A Novel Load Balancing Strategy of Software-Defined Cloud/Fog Networking in the Internet of Vehicles

Xiuli He[1] , Zhiyuan Ren[1]*, Chenhua Shi[1], Jian Fang[2]

[1]The State Key Laboratory of ISN in School of Telecommunications Engineering, No.2, South Taibai Road, YanTa District, Xidian University, Xi'an, Shaanxi, 710071, China

[2]The State Radio Monitoring Center and Testing Center, No.80, North Lishi Road, Xicheng District, Beijing, 100032, China

Abstract: The Internet of Vehicles (IoV) has been widely researched in recent years, and cloud computing has been one of the key technologies in the IoV. Although cloud computing provides high performance compute, storage and networking services, the IoV still suffers with high processing latency, less mobility support and location awareness. In this paper, we integrate fog computing and software defined networking (SDN) to address those problems. Fog computing extends computing and storing to the edge of the network, which could decrease latency remarkably in addition to enable mobility support and location awareness. Meanwhile, SDN provides flexible centralized control and global knowledge to the network. In order to apply the software defined cloud/fog networking (SDCFN) architecture in the IoV effectively, we propose a novel SDN-based modified constrained optimization particle swarm optimization (MPSO-CO) algorithm which uses the reverse of the flight of mutation particles and linear decrease inertia weight to enhance the performance of constrained optimization particle swarm optimization (PSO-CO). The simulation results indicate that the SDN-based MPSO-CO algorithm could effectively decrease the latency and improve the quality of service (QoS) in the SDCFN architecture.

Key words: internet of vehicles; cloud computing; cloud/fog network; software defined networking; load balancing

I. INTRODUCTION

The Internet of Vehicles (IoV) is a typical application of Internet of Things (IoT) technology in the field of the intelligent transportation system (ITS). The IoV not only carries on wireless communication between vehicles and vehicles, roads, pedestrians and the Internet, but also realizes intelligent traffic management, intelligent dynamic information service and intelligent vehicle control network integration. Various services are expected to be provided in the IoV, such as collision warning, traffic congestion detection, route planning, infotainment, and etc, which make human traffic travel more convenient. Therefore, a powerful data center is necessary to support those applications.

Cloud computing is considered as a computing paradigm that enables providing computing resources on "pay-as-you-go" basis[1], and is easily accessed by users from everywhere and at any time through the

Internet[2]. Recently, it has been used for data storage, data processing and data analysis in the IoV. Meanwhile, some applications about vehicular networking have been deployed to the cloud to provide related services for customers. However, the number of vehicles and mobile terminals increase exponentially, which makes the burden of the cloud heavier. Besides, the cloud computing data centers are relatively far from end users, leading to high processing latency. That becomes a serious problem for the latency-sensitive applications in the IoV, e.g., an ambulance needs to obtain its surrounding traffic information in real time to arrive at the rescue place timely, and a running car needs instantaneous information for collision warning.

In order to lighten the burden of the cloud and decrease task processing latency, in this paper, we propose a novel network architecture which integrates the cloud computing, the fog computing and the software defined networking (SDN) entirely. SDN as an emerging networking paradigm, is one of the most popular research fields in IT industry[3]. And the character that is controlling the network in a systematic, centralized and programmable manner by decoupling the data plane and control plane, makes SDN an attractive approach to the IoV[4]. Fog computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers[5], which has been considered as a cloud close to end users in order to offer computing and services with less latency[3]. Nevertheless, fog network consists of numerous devices which own weak computing capacity in reality. And one single device may difficult to deal with large amounts of data effectively. Thus, it is necessary to execute distributed computing in fog network by using massive network equipment.

The load balancing as an important component in the distributed computing technology, has attracted worldwide attentions. For balancing load between different entities, the tasks are assigned to many entities according to one policy, which could minimize processing time. In recent years, with the development of cloud computing, academic circles begin to study the load balancing strategies of cloud computing [6-8]. Although fog is considered as a cloud close to the ground[5], load balancing strategies of cloud computing cannot be directly adopted in the fog network since the heterogeneity of fog. Until now, a multi-band load balancing strategy[9] has been proposed for the fog network, which could control mobile devices access to the band with less connections. The mechanism can make full use of wireless spectrum resources, balance the number of users between two bands to alleviate the network congestion. But the load balancing of task processing is not taken into account in fog network. In addition, a dynamic load balancing mechanism based on graph repartitioning of fog computing is proposed in [10], which forms the virtual machine nodes graph model firstly and then provide services to the users by graph partitioning and clustering. Yet it may consume much time and network resources to reach a new balance after a fog node is out of order, and it only considers the load balancing of fog network. Besides, the tradeoff between power consumption and delay in a cloud-fog computing system is studied in [11], which develops an approximate solution to decompose the primal problem into three corresponding sub-problems. On the basis of the above, we use SDN for centralized control fog network in the IoV, which can help us get the required information before load balancing. Considering that though the cloud is far from users, it also can be responsible for part of the task. Thus, we integrate cloud, SDN controller, and fog to form the software defined cloud/fog networking (SDCFN) architecture for applying to the IoV. To achieve optimal load balancing, we modify constrained optimization particle swarm optimization (PSO-CO). Experiments have proved that the modified constrained optimization particle swarm optimization (MPSO-CO) algorithm decreases latency and enhances the quality of service (QoS) effectively, which could apply to the IoV to process latency-sensitive tasks more efficiently.

