Maximizing the Utilization of Fog Computing in Internet of Vehicle using SDN

Ahmed Jawad Kadhim and Seyed Amin Hosseini Seno

***Abstract*— The IoV sensors generate huge data that are processed by cloud servers, but this leads to high latency and bandwidth occupies. This was the main motivation behind fog computing. Software Defined Network (SDN) was developed to simplify the traditional networks architecture. Sometimes, when a fog server fails to execute some tasks, it sends them to cloud servers while other fog servers are idle. This leads to inefficient utilization of fog computing resources. This paper aims to optimize the resource utilization of fog computing with meeting the deadlines of tasks via assigning them to an optimal fog server using SDN. Moreover, local and global load balance techniques are proposed. The simulation results revealed that our proposed system decreases the transference of tasks to the cloud and as a result decreases the latency and bandwidth consumption.**

Keywords—Resource Utilization; Fog Computing; SDN; IoV;

I. Introduction

Internet of Vehicle (IoV) network refers to a subclass of the Internet of Things (IoT) that extended the paradigm of Vehicular Ad hoc Networks (VANET) [1]. It is a connection of numerous heterogeneous sensors installed on vehicles. Cloud computing is considered as a suitable framework to address the challenges of processing and storing big data of IoV network [2]. However, the long distance between the IoV network and cloud increases the latency. Moreover, due to the mobility of vehicles, sending tasks to the cloud will not be an optimal solution. This was the main drive behind discovering fog computing [1]. The goal of fog computing is to create a complement to the cloud computing by moving the resources to the edge of network with the aim of providing computing and storage services. It can get rid of the long distance between the cloud and sensors. Also, it can avoid the wasting of network bandwidth that occurs from the transferring of task requests to the remote cloud [3]. SDN was designed to tackle the challenges of traditional networks by dividing the architecture into two planes: data and control. Integrating the SDN concept with fog computing improves the performance of IoT. The SDN controller has completed knowledge and information about the tasks, available resources and IoT devices [4]. Generally, the launched tasks from the IoT sensors are sent to the closest fog server to be handled quickly. Sometimes, there are one or more idle fog servers while the others are overloaded. It indicates unfair distribution of the work and may cause catastrophes by exceeding the deadlines of hard real time tasks [5]. Therefore, this paper focuses on optimizing the resource utilization of fog computing to enhance the performance and decrease the need for the cloud servers. The main contributions of this paper are as follows:

- Proposing a resource management strategy to increase the utilization of fog computing and to prevent the transference of tasks to the cloud computing.
- Presenting two load balancing strategies to balance the load and minimize the response time to meet the deadline.
- Producing a new mathematical model to maximize the resource utilization of fog computing servers.

Manuscript received May 8, 2018; revised July 28, 2018 and September 15, 2018; accepted October 23, 2018. The associate editor coordinating the review of this paper and approving it for publication was jalel BEN OTHMAN *(Corresponding author: Seyed Amin Hosseini Seno.)*

The authors are with College of Engineering, Ferdowsi University of Mashhad, Iran (e-mail: ahmed.kadhim@mail.um.ac.ir; hosseini@um.ac.ir).

II. Related Works

The authors in [1, 6-10] used the fog computing and studied the collaboration between fog and cloud computing to optimize the performance of various types of networks. TABLE I presents a comparison between our proposed system and the related works according to different parameters.

TABLE I. Comparsion of Related Works (H = High, M = Medium, L = Low, x = The parameter has been taken into account)

Paper / Parameter	[1]	[6]	[7]	[8]	[9]	[10]	Our Proposed
Method	Load Balance	Resource Allocation + Load Balance	Task Placement + Task Scheduling	Load Balance	Load Balance	Resource Allocation	Local and Global Load Balance + Task Scheduling + Task Assignment + Monitoring
SDN based?	Yes	No	No	No	No	No	Yes
Deadline							X
Energy		X					
Delay	X	X	X	X	X	X	X
Bandwidth Consumption	M	H	H	H	H	M	L
Transmitted Tasks to Cloud	M	H	H	H	H	H	L
Fog Resource Utilization	M	L	L	L	L	L	H

