the network senses increment in packet loss. API in Python script is used to make python and PHP5 with MySQLi scripts work together in the Fog layer and provide interoperability between MySQL and NoSQL databases.

9. Conclusions

The work in this paper proposes an architecture based on Fog and Cloud Computing to solve the performance degradation in the Cloud response time when large volume of Things traffic towards that Cloud. Introducing Fog layer nearby the Things layer contributes to network performance enhancement in terms of response time and packet loss. Two IoT architectures are proposed with LB (formed by two types of load balancer, namely: HAProxy and SLB router) and proposed DM technique using MQTT protocol with QoS level 0 and 1 in the Fog layer. The LB distribute the received messages from Things across a specified number of Fog servers according to LeastConn technique and results in reduction of packet loss, delay, RTT, and response time to half that without LB in the carried tests and achieves the highest throughput. Up to the author's knowledge, both LBs have not been evaluated on Fog layer by researchers previously. Finally, the results obtained have been conducted through practical implementation and can be considered as a possible solution to IoT based Fog/Cloud performance enhancement.

Bibliography.

Istabraq M. Al-Joboury has a M.Sc. (2017) and B.Sc. (2015) from the College of Engineering, department of Networks Engineering – Al-Nahrain University. She participated in many symposiums and workshops and published many papers in IoT related fields. Her fields of interests include IoT, computer networks and Internet technologies.

Emad H. Al-Hemiary is an associated professor and faculty member of the college of Information Engineering – Al-Nahrain University. He holds Ph.D. (2001), M.Sc. (1996), and B.Sc. (1993) from the College of Engineering – Al-Nahrain University in the field of electronics and Communications Engineering. He is currently the Head of the department of Networks Engineering and has many activities in research and cooperation with other institutes in networks and information technology. His fields of interests include modern networks and data communications, IoT, Network protocols and services and other related fields.

Acknowledgements.

The authors would like to acknowledge both the college of Information Engineering at Al-Nahrain University and the directorate of research and development at the ministry of higher education / Iraq for making the facilities available to the authors throughout the work period.

References

[1] Roy, S., Chowdhury, C., 2017. Integration of Internet of Everything (IoE) with Cloud. Internet of Things Beyond the Internet of Things 199–222.

[2] Al-Joboury, I.M., Al-Hemiary, E.H., 2017. Internet of Things (IoT): Readme. Qalaai Zanist Scientific Journal 2, 343–358.

[3] Negash, B., Rahmani, A.M., Liljeberg, P., Jantsch, A., 2017. Fog Computing Fundamentals in the Internet-of-Things. Fog Computing in the Internet of Things 3–13.

[4] Gilchrist, A., 2016. The Technical and Business Innovators of the Industrial Internet. Industry 4.0 33–64.