The rest of this paper is organized as follows. In the section II, we introduce the SDCFN architecture in the IoV scenario. In the section III, we propose a theoretical analysis model of load balancing in the SDCFN and propose a novel MPSO-CO algorithm based on SDN centralized control to solve the problem of the load balancing. Next, in section IV we show the simulation result to evaluate the high efficiency of the SDN-based MPSO-CO algorithm applied to the SDCFN. Finally, we conclude this paper in section V.

II. SDCFN Architecture

This section provides a brief introduction of the IoV scenario and describes the SDCFN architecture and its functions.

The IoV consists of vehicles, road side infrastructure and provides vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-base station communications. Vehicles in the IoV are highly mobile usually, which means they have a high demand for services processing time, especially the latency-sensitive services, i.e., the measurement of the distance between vehicles, the ambulance real-time monitor of road conditions, the route map query, and etc. Although traditional cloud computing architecture could support those services, the latency is extremely high since the cloud center servers are far from the vehicles. In order to solve this problem, we introduce the fog network into the cloud-based architecture, which could satisfy low latency requirement. And SDN controllers are also necessary because of its flexible centralized control, which could gather the global load conditions of all fog devices and the cloud. The overall architecture is shown in Fig.1.

The network architecture is divided into four layers: cloud computing layer, SDN control layer, fog computing layer and infrastructure layer.

The infrastructure layer is made up of vehicles with on board units (OBUs) installed. The OBUs comprise processing units, sensors, localization systems (e.g., global positioning system (GPS)), one radio transceiver for wireless access in vehicular environment (WAVE), and another radio transceiver such as WiMAX/3G/4G LTE to communicate with cellular base station (BS).

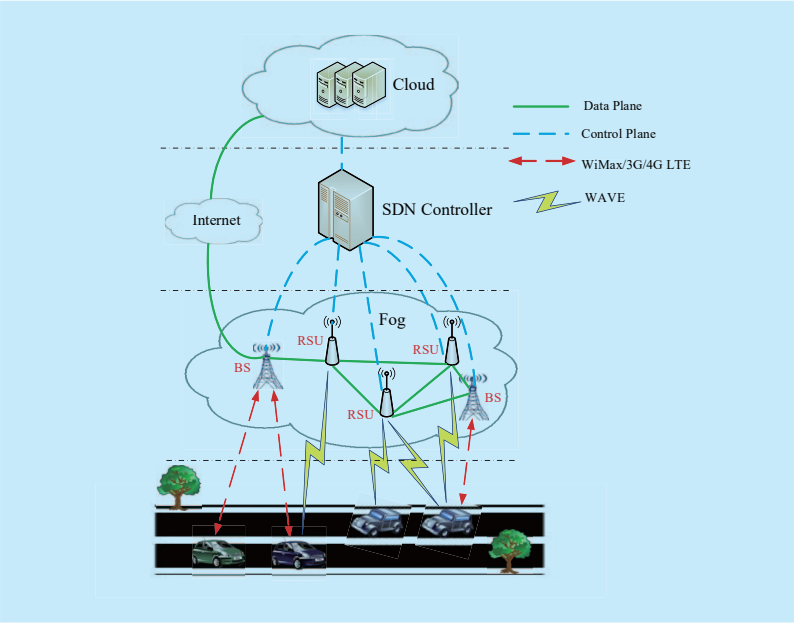

Fig.1 *Software defined cloud/fog network architecture*

The fog computing layer consists of BSs and road side units (RSUs) with the capability of computing and storage. Both of the BSs and RSUs run openflow protocol to communicate with the SDN controllers. In the SDCFN architecture, BSs and RSUs are collectively referred to as fog nodes (FNs). A FN not only can get required data and service from cloud severs by means of active caching, but also can obtain the traffic condition information from its surrounding FNs and store useful information transmitted over it. Moreover, the working conditions and dynamic information of vehicles which are collected by FNs could be uploaded to the cloud for global information sharing. However, tasks are processed only on a single BS or RSU could not meet the requirement of low latency because of large numbers of connection vehicles and numerous tasks. Thus, it is necessary to carry out the distributed computing to balance load, and then decrease latency.

SDN control layer is composed of SDN

controllers. The key technology of SDN is decoupling data plane and control plane. Here, the SDN controllers interact with the FNs and cloud through the openflow protocol, send flow table to cloud/fog network and set data forwarding rule to control the cloud/fog network centrally and globally. Since SDN controllers could gather the global information of the SDCFN including load, processing speed and communication latency, it can formulate optimal load balancing strategies for the network. Meanwhile, SDN controllers provide open programming interfaces to support network routing and resources management, users access control by software programming.