III. System Model and Problem Formulation

The proposed architecture consists of five layers: IoV vehicles, clusters of fog devices, local load balancers, SDN controller and cloud servers as shown in Fig. 1. In the first layer, the IoV vehicles have sensors, GPS, and LTE interface. Each group of these vehicles found in a zone connects to the closest OpenFlow switch via LTE in a single hop. According to (1), the closest OpenFlow switch s_c to the IoV vehicle v can be determined by sending beacon packets. The OpenFlow switch responding to these packets with minimum delay is considered as a closest one. The IoV vehicles send information about the current location, speed and direction to the closest OpenFlow switch periodically. In the second layer, each cluster contains one OpenFlow switch and a number of fog servers. These fog devices can be installed in common places within cellular base stations. The OpenFlow switches have storage and processing capabilities. Moreover, each OpenFlow switch connects to the SDN controller and only one local load balancer. The OpenFlow switches send vehicle information to the SDN controller periodically. Also, each of fog servers sends information about its state including the number of waiting jobs in the queue and the average service time to the third layer (i.e. the local load balancer) periodically. Therefore, the load balancer can assign tasks to fog servers locally and balance the load according to this information. The local load balancers send their information to the SDN controller (fourth layer) which applies several modules in its application layer as shown in Fig 2. It monitors the fog computing resources and vehicles centrally and stores the vehicle information and fog server information in tables called *veh_table* and *fog_ table* respectively. It manages resources by balancing the load globally and assigning tasks to the optimal fog servers based on the number of waiting jobs, average service time of fog servers, load of network links, and vehicle mobility to minimize the idle time of fog servers and response time of tasks. The last layer represents cloud servers.

$$s_{c,v} \leftarrow \arg \min_{sc\in O} \ delay_{v,sc}\,, \ \forall\ v \in V,\ c \in \delta, \qquad (1)$$

To model the system, suppose V be the set of IoV vehicles and T the set of hard real time tasks that are launched according to a Poisson distribution from IoV vehicles. These tasks vary in term of data size and each one has some specific parameters $\{\Delta_i, \partial_i\}$. Other notations are found in TABLE II.

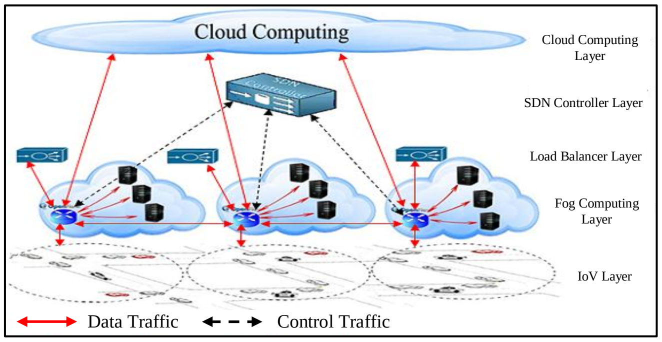

Fig. 1. The proposed architecture.

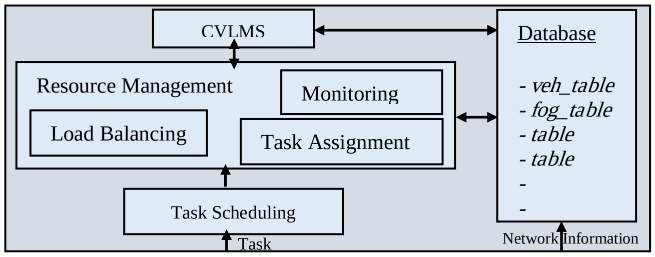

Fig. 2. The application layer of SDN controller.

