Journal of Computers Vol. 30 No. 1, 2019, pp. 1-14
doi:10.3966/199115992019023001001

Services Preloading Scheme Based on Improved Threshold-PST in IOV

Wei-Hao Dong[1], Zhen-Jiang Zhang[2*], Jing He[3,4], Bo Shen[5]

[1] Department of Electronic and Information Engineering, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education Beijing Jiaotong University, Beijing 100044, China
15120062@bjtu.edu.cn

[2] Department of Software Engineering, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education Beijing Jiaotong University, Beijing 100044, China
zhangzhenjiang@bjtu.edu.cn

[3] Institute of Information Technology, Nanjing University of Finance and Economics, Nanjing 210023, China
lotusjing@gmail.com

[4] School of Software and Electrical Engineering, Swinburne University of Technology, Victoria 3122, Australia

[5] Department of Electronic and Information Engineering, Key Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education Beijing Jiaotong University, Beijing 100044, China
bshen@bjtu.edu.cn

Received 9 January 2018; Revised 22 January 2018; Accepted 22 January 2018

Abstract. As one of the core technologies of the internet of things and smart city, internet of vehicles (IOV) has been rapidly developed in recent years. And it has been playing a great role in improving urban traffic conditions. However, with the increasing size of vehicles and the types of applications, some bottlenecks have appeared such as high heterogeneity, poor mobility support and lack of extensibility. So how to develop the new generation of IOV has become one of the most important research topics. In this paper, we apply the software-defined network (SDN) and cloud computing to the heterogeneous IOV to bridge the gaps. The proposed architecture includes four layers, which can provide flexibility, scalability and mobile support which didn't exist in traditional ways. Moreover, a services preloading scheme based on trajectory prediction is proposed for effective resource management to minimize communication cost and transmission delay. In order to improve the effectiveness of the scheme, the improved threshold probability suffix tree algorithm is designed by training the historical trajectory and making the real-time prediction. We also evaluate the proposed approaches by experiments. The effectiveness and the efficiency are validated by the results.

Keywords: IOV, services preloading, software-defined network, trajectory prediction

1 Introduction

With the continuous improvement of communication technology, intelligent city has been vigorously developed [1]. Vehicles are considered to be the next intelligent solution that could have a significant impact on people's lives. The internet of vehicles (IOV) is a typical application of the Internet of Things (IOT) in the field of Intelligent Transportation Systems (ITS). It can provide a variety of services, such as crash warning [2], traffic jam detection [3] and so on. Although the IOV has a very good development prospect, there are still a lot of problems, such as poor mobility support and lack of extensibility. It is mainly caused by the lack of uniform standards of communication equipment and protocols [4-5]. At the

* Corresponding Author

same time, the emergence of new technologies and architectures provides an opportunity to improve these problems.

Several new architectures have been developed for the IOV, such as mobile priority network [6] and nebula [7]. Software definition network (SDN) [8] has become a hot topic because of its separation of control and data plane [9-10]. We are confident that it will greatly reduce the gap between application requirements and current limitations with its advantages. At the same time, cloud computing also provides an opportunity for the development of the IOV [11-12]. It allows users to easily access the cloud computing center from anywhere, anytime over the internet. Recently, it has gradually been used for data processing, storage and analysis in the internet of vehicles.

In this paper, we mainly focus on the following works: Firstly, we combine soft defined network, cloud computing with IOV to carry out a new architecture. Then, with the advantages brought by this architecture, we propose a services preloading scheme based on trajectory prediction. Finally, we set experiments to prove the accuracy of the trajectory prediction algorithm and the promotion of the services.

2 SDC-IOV Architecture

As shown in Fig. 1, the traditional IOV is mainly composed of vehicles, application servers and roadside units (RSU), providing vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and vehicle-to-base station (V2B) communication [13].