Cloud computing layer is composed of high performance server clusters, which is able to store, analyze the collected vast amounts data from a variety of terminal equipments and provide all sorts of comprehensive services. In this novel SDCFN architecture, cloud may be seen as a computing entity to enhance the overall performance.

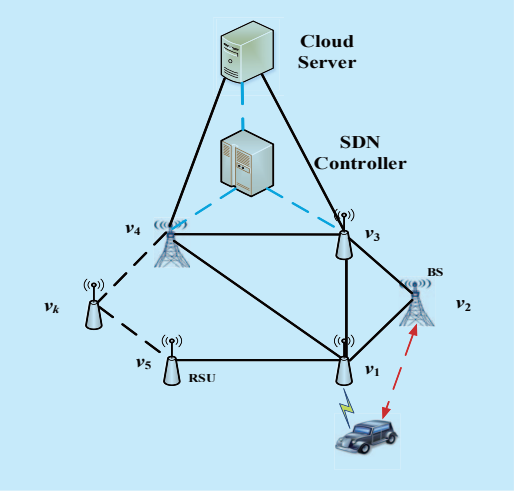

Fig.2 *Topology graph of SDCFN*

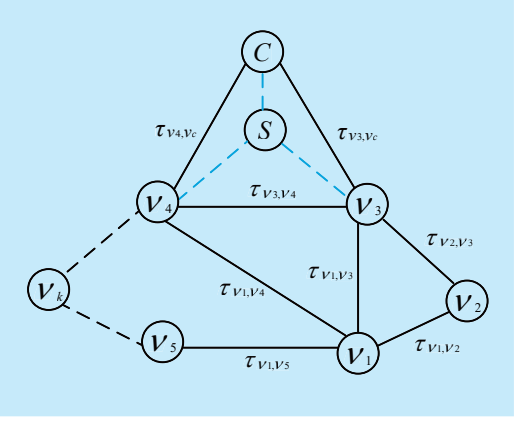

Fig.3 *Weighted undirected graph of SDCFN*

III. The SDCFN Load Balancing Algorithm

In this section, we describe the proposed theoretical model of SDCFN and the MPSO-CO load balancing algorithm.

3.1 Theoretical model of SDCFN

The fog network is composed of fragmented network devices with weak power of computation. Therefore, how to allocate computing tasks and balance the load of FNs according to equipment performance and communication overhead for latency reduction is an important question in the IoV, especially for latency-sensitive services. For tackling the above problem, a novel load balancing strategy of SDCFN is researched in this paper.

In the IoV scenario, we consider the SDCFN consists of vehicles, FNs, cloud servers and SDN controllers. The network structure diagram is demonstrated in Fig.2.

Based on the graph theory, the network topology in Fig.2 can be formulated as a weighted undirected graph $G=(\boldsymbol{V}, \boldsymbol{E})$ [12], which is shown in Fig.3.

In Fig.3, $\boldsymbol{V}=\{v_1, v_2, \cdots, v_k, S, C\}$ is a vertex set. Node $v_i \in V$, S and C represent RSU(BS), SDN controllers and cloud servers respectively. $\boldsymbol{E}=\{e_{v_1,v_2}, ..., e_{v_i,v_j}, ..., e_{v_{k-1},v_k}, e_{v_3,c}, e_{v_4,c}\}$ represents an edge set, and $e_{vi,vj}$ denotes the communication link between two nodes. For the purpose of performance improvement by using the cloud computing center, we consider cloud center as a distributed computing node and assume it has the computing capability C_d. Meanwhile, the computing capability of v_i is C_{vi}, and the weight $\tau_{vi,vj}$ denotes the communication latency between node v_i and v_j.

When a task, i.e., U, transferred from end users to the RSU/BS, it should be divided into

several sub tasks, i.e., u_i firstly, that means $u_i = \delta_i U$. Secondly, u_i would be transferred to the fog node v_i or cloud node C for calculation. Of course, the node that distributes the task, will also execute part of the task. And finally, the results should be merged and sent back to the end users. Therefore, the total latency t of the U in SDCFN can be described as:

$$t(\delta_i, \delta_{Cd}) = \max\left\{\frac{\delta_i U}{C_{v_i}} + \tau_{v_i,v_j} m_{v_i,v_j}, \frac{\delta_{Cd} U}{C_d} + \tau_{v_i,c} m_{v_i,c}\right\} \quad i,j = 1,2,...,k \tag{1}$$