TABLE II. NOTATIONS

Symbol	Definition
v, V	Index, set of IoV vehicles.
c, δ	Index, set of of clusters of fog servers.
s_c, O	The OpenFlow switch found in cluster c, the set of OpenFlow switches.
j, M	Index, number of fog servers.
i, N	Index, number of tasks.
$\lambda_{i,j}$	The task i that is executed by the fog server j.
p_{ij}	It is a variable represents the value of executing task i by j^{th} fog server. The highest value indicates that j^{th} fog server is more suitable to execute task i than other fog servers are.
Δ_i, ∂_i	Deadline of task i, required resources to execute task i.
τ, t_i	Time period, response time of task i.
C_j	Capacity of fog server j.

Maximizing the resource utilization of fog computing can reduce the required bandwidth between the fog and cloud computing and diminish the latency, thereby meeting the task deadline. The resource utilization of fog computing servers (U) represents the number of tasks handled by all fog servers in a certain period of time. The value of U can help the fog resource manager to obtain completed information about the resources and their utilization. Also, it allows the manager to make most of fog resources available to execute tasks within their deadlines and decrease the tasks that are sent to the cloud computing. It can be computed as follows [11]:

$$U = \Big(\sum_{j=1}^{M}\sum_{i=1}^{N} p_{i,j}\,\lambda_{i,j}\Big)/\tau\,, \qquad (2)$$

The objective of this paper is to maximize the resource utilization of fog computing and takes the deadlines of hard real time tasks into account. This objective is as follows:

Objective: Maximize U

S.T.

$$t_i \le \Delta_i\,,\ \forall\ i \in N, \qquad (3)$$

$$\sum_{j=1}^{M} \lambda_{ij} = 1,\ \forall i \in N, \qquad (4)$$

$$\sum_{i=1}^{N} \partial_i \le \sum_{j=1}^{M} C_j\,, \qquad (5)$$

$$C_j \ge \sum_{i=1}^{N} \partial_i \lambda_{ij}\,,\ \forall j \in M, \qquad (6)$$

$$\lambda_{i,j} \in \{0,1\},\ \ \forall\ i \in N,\ j \in M, \qquad (7)$$

$$t_i > 0,\ \forall\ i \in N, \qquad (8)$$

First constraint indicates that the response time of task i must be less than or equal to the deadline. According to the second constraint, each task must be executed by only one fog server. The third constraint illustrates that the required resources to execute N tasks must be less than or equal to the capacity of all fog servers M. The fourth constraint indicates that the required resources to execute number of tasks must be less than or equal to the capacity of fog server assigned to it. Fifth constraint explains that the value of $\lambda_{i,j}$ will be 1 if task i is executed in the fog server j; otherwise, it will be 0. Finally, the response time of each task must be more than 0.

The mathematical model is solved using CPLEX. Several examples are implemented using a computer (RAM 6GB and CPU core i5, 2.5GHz) to examine the efficiency of this model in optimizing the resource utilization. For example, let the number of fog servers (M) be 10 with a capacity (C_j) of 100, 150, 75, 130, 95, 70, 85, 105, 90 and 95. The number of tasks (N) is 50 with deadlines (Δ_i) ranging from 3 to 21 seconds and the required resources to execute the tasks (∂_i) are in the range of 3 to 12. The values of p_{ij} are random numbers between 0.1 and 1 and the time period (τ) is 60 seconds. The optimal solution (Maximum (U)) of this example is 0.6383. The objective function is maximum, therefore; the value of U explains that the resources of fog servers were exploited in an optimal manner to execute all tasks within the deadline.

IV. The Proposed Load Balance Technique

The local and global load balance techniques are proposed to decrease the idle time of fog servers. The local one applied in each local load balancer to balance the load in each cluster of fog servers while the global one is applied in the SDN controller to balance the load between different clusters of fog servers. According to these load balance techniques, the load balancers and SDN controller divide the capacity of each fog server logically into two parts: under loaded and overloaded, and assume a load threshold for each part as shown in Fig. 3. The under loaded and overloaded thresholds are 80% and 90% of CPU utilization, respectively. The rest resources (10%) are reserved for the ideal use. The value of overloaded part is more than 80% and less than 90% of the CPU utilization.