[5] Perera, C., Qin, Y., Estrella, J.C., Reiff-Marganiec, S., Vasilakos, A.V., 2017. Fog Computing for Sustainable Smart Cities. ACM Computing Surveys 50, 1–43.
[6] Srirama, S.N., 2017. Mobile web and cloud services enabling Internet of Things. CSI Transactions on ICT 5, 109–117.
[7] Papadokostaki, K., Mastorakis, G., Panagiotakis, S., Mavromoustakis, C.X., Dobre, C., Batalla, J.M., 2016. Handling Big Data in the Era of Internet of Things (IoT). Studies in Big Data Advances in Mobile Cloud Computing and Big Data in the 5G Era 3–22.
[8] Al-Joboury I.M., Al-Hemiary E.H., 2017. F2CDM: Internet of Things for Healthcare Network Based Fog-to-Cloud and Data-in-Motion Using MQTT Protocol. In: Sabir E., García Armada A., Ghogho M., Debbah M. (eds) Ubiquitous Networking. UNet 2017. Lecture Notes in Computer Science, vol 10542, p.p. 368–379Springer, Cham.
[9] Klement, M., 2017. Models of integration of virtualization in education: Virtualization technology and possibilities of its use in education. Computers & Education 105, 31–43.
[10] Kim, H.-S., Kim, H., Paek, J., Bahk, S., 2017. Load Balancing Under Heavy Traffic in RPL Routing Protocol for Low Power and Lossy Networks. IEEE Transactions on Mobile Computing 16, 964–979.
[11] Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., Katos, V., 2016. Which IoT Protocol? Comparing Standardized Approaches over a Common M2M Application. 2016 IEEE Global Communications Conference (GLOBECOM).
[12] Nastase, L., 2017. Security in the Internet of Things: A Survey on Application Layer Protocols. 2017 21st International Conference on Control Systems and Computer Science (CSCS).
[13] Sethi, P., Sarangi, S. R, 2017. Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering, Hindawi.
[14] Al-Joboury I.M., Al-Hemiary E.H., 2018. Performance Analysis of Internet of Things Protocols Based Fog/Cloud over High Traffic. Journal of Fundamental and Applied Sciences, 10 (6S), 176-181.
[15] HAProxy. powered by HAPROXY. URL https://www.haproxy.org/ (accessed 18.2.7).
[16] MQTT Load Balancing and Session Persistence with NGINX Plus, 2017. NGINX. URL https://www.nginx.com/blog/nginx-plus-iot-load-balancing-mqtt/ (accessed 18.2.7).
[17] Scalable and Secure MQTT Load Balancing with Elastic Beam and HiveMQ, 2016. HiveMQ. URL http://www.hivemq.com/blog/scalable-and-secure-mqtt-load-balancing-with-elastic-beam-and-hivemq/ (accessed 18.2.7).
[18] Prasetijo, A.B., Widianto, E.D., Hidayatullah, E.T., 2016. Performance comparisons of web server load balancing algorithms on HAProxy and Heartbeat. 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE).
[19] Yu, Y., Li, X., Qian, C., 2017. Sdlb: A Scalable and Dynamic Software Load Balancer for Fog and Mobile Edge Computing. Proceedings of the Workshop on Mobile Edge Communications - MECOMM 17.
[20] Fylaktopoulos, G., Skolarikis, M., Papadopoulos, I., Goumas, G., Sotiropoulos, A., Maglogiannis, I., 2018. A distributed modular platform for the development of cloud based applications. Future Generation Computer Systems 78, 127–141.
[21] Tiso, J., 2014. Designing Cisco network service architectures (ARCH): Foundation learning guide. Indianapolis, IN: Cisco Press.
[22] Al-Joboury I.M., Al-Hemiary E.H., 2018. Internet of Things Architecture Based Cloud for Healthcare. Iraqi Journal of Information & Communications Technology, 1 (1), 18-26.
[23] McQuerry, S., Jansen, D., Hucaby, D., 2009. Cisco LAN Switching Configuration Handbook: a concise reference for implementing the most frequently used features of the Cisco Catalyst family of switches. Cisco Press, Indianapolis, IN, USA.
[24] Deal, R.A., 2005. Cisco router firewall security. Cisco Press, Indianapolis.
[25] Yu, J., Lou, G., 2013. The Study of Server Load Scheduling Strategy. Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013).
[26] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, “Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems,” IEEE Wireless Communications, vol. 23, no. 5, pp. 120–128, 2016.
[27] Masip-Bruin, X., Marin-Tordera, E., Alonso, A., Garcia, J., 2016. Fog-to-cloud Computing (F2C): The key technology enabler for dependable e-health services deployment. 2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net).
[28] Souza, V.B., Masip-Bruin, X., Marin-Tordera, E., Ramirez, W., Sanchez, S., 2016. Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios. 2016 IEEE Global Communications Conference (GLOBECOM).
[29] Souza, V.B.C., Ramirez, W., Masip-Bruin, X., Marin-Tordera, E., Ren, G., Tashakor, G., 2016. Handling service allocation in combined Fog-cloud scenarios. 2016 IEEE International Conference on Communications (ICC).
[30] Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H., 2016. Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption. IEEE Internet of Things Journal 1–1.
[31] Nandyala, C.S., Kim, H.-K., 2016. From Cloud to Fog and IoT-Based Real-Time U-Healthcare Monitoring for Smart Homes and Hospitals. International Journal of Smart Home 10, 187–196.
[32] Hou, L., Zhao, S., Xiong, X., Zheng, K., Chatzimisios, P., Hossain, M. S., Xiang W., 2016. Internet of Things Cloud: Architecture and Implementation, IEEE Communications Magazine, 54, 12, 32–39.
[33] Heartbeats, Lickety-Split. World Famous Electronics llc. URL https://pulsesensor.com/ (accessed 10.1.17).
[34] NodeMcu -- An open-source firmware based on ESP8266 wifi-soc. URL http://nodemcu.com/index_en.html (accessed 10.1.17).
[35] GNS3 | The software that empowers network professionals. URL https://www.gns3.com/ (accessed 18.2.7).
[36] Oracle VM VirtualBox. URL https://www.virtualbox.org/ (accessed 18.2.7).
[37] Workstation - VMware Products, 2017. VMWare. URL http://www.vmware.com/products/workstation.html (accessed 18.2.7).
[38] Tsung, 2017. URL http://tsung.erlang-projects.org/ (accessed 18.2.7).
[39] JMeter. URL http://jmeter.apache.org/ (accessed 18.2.7).

[40] Ardunio IDE. URL https://www.arduino.cc/en/Main/Software (accessed 18.2.7).
[41] Mosquitto. URL https://mosquitto.org/ (accessed 18.2.7).
[42] MQTT brokers, "GitHub,". URL https://github.com/mqtt/mqtt.github.io/wiki/servers (accessed 18.2.7).
[43] Kamil Baczkowicz, " MQTT Toolbox – mqtt-spy,". URL http://www.hivemq.com/blog/mqtt-toolbox-mqtt-spy (accessed 18.2.7).
[44] Wireshark. URL https://www.wireshark.org/ (accessed 18.2.7).
[45] Putty. URL www.putty.org/ (accessed 18.2.7).
[46] Processing Development Environments. URL https://processing.org/reference/environment/ (accessed 18.2.7).
[47] Mamunur, M., Datta, P., 2017. Performance Analysis of Vehicular Ad Hoc Network (VANET) Considering Different Scenarios of a City. International Journal of Computer Applications 162, 1–7.
[48] Shriya, S., Suriyakrishnaan, K., Thenmozhi, s., Bhairavi, M., 2017. Improved Health Monitoring Using Location Aware Sensor Routing Protocol, International Journal of Advance Research, Ideas and Innovations in Technology, 3, 2.
[49] Zander, S., Armitage, G., 2013. Minimally-intrusive frequent round trip time measurements using Synthetic Packet-Pairs. 38th Annual IEEE Conference on Local Computer Networks.