Here, $\delta_i U/C_{vi}$ denotes the computing latency of u_i on fog node v_i, and $\tau_{vi,vj} m_{vi,vj}$ denotes the communication latency between fog node v_i and v_j. Furthermore, $m_{vi,vj}$ denotes there is a relationship of sub task allocation or not between fog node v_i and v_j. When $m_{vi,vj}$ equals 1, it means that the relationship exists; when $m_{vi,vj}$ equals 0, it means that the relationship does not exist. Similarly, $\delta_{Cd} U/C_d$ represents the computing latency of u_{C_d} on the cloud. And $\tau_{vi,c} m_{vi,c}$ is the communication overhead between fog node v_i and cloud. For the purpose of minimizing the whole latency, the optimal set of $\{\delta_i, \delta_{Cd}\}$ needs to be identified. To sum up, the problem can be modeled as follows:

$$\min\left\{\max\left[\frac{\delta_i U}{C_{v_i}} + \tau_{v_i,v_j} m_{v_i,v_j}, \frac{\delta_{Cd} U}{C_d} + \tau_{v_i,c} m_{v_i,c}\right]\right\} \quad i,j = 1,2,...,k$$

$$s.t.\ m_{v_i,v_j}, m_{v_i,c} = \begin{cases} 1, & \delta_i \text{ or } \delta_{Cd} \neq 0 \\ 0, & \delta_i \text{ or } \delta_{Cd} = 0 \end{cases}, \sum_{i=1}^{k} \delta_i + \delta_{Cd} = 1 \tag{2}$$

3.2 SDN-based MPSO algorithm

For the model of SDCFN in section 3.1, it is necessary that a set of optimal load distribution coefficients $\{\delta_i, \delta_{Cd}\}$ should be identified to minimize the latency t. In the SDCFN, the sub task, i.e., u_i, allocated to each FN, satisfies $u_i = \delta_i U$. Therefore, tasks allocated to FNs and cloud can form a k+1 dimensional vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_k, u_{Cd})^{\hat{O}}$. Assuming that FN v_1 receives the current task and needs to assign sub tasks to other FNs, from Eq. (1), the total latency t can be mentioned as follows:

$$t(\boldsymbol{u}) = \max\left\{\frac{u_1}{C_{v_1}} + \tau_{v_1,v_1} m_{v_1,v_1}, \ldots, \frac{u_k}{C_{v_k}} + \tau_{v_1,v_k} m_{v_1,v_k}, \frac{u_{Cd}}{C_d} + \tau_{v_1,c} m_{v_1,c}\right\} \tag{3}$$

Therefore, solving the mapping of computing task, i.e., u_i to each processing node, namely the solution to vector $\boldsymbol{u}$, could be concluded as a following optimization problem:

$$\begin{aligned} \boldsymbol{u} &= \arg\min_{\boldsymbol{u} \in \boldsymbol{I}} \{t(\boldsymbol{u})\} \\ s.t.\ & 0 \le u_i \le U, i = 1,..,k \\ & 0 \le u_{Cd} \le U \\ & \sum_{i=1}^{k} u_i + u_{Cd} = U \end{aligned} \tag{4}$$

And the searching space I is:

$$\boldsymbol{I} \triangleq \prod_{i=1}^{k} [U_{i\min}, U_{i\max}] = \prod_{i=1}^{k} [0, U] \tag{5}$$

Since particle swarm optimization (PSO) [13] algorithm has the advantage of strong global search ability, fast convergence, easy programming, we intend to use PSO to solve the optimization problem in Eq. (4). In the process, particles in the swarm $\{\boldsymbol{X}_i^l\}_{i=1}^{s}$ move in search space I to find the best position $\boldsymbol{X}$, i.e., $\boldsymbol{u}$, where s denotes the size of the swarm and $l \in \{0,...,L_{\max}\}$ is the iteration number. Each particle position is $\boldsymbol{X}_i^l = (x_{i1}^l, x_{i2}^l, ..., x_{ik}^l)$. The velocity vector is $\boldsymbol{v}_i^l = (v_{i1}^l, v_{i2}^l, ..., v_{ik}^l)$, and $\boldsymbol{v}_i^l \in \boldsymbol{M}$, $\boldsymbol{M} \triangleq \prod_{i=1}^{k} [-v_{i\max}, v_{i\max}]$, where $v_{i\max} = (1/2)(U_{i\max} - U_{i\min})$.

However, the standard PSO algorithm cannot be used to solve the constrained optimization problem directly. Thus, we leverage the constrained optimization PSO (PSO-CO) [14] which transforms the above problem into an unconstrained optimization problem to resolve. Unlike the standard PSO algorithm, the following fitness function $f(\boldsymbol{X})$ is adopted to evaluate the fitness of particles,

$$f(\boldsymbol{X}) = \begin{cases} t(\boldsymbol{X}) & \boldsymbol{X} \in \boldsymbol{F} \\ t(\boldsymbol{X}) + r\sum_{j=1}^{k+2} t_j(\boldsymbol{X}) + \varphi(\boldsymbol{X}, l) & \boldsymbol{X} \in \boldsymbol{I} - \boldsymbol{F} \end{cases} \tag{6}$$

In Eq. (6), F denotes the feasible region in the searching space I, which means I - F is the infeasible region. r represents the penalty factor, and $t_j(\boldsymbol{X})$ denotes the constraint violation measure of the infeasible particles on the jth constraint. Moreover, $\varphi(\boldsymbol{X}, l)$ denotes additional

heuristics value for infeasible particles in the lth generation of the algorithm [14]. $t_j(\boldsymbol{X})$ and $\varphi(\boldsymbol{X},l)$ could be described as Eq. (7) and Eq. (8) respectively.