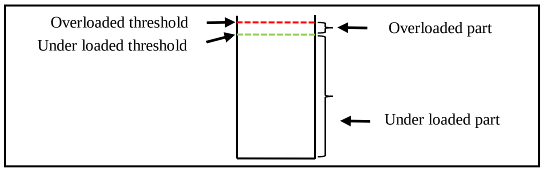

Fig. 3. The parts of fog server capacity and load thresholds.

The IoV vehicles send tasks to the closest OpenFlow switch that stores the task data and prepares some request contains information about that task (i.e. Δ_i, ∂_i). Then it sends this request to the local load balancer that schedules the incoming requests based on their priorities (i.e. the request with minimum deadline takes the highest priority).

The proposed load balance technique is based on the information received from fog servers (number of waiting jobs and average service time) and the above thresholds. It assigns tasks to optimal fog servers and balances the load as follows:

- If the load of all fog servers in a cluster is less than or equal to the under loaded threshold, then the load balancer selects an optimal fog server (the fog server with the minimum load) for each task according to (9).

$$\min_{j \in M} \text{load}_{(j)}, \tag{9}$$

$$\text{where} \qquad \text{load}_{(j)} = w_j * avg_j, \tag{10}$$

where $load_{(j)}$ is the load, w_j is the number of waiting jobs and avg_j is the average service time of fog server j.

- If the load of all fog servers in a cluster is higher than the under loaded threshold and less than the overloaded threshold, then the load balancer selects an optimal fog server for each task according to (9). Then it sends a notification to the SDN controller to inform it that all fog servers are in the overloaded range. At this point, the SDN controller checks the *fog_table*. If there is a fog server(s) in the under loaded range in another cluster, it sends a positive response to the notifying load balancer; otherwise, the response will be negative.

  After that, if the load of all fog servers in this cluster reached to the overloaded threshold and the SDN controller's response was positive, then the local load balancer sends a notification to the OpenFlow switch. This notification tells the OpenFlow switch that the fog servers are overloaded and the future task requests must be sent to the SDN controller directly. Therefore, the OpenFlow switch extracts the data of next incoming tasks and prepares requests to be sent to the SDN controller. Each request contains information about the deadline, required resources to execute the task and the vehicle that sent the task. When the requests reach the SDN controller, it schedules them based on the deadline. Then, it uses the *veh_table* to estimate the new location of the vehicle using the Collaborative Vehicle Location Management Service (CVLMS) algorithm [12] and then determines the new zone. After that, it uses the *veh_table* and *fog_table* to select an optimal fog server based on the priority. Here, the SDN controller determines the priority of fog servers based on the direction of vehicle's movement and new vehicle's location. Therefore, if the fog servers found in the new zone (the new zone is the area that the vehicle is expected to reach) are not overloaded, it chooses one using (9). Otherwise, if fog servers of other zones allocated on the direction of vehicle's movement are under loaded, it selects the nearest and optimal fog server using (11). Otherwise, it selects the optimal fog server from other zones (i.e. zones that are not allocated on the direction of vehicle's movement) using (11). Note: Eq. (11) seeks to achieve a tradeoff between the fog server load and latency.

$$\min_{j \in M, \rho \in path} (\text{load}_{(j)} + \text{latency}_{(\rho)}), \tag{11}$$

$$\text{where} \qquad \text{latency}_{(\rho)} = \sum_{l \in L} \text{delay}_l, \tag{12}$$

  where ρ is a path between every two OpenFlow switches, $latency_{(\rho)}$ is the latency of path ρ, *path* is the set of paths available between every two OpenFlow switches, L is the set of links that belong to path ρ and $delay_l$ is the delay of each link l in that path.

  Afterwards, the SDN controller sends the *ip* address of the selected fog server and the updated flow tables to the requesting and intermediate OpenFlow switches.

- If the load of all fog servers in a cluster reached the overloaded threshold and the response of SDN controller was negative, the local load balancer sends a notification to the OpenFlow switch to inform it that the requests of future arriving tasks must be sent to the cloud directly.