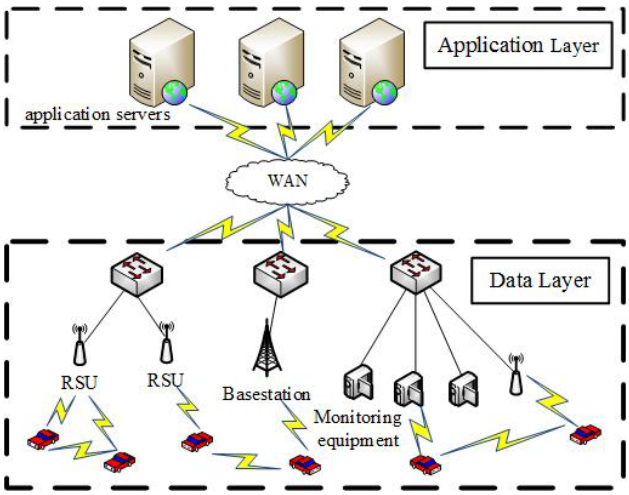

Fig. 1. Two-layer structure of traditional IOV

In this paper, we introduce SDN and cloud computing into the traditional IOV and propose a new architecture naming software-defined cloud internet of vehicles (SDC-IOV). This architecture is divided into four layers as shown in Fig. 2.

Cloud services layer. Different from the traditional dedicated deployment of the application servers, we deploy a unified cloud server on the top layer. It is composed by high-performance server clusters, which can store and analyze a variety of data and provide different services.

Control layer. In this layer, we regard the OpenFlow switches as controllers. The data is forwarded according to the flow tables. Because the controllers have the latest global view of the entire network, it can convert the policy applied in the cloud services layer (for example, path selection or access control) to rules in a particular switch so that the controllers can provide custom network for applications, which is not possible in traditional networks.

Infrastructure layer. We further divide the traditional data layer into two parts. This layer only includes access point (AP), roadside units, base stations and ordinary switches and servers without cars. They play a different role in the communication process and a car can be connected to multiple infrastructures at the same time.

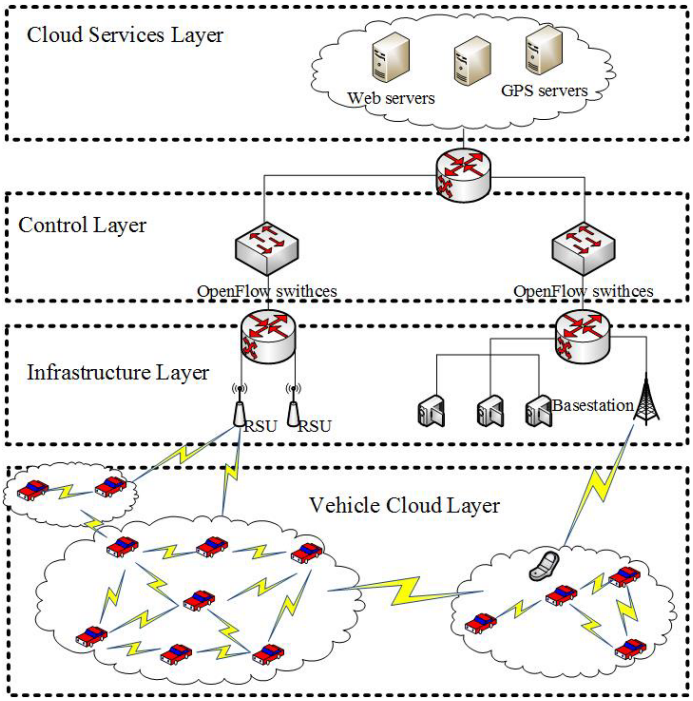

Fig. 2. Four-layer architecture of SDC-IOV

Vehicle cloud layer. Vehicles are divided into different sets according to certain principles. The vehicles in the same set can build up a small cloud through V2V communication. They are regarded as mobile cloud sites.

Compared with the traditional architecture, the new architecture we proposed has brought many unique advantages. We emphasize the following points:

Heterogeneous network integration. With the network virtualization and abstraction, all vehicles, roadside units and wireless infrastructures in the IOV can be regarded as SDN devices and managed by a unified interface, which significantly improves the heterogeneity.