$$t_j(\boldsymbol{X})=\begin{cases}\max\left(0,-\boldsymbol{X}(j)\right) & 1\le j\le k+1\\ \left|\sum_{i=1}^{k}\boldsymbol{X}(i)-U\right| & j=k+2\end{cases} \tag{7}$$

$$\varphi(\boldsymbol{X},l)=O(l)-\min_{\boldsymbol{X}\in\boldsymbol{I}-\boldsymbol{F}}\left\{t(\boldsymbol{X})+r\sum_{j=1}^{k+2}t_j(\boldsymbol{X})\right\} \tag{8}$$

$$O(l)=\max\left\{O(l-1),\max_{\boldsymbol{X}\in\boldsymbol{F}}\left\{t(\boldsymbol{X})\right\}\right\} \tag{9}$$

$O(l)$ in the Eq. (9) records the feasible particles with the maximum fitness value after the evolution of the lth generation, which ensures that all the feasible particles are better than all the infeasible particles in the iterative process.

When the PSO-CO algorithm is performed, the particle swarm should be initialized firstly. The particle swarm $\left\{\boldsymbol{X}_i^0\right\}_{i=1}^{s}$ needs to be initialized randomly by the values in the searching space I. The speed of particles is initialized to 0, i.e., $\left\{\boldsymbol{v}_i^0=0\right\}_{i=1}^{s}$. Then, the position and velocity of particles should be updated according to the following formula.

$$\boldsymbol{v}_i^{l+1}=\omega\boldsymbol{v}_i^l+rand(\,)c_1\left(\boldsymbol{p}_i^l-\boldsymbol{X}_i^l\right)+rand(\,)c_2\left(\boldsymbol{g}^l-\boldsymbol{X}_i^l\right) \tag{10}$$

$$\boldsymbol{X}_i^{l+1}=\boldsymbol{X}_i^l+\boldsymbol{v}_i^{l+1} \tag{11}$$

In the Eq. (10), ω is inertia weight. *rand*() is random function and its value distributed in the interval [0,1] uniformly. Both of c_1 and c_2 are acceleration factor, which represent the weight of velocity of the particles flying to the local and global best position respectively. What follows is calculating the fitness value of particles in the lth iteration process according to the function $f(\boldsymbol{X})$, saving the optimal position $\boldsymbol{p}_i^l$ of the particle i in the search space and the best location g^l of all the particles in the whole group and updating $\boldsymbol{p}_i^l$ and $\boldsymbol{g}^l$ according to fitness values.

In the PSO-CO algorithm implementation process, premature convergence will appear due to the decline of population diversity. In order to avoid falling into local optimum, we introduce the reverse of the flight of mutation particles [15] to the PSO-CO algorithm, which called MPSO-CO algorithm in this paper. The velocity and position of the mutation particles in the MPSO-CO algorithm are updated as follows:

$$\boldsymbol{v}_i^{l+1}=-\boldsymbol{v}_i^l \tag{12}$$

$$\boldsymbol{X}_i^{l+1}=\boldsymbol{X}_i^l-\boldsymbol{v}_i^{l+1} \tag{13}$$

Moreover, inertia weight ω updates according to the following Eq. (14).

$$\omega=\omega_{\min}-\frac{\omega_{\min}-\omega_{\max}}{H}\times h \tag{14}$$

Where ω uses the strategy from 0.9 linear decline to 0.4, $\omega\in[0.4,0.9]$, $\omega_{\min}=0.4$, $\omega_{\max}=0.9$, H is the maximum of iterative number, h is the current number of iterations.

From the analysis of the above MPSO-CO algorithm, it is obvious that load, computing capacity and transmission latency of all FNs in the SDCFN should be known before load balancing. Therefore, a central node is necessary which can get the whole network real-time information to assist the MPSO-CO to make the best load balancing strategy. Since the SDN controllers can gather the information of all FNs and the cloud, make an optimal load balancing strategy based on the information and inform FNs the strategy by sending flow table. Therefore, we introduce SDN to achieve centralized load balancing. The SDN-based MPSO-CO algorithm is shown in the Algorithm 1.

Algorithm 1 SDN-based MPSO-CO algorithm

Input: U, C_{vi}, C_d, $\tau_{vi,vj}$, $\tau_{vi,c}$, $O(0)$ (all get from SDN controllers)

Output: $t(\boldsymbol{X})$

1: for each particle i

2: Initialize position $\boldsymbol{X}_i^0$ and velocity $\boldsymbol{v}_i^0=0$;

3: end for

4: for l in iteration

5: for each particle i

6: Calculate the value of $t_j(\boldsymbol{X})$ using equation (7),(8),(9);

7: end for

8: for each particle i

9: Update the velocity and position of particle i;

Table I *The related parameters of SDCFN*

Parameter type	C	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
Cvi/Cd (Gbps)	10	2	1	1	0.5	0.4	0.7	0.6	0.3	0.5	1
Up-link Bandwith(Mbps)	2	83.7	85	70	78	86	90	88	89	87	78
Down-link Bandwith(Mbps)	1.9	101.9	101	99	100	102	105	103	104	102	98
Other delay in transmission(s)	0.01	0.001	0.001	0.0012	0.0011	0.0012	0.001	0.0013	0.001	0.0013	0.001