  After a while, if the load of fog servers of this cluster decreased and reached the under loaded part, then the local load balancer sends another notification to the OpenFlow switch. This notification informs the OpenFlow switch that the load balancer is ready to provide necessary resources for the future task requests.

V. Simulation and Results

OMNeT++ version 4.6 and SUMO version 0.19.0 setting on Windows 7 are used to simulate the environment. The number of vehicles in the simulation environment is 1000 moving at speed 20 m/s. These vehicles use LTE to communicate with fog devices. The POX controller and OpenFlow protocol version 1.0 are used in this paper. For simplicity but without loss of generality, we considered a scenario with three clusters. Each cluster has one OpenFlow switch, three homogeneous fog servers and one local load balancer. The cloud computing layer contains three servers. To evaluate our work, we applied VANET with Fog-Cloud architecture [6] and Cloud architecture (i.e. VANET-Cloud). These architectures were compared to our proposed system in terms of resource utilization, response time, meeting the deadline and bandwidth usage. The results are as follows:

A. Senario1: The Percentage of Resource Utilization

This subsection explains the percentage of resource utilization of fog computing in 60 seconds using the proposed system and fog-cloud system. The numbers of tasks are 50, 100, 150 and 200 tasks with a fixed size of 1000Mb. The bandwidth of the link between the fog and cloud computing is 100Mbps. Fig. 4. illustrates that the proposed system is the best because it selects an appropriate fog server to each task while in the Fog-Cloud architecture, if a fog server cannot handle a task, it does not check other fog servers and sends this task directly to the cloud. Therefore, the resource utilization of fog servers decreases according to (2).

B. Scenario 2: The Average Response Time

This subsection investigates the task response time. The tasks sizes are 0, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000Mb. For each task size, 100 tasks were used and then the average response time was computed. The bandwidth of the link between the fog and cloud computing is 150Mbps. Fig. 5. shows that the proposed system is better than other systems since it reduces the number of tasks handled by cloud servers, and therefore shortens the transmission time. The Fog-Cloud system sends some tasks to the cloud servers and thus the transmission time is high. In the Cloud system, the transmission time is very high, but the processing time is low due to the strength of cloud servers. Therefore, the response time is not significantly higher than that of other systems.

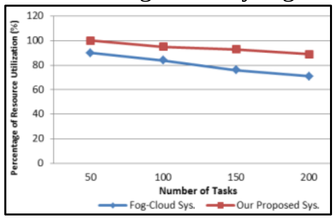

Fig. 4. Percentage of resource utilization.

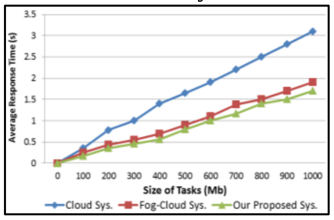

Fig. 5. Average response time

C. Scenario 3: The Percentage of Meeting the Deadline

This scenario investigates the percentage of tasks that meet the deadlines. The numbers of tasks that used here are 50, 100, 150 and 200 tasks with a fixed size of 1000Mb. The bandwidth of the link between the fog and cloud computing is 200Mbps. According to Fig. 6., the percentage of tasks that meet deadlines using the proposed system is higher than that of other systems since it is able to handle almost all tasks in the fog servers. Therefore, it does not incur a high transmission delay. Also, this percentage is acceptable in the Fog-Cloud system as it processes most of tasks in the fog servers. However, it is small in the Cloud system due to the distance between vehicles and cloud.

D. Scenario 4: The Percentage of Bandwidth Usage

The operations of sending tasks to the cloud computing need to high bandwidth. Therefore, it is important to handle the tasks at the network edge. This subsection deals with the percentage of using the bandwidth of the link that connects the fog computing layer or IoV layer to the cloud computing. In this subsection, we used 100 tasks with a fixed size of 1000Mb and the uplink bandwidth to the cloud computing was 95, 105, 115 and 125 Mbps. As shown in Fig. 7., the percentage of link bandwidth usage in the Cloud system is high since all tasks are sent to the cloud servers. In the Fog-Cloud system, some (not all) tasks are sent to the cloud servers, so the percentage of bandwidth usage is lower than that of Cloud system. However, this percentage is very good in our proposed system since it almost handled all tasks in the fog computing layer. Therefore, it is well suited for networks with high limited bandwidth.