Improve network resource utilization. Through the control center in cloud services layer and centralized control switches in control layer, we can customize the network and allocate network resources effectively. In addition, the control plane can also adjust the wireless transmission power, change the coverage area and reduce the probability of packet collision by the coordination of the transmission range.

Fast configuration. The control plane can adaptively choose protocols and adjust their parameters according to the rapidly changing external environment. In this way, existing vehicular communication protocols can be better utilized.

3 Services Preloading Scheme

3.1 The Latency of SDN in IOV

Although the SDN has a strong compatibility with the traditional network, due to the different design concepts, the structural differences between them have different effects in the practical application. In the traditional network, the transmission and control of data are done in the same plane. Each router not only needs to be responsible for forwarding data but also needs to control data forwarding. However, SDN sets up the control plane by separating control functions from the routers. The routers and switches in the data layer only need to forward data while the control of the network is done by OpenFlow switches in the control layer. The data in the network is forwarded based on the flow table in the switches. Therefore, the transmission of data in the network requires the interaction between the control plane and the data plane, and the corresponding delay is shown as follows:

$$T = t_s + t_{s\_c} + t_c + max\left\{t_{c\_s1}, t_{c\_s2}, ..., t_{c\_sn}\right\}. \quad \textbf{(1)}$$

In the above formula, t_s indicates that the processing delay of the first data packet in the switch which mainly includes the retrieval and maintenance of the flow tables. t_{s_c} indicates the time taken by the switch to send the controller a Packet-in message indicating that the first data packet is received. It is mainly affected by the distance between the controller and the switches. The controller needs to establish a processing solution for the first data packet to determine the transmission mode of the following data packet, and the delay is shown as t_c .Finally, after the controller sets up the control information, the flow table needs to be delivered to each underlying switches. This process is performed simultaneously so the delay should be the longest time for the sending process, as shown in the formula.

According to the above analysis, when the first data packet enters the network, the establishment of the flow table will result in the much higher delay than the subsequent data transmission [14]. Studies have shown that the average length of data streams in the network is 20 packets. Therefore, the delay caused by the interaction between the control plane and the data plane can't be ignored. This situation will be more prominent in the IOV. On the one hand, the number of nodes is large and the real-time services demanding is high. So a huge amount of data needs to be deal with, which usually requires multiple data streams. On the other hand, the prominent feature of vehicles as nodes is strong mobility. Whenever a vehicle drives to a new node, service acquisition needs to be repeated. This not only needs the forwarding between the underlying devices, but also needs the data transmission with the upper application, which greatly affected the real-time services. In order to improve the above problem, this paper proposes a new service preloading scheme.

3.2 Basic Model and Operating Mechanism

Service preloading refers to load the relevant flow table in advance to the underlying transmission equipment so the information transmission will be smooth by the use of new communications technology. The solution is based on the ideal assumption that, knowing the transmission requirement of services, flow tables matching the data transmission is pre-loaded on all underlying devices so that it doesn't require the deployment of control layer for the first packet. Users can get services directly. However, this extreme mechanism is not allowed in the real world. Firstly, the performance of switches in the software defined network is limited by the size of the flow tables. However, the number and types of vehicles in the car networking are numerous so the unified deployment of service preloading will seriously affecting the forwarding function of the switches. On the other hand, the unified deployment will also limit the flexibility of the network. The global view has become dispensable which does not meet the requirements of the internet of vehicles. Therefore, one of the main points of whether the program can be established is how to preload the services flexibly and properly.

One of the most important features of SDN is network programmability. The control system can be divided into two operation modes: reactive and proactive. The reactive mode is a passive network behavior. Only when the data flow enters the network, the corresponding flow tables will be established. When the first packet of the data stream appears, the switch will process and generate a corresponding message to the controller. The application running on the controller will interpret the message and generate control instructions which will be passed to the underlying devices participating in the transmission. So the matching flow tables will be passively installed on the devices. In the proactive mode, the application in controllers can be customized and control message can be proactively generated and sent to the underlying devices by artificial settings. Therefore, the data stream can be forwarded directly according to the generated rules.