10: if sum($\boldsymbol{X}_i^l$)= U and $\boldsymbol{X}_i^l \geq 0$ and $\boldsymbol{X}_i^l \leq$U

11: compute_fit_feasible (particle i, C_{vi}, C_d , $\tau_{vi,vj}$, $\tau_{vi,c}$);

12: else

13: compute_fit_infeasible(particle i, $t_j(\boldsymbol{X})$, $\varphi(\boldsymbol{X},l)$, C_{vi}, C_d , $\tau_{vi,vj}$, $\tau_{vi,c}$);

14: Update the value of $\boldsymbol{p}_i^l$ and $\boldsymbol{g}^l$;

15: The number of mutation particles n=random();

16: for each particle in n

17: Update the velocity and position of mutation particle according to equation (12),(13);

18: end for

19: Update inertia weight ω;

20: end for

21: end for

22: Calculate the value of $t(\boldsymbol{X})$ using equation (3).

IV. Numerical Results

Without loss of generality, we let the number of FN in fog layer equals 10. The related parameters of SDCFN in the simulation are given in Table I. Taking the actual situation into account, communication latency $\tau_{vi,vj}$ is equal to the sum of delivery latency and other latency. Other latency includes the latency generated during the transmission process except for the delivery latency. Combined with the real network environment, the values are all given in Table I.

The basic parameters of MPSO-CO algorithm are: the size of the swarm s is 50, maximum iteration number L_{max} is 1000, acceleration factor c_1 and c_2 are all equal to 1, inertia weight ω is 1.2, the penalty factor r is 10, control parameter $O(0)$ is 10^6.

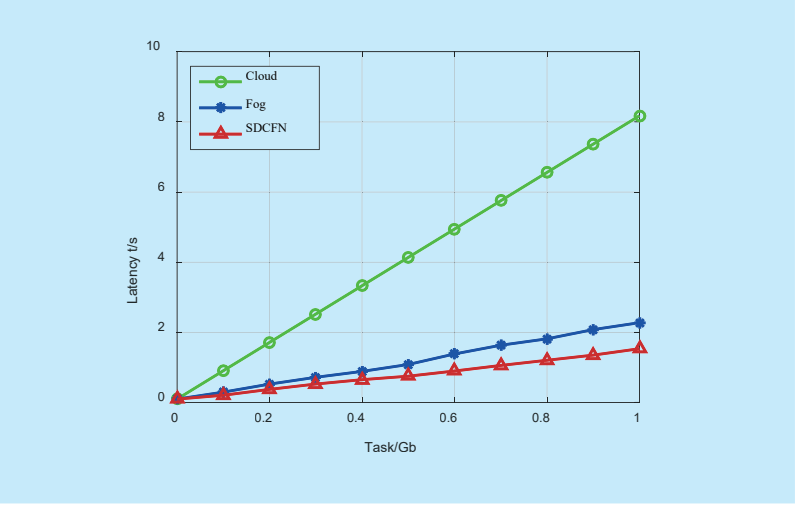

Fig.4 *Latency performance comparison of SDCFN vs. Fog and Cloud*

In the simulation scenario, the request data from vehicles received by v_1 or cloud called task in this paper, varies from 0Gb to 1Gb. And the requests include the distance between one vehicle and its neighbors, surrounding road traffic information, and etc. The capacity of these data is set according to the equal proportion. The simulation results are shown in section 4.1, 4.2, and 4.3.

4.1 Latency performance comparison of three architectures

Based on the MPSO-CO algorithm, we evaluate the latency performance of SDCFN compared with cloud network and fog network.

The comparison of latency is shown in Fig. 4. Since the computation speed of cloud is higher than FN, the difference of latency among cloud, fog and SDCFN is small when

the task load lower than 0.05Gb. However, with the increasing of task load, the latency of cloud is significantly higher than the other two architectures due to the increase of transmission latency. Compared with the cloud, the fog has lower latency since the cloud servers are far from the end users which increases the transmission latency significantly. Besides, in Fig.4, we could also find that when the task load is large, since the cloud server enhance the system computation speed, the latency of SDCFN is lower than the fog obviously. Because the computing capability of FN is limited in reality. Therefore, SDCFN can improve the QoS and user experience efficiently in latency-sensitive scenario.