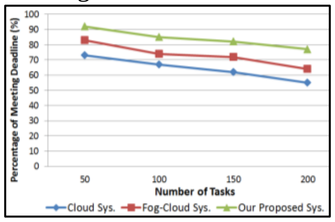

Fig. 6. Percentage of meeting deadline

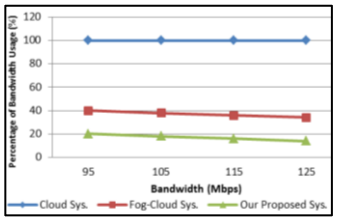

Fig. 7. Percentage of bandwidth usage

VI. Conclusions

A new architecture was presented by merging the clustering principle and fog computing with SDN for efficient utilization of fog resources. It helps handle tasks launched from the IoV sensors and decrease the transference of these tasks to the cloud to decrease the response time. It reduced the required bandwidth between the fog and cloud computing. Moreover, local and global load balance approaches were proposed to decrease the number of idle fog servers. The results show that the proposed system is more effective than Fog-Cloud and VANET-Cloud systems. It is more suitable for networks with limited bandwidth and for tasks with high sensitive delay.

References

[1] X. He, Z. Ren, C. Shi, and J. Fang, “A Novel Load Balancing Strategy of Software-Defined Cloud/Fog Networking in the Internet of Vehicles,” *China Commun.*, vol. 13, no. 2, pp. 140-149, 2016.

[2] O. Kaiwartya, A.H. Abdulla, Y. Cao, A. altameem, M. prasad, C. Lin, and X. Liu, “Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects,” *IEEE Access*, vol. 4, no. 99, pp. 5356-5373, Sep. 2016.

[3] K. Bilal, O. Khalid, A. Erbad, and S.U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Netw.*, vol. 130, pp. 94-120, Jan. 2018.

[4] C. Li, Z. Qin, E. Novak, and Q. Li, "Securing SDN Infrastructure of IoT-Fog Networks From MitM Attacks," *IEEE Internet Things*, vol. 4, no. 5, pp. 1156-1164, Oct. 2017.

[5] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, “Resource Provisioning for IoT Services in the Fog,” *in Proc. 2016 IEEE 9th Int. Conf. on Service-Oriented Computing and Applications (SOCA)*, pp. 32-39, Macau, China, Nov. 2016.

[6] R. Deng, R. Lu, C. Lai, T.H. Luan, and H. Liang, “Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption,” *IEEE Internet Things*, vol. 3, no. 6, pp. 1-11, Dec. 2016.

[7] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, “Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System,” *IEEE Trans. on Computer*, vol. 65, no. 12, pp. 1-11, Dec. 2016.

[8] S. Verma, A.K. Yadav, D. Motwani, R.S. Raw, and H.K. Singh, “An efficient data replication and load balancing technique for fog computing environment,” *in Proc. 2016 IEEE 3rd Int. Conf. on Computing for Sustainable Global Development (INDIACom)*, pp. 5092-5099, New Delhi, India, Mar. 2016.

[9] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, “Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning,” *China Commun.*, vol. 13, no. 3, pp. 156-164,Mar. 2016.

[10] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, “Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets,” *IEEE Internet Things*, vol. 4, no. 5, pp. 1216-1228, May 2017.

[11] K. PushpaLatha, R. S. Shaji, and J. P. Jayan, “A Cost Effective Load Balancing Scheme for Better Resource Utilization in Cloud Computing,” *J. emerg. techol.* web *intell.*, vol. 6, no. 3, pp. 280-290, Aug. 2014.

[12] E. Al-Ezaly, M. Abu-Elkeir, and A. Riad, “Collaborative Vehicle Location Management Service for Enhanced Hybrid Reactive and Proactive Multicast in VANETs,” *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 691-704, Feb. 2017.