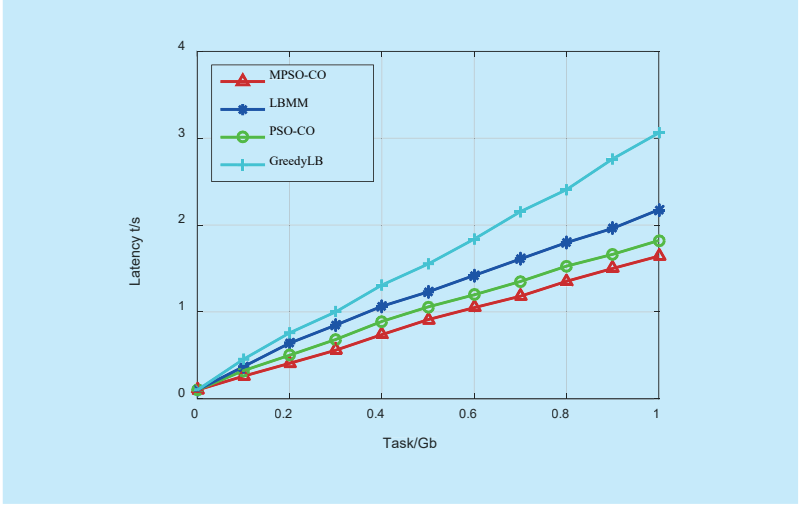

Fig.5 *Latency performance comparison of multiple load balancing algorithm*

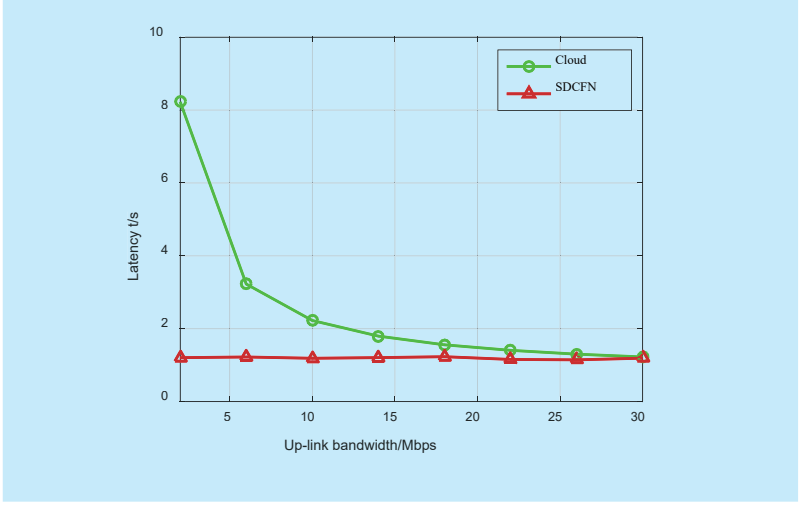

Fig.5 *Latency performance comparison of multiple load balancing algorithm*

4.2 Latency performance comparison of multiple load balancing algorithm

In this section, for the scenario of SDCFN, the high efficiency of the MPSO-CO algorithm is analyzed by comparing with other load balancing algorithms, such as PSO-CO[14], greedy load balancing algorithm (Greedy-LB) [16], and Max-Min load balancing algorithm (LBMM)[17].

Fig.5 describes the implemented results of four algorithms in the SDCFN scenario. In the PSO-CO algorithm, a new kind of fitness function is introduced to evaluate the infeasible particles on the basis of the standard PSO algorithm. However, it can't avoid the algorithm falling into local optimum. Therefore, we modify PSO-CO algorithm with the reverse of the flight of mutation particles, i.e., MPSO-CO algorithm, and simulation result is presented in Fig.5. From Fig.5, we can find that the MPSO-CO algorithm obtains lower latency compared to PSO-CO, LBMM, Greedy-LB, since the LBMM and Greedy-LB do not take the transmission latency into account, and PSO-CO may fall into the local optimum. Moreover, Greedy-LB does not consider the computing capacity of nodes, which leads to highest latency when Greedy-LB strategy is adopted. Therefore, applying the MPSO-CO algorithm to the SDCFN architecture can reduce the task processing latency of IoV more effectively.

4.3 Influence of up-link bandwidth for SDCFN

In fact, the obtained bandwidth varies in real time when the task load is uploaded to the cloud. Therefore, it is important to estimate the latency performance of bandwidth. In this part, we investigate how the up-link bandwidth of cloud affects the task processing latency in the SDCFN by comparing it with the cloud. The task load is fixed to 1Gb, and the result is shown in Fig.6. It is obviously that the latency

of cloud decreases when the up-link bandwidth increases, and it will be equal to the SDCFN when the up-link bandwidth higher than 28Mbps. Compared with cloud, the SDCFN keeps in a low level, which means SDCFN is more suitable for latency-sensitive services in the IoV.

V. Conclusions

In this paper, we have presented a novel architecture which integrates the fog computing and SDN to the IoV in order to improve the latency performance for processing latency-sensitive services. Then, we construct the theoretical model of the software defined cloud/fog network (SDCFN) by combining the graph theory and characteristics of fog network, SDN, and cloud. And on the basis, we propose a SDN-based modified constrained optimization particle swarm optimization(MPSO-CO) centralized load balancing algorithm to balance workload between cloud/fog devices, so that the task processing latency is effectively reduced. Simulation results prove that this algorithm is effective and can be applied to reduce latency and improve the QoS in the SDCFN. Moreover, the simulation results also verify that the SDCFN can be applied to process latency-sensitive services in the IoV more efficiently. In further research work, other load balancing algorithms those can be applied to the SDCFN will be studied. Besides, more aspects of QoS should be considered not only latency, such as security, capacity and etc.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China (No.61401331, No.61401328), 111 Project in Xidian University of China (B08038), Hong Kong, Macao and Taiwan Science and Technology Cooperation Special Project (2014DFT10320, 2015DFT10160), The National Science and Technology Major Project of the Ministry of Science and Technology of China (2015zx03002006-003) and Fundamental Research Funds for the Central Universities (20101155739).

References

[1] A. Eleyan, D. Eleyan. "Forensic Process as a Service (FPaaS) for Cloud Computing", European Intelligence and Security Informatics Conference (EISIC), pp 157-160, 2015.

[2] N. Alsaeed, M. Saleh. "Towards Cloud Computing Services for Higher Educational Institutions: Concepts & Literature Review", International Conference on Cloud Computing (ICCC), pp 1-7, 2015.

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

[4] N.B. Truong, G.M. Lee, Y. Ghamri-Doudane. "Software defined networking-based vehicular Adhoc Network with Fog Computing". IEEE, pp 1202-1207, 2015.

[5] F. Bonomi, R. Milito, J. Zhu, S. Addepalli. "Fog Computing and Its Role in the Internet of Things", First Edition of the Mcc Workshop on Mobile Cloud Computing, pp 13-16, 2012.

[6] Y. Sahu, R.K. Pateriya, R.K. Gupta. "Cloud Server Optimization with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm", International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, pp 527-531, 2013.

[7] S. Dam, G. Mandal, K. Dasgupta, P. Dutta. "Genetic algorithm and gravitational emulation based hybrid loadbalancing strategy in cloud computing",International Conference on Computer, Communication, Control and Information Technology (C3IT). IEEE, pp 1-7, 2015.

[8] N. Sowmya, M. Aparna, P. Tijare, N. Nalini. "An adaptive load balancing strategy in cloud computing based on Map reduce", International Conference on Next Generation Computing Technologies (NGCT). IEEE, pp 86-89, 2015.

[9] D.M. Cheng, Z. Li. "Fog computing based hospital information service system". Computer Science, vol.42, no.7, pp 170-174, 2015.

[10] S. Ningning, G. Chao, A. Xingshuo, Z. Qiang. "Fog Computing Dynamic Load Balancing Mechanism

Based on Graph Repartitioning". China Communications, vol.13, no.3, pp 156-164, 2016.

[11] R. Deng, R. Lu, C. Lai, T.H. Luan. "Towards Power Consumption-Delay Tradeoff by Workload Allocation in Cloud-Fog Computing", IEEE International Conference on Communications (ICC), pp 3909-3914, 2015.

[12] E. Robert, M. Burkhard, P. Robert. "Diffusion Schemes for Load Balancing on Heterogeneous Networks". Theory of Computing Systems, vol.35, no.3, pp 305-320, 2002.

[13] J. Kennedy, R.C. Eberhart. "Particle swarm optimization", IEEE International Conference on Neural Networks, pp 1942- 1948, 1995.

[14] X.Y. Li, P. Tian, M. Kong. "A New Particle Swarm Optimization for Solving Constrained Optimization Problems". Journal of Systems & Management, vol.16, no.2, pp 120-129, 2017.

[15] W.J. Liu, M.H. Zhang, W.Y. Guo. "Cloud Computing Resource Schedule Strategy Based on MPSO Algorithm", Computer Engineering, vol.37, no.11, pp 43-44, 2011.

[16] B. Sahoo, D. Kumar, S.K. Jena. "Performance analysis of greedy Load balancing algorithms in Heterogeneous Distributed Computing System", International Conference on High Performance Computing and Applications(ICHPCA), pp 1-7, 2014.

[17] N.S. Ghumman, R. Kaur. "Dynamic Combination of Improved Max-Min and Ant Colony Algorithm for Load Balancing in Cloud System", International Conference on Computing, Communication & Networking Technologies, pp 1-5, 2015.

Biographies

Xiuli He, received Bachelor's degree in Network Engineering from Agricultural University of Hebei Province, China in 2014. She is currently a master in School of Telecommunications Engineering, Xidian University, China. Her research interests include broadband wireless communications, fog computing and distributed computing.

Zhiyuan Ren, the corresponding author, email: zyren@xidian.edu.cn, received his M.S. and PhD. degrees from Xidian University, China in 2007 and 2011, respectively, in Communication and Information System. He is currently an associate professor in the School of Telecommunication Engineering, Xidian University, Xian 710071,China. His research interests include distributed computing, Mobile Edge Computing and Network Virtualization in 5G.

Chenhua Shi, received the B.A. degree in communication engineering from Hunan University, China, in 2015. She is pursuing the M.S. degree in communication and information systems from College of Telecommunication Engineering, Xidian University . Her research interests include fog computing and wireless communication.

Jian Fang, received the M.S. degree in Signal and Information Processing from Beijing University of Posts and Telecommunications, China in 2009. He is currently a senior engineer of the State Radio Monitoring Centre and vice-president of CCSA TC5 WG8. His research interests include frequency planning, electromagnetic compatibility analysis and cognitive radio.