The nodes in IOV are vehicles and users' mobile devices most probably. Compared with the nodes in traditional sensor network, they have a unique characteristic of predictable trajectory. Because their movement is often limited by the distribution of the traffic network in real world. By the analysis of movement patterns and historical trajectory information of vehicles, the future trajectory can be effectively predicted. This feature has been widely used in the field of transportation, such as driving route planning and radio advertising on board. Currently, researchers have developed macroscopic and microscopic models of trajectory prediction [15], providing the theoretical basis for IOV. At the same time, cloud computing is applied in our software-defined network architecture, which can efficiently handle and store a large amount of vehicle trajectory information steadily. The excellent computing ability not only speed the analysis of vehicles' trajectory, but also make the prediction results more accurate and credible compared with traditional applications.

Exactly based on the above situation and characteristics of c software-defined cloud internet of vehicles, this paper presents a new services preloading scheme based on vehicle trajectory prediction. In this project, we can predict the most probable driving route in a certain coming period of time by processing the trajectory information of the vehicle. Then, referring to the coverage of the underlying communications devices, the related devices on the predicted path are preloaded with the flow tables required for obtaining the services by the active deployment of the control layer. In this case, when the vehicle travels to this node, there is no need to repeat the process of requesting connection establishment to the control layer. When the first data packet is transmitted to the switch, the matched flow table already exists so the data streams can be forwarded directly. Thus the transmission delay is reduced and the quality of services is improved.

The model of our scheme is shown in the Fig. 3. The vehicle at node A obtains the real-time traffic information service from node C through the network connection and there is matching flow tables in the corresponding switches. When the vehicle moves, through the processing of the trajectory information by the cloud servers, the most possible destination can be predicted, which is node B. So, through preloading the flow tables matching the connection between node B and C by controllers, the car can get the real-time traffic at C directly according to the flow tables without communicating with the control layer again.

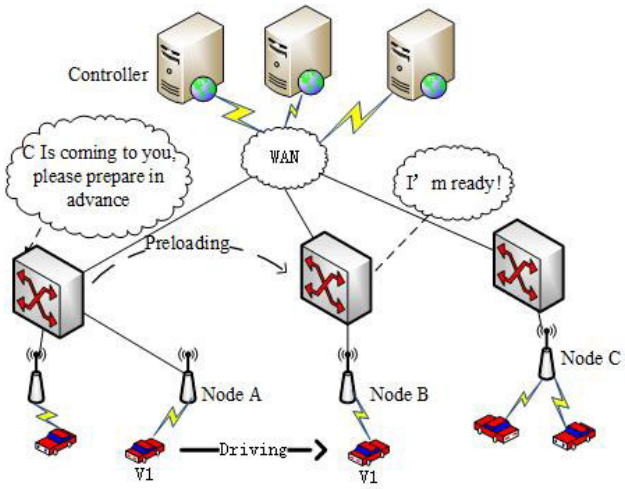

Fig. 3. The work mode of service preloading

It should be noted that, in order to ensure the performance of the switches, the timeout cleaning is used in our scheme. The essence of the new service preloading scheme in this paper is to preload services on the most probable path. It is not guaranteed to be completely accurate, the performance of switches will be affected by the additional installation of flow tables over time. By setting a timeout for flow tables, they will be deleted if they are not activated within this period. This period can be adjusted according to the actual traffic conditions.

It is not difficult to see from the above mechanism of services preloading that simple preloading is a kind of large-scale behavior of deploying services regardless of cost, which is not desirable in practice. Therefore, our solution limit the size of services preloading by trajectory prediction. It will greatly reduce the size of the flow tables. However, the accuracy of prediction has a great impact on the results. If the prediction error is large, additional flow tables will affect the performance of switches and waste network resources instead of improving the quality of services. Therefore, designing a prediction algorithm with high accuracy and reasonable complexity is the key of the new service preloading scheme in this paper. We conducted a follow-up study in the next section.

4 Improved Trajectory Prediction Algorithm

4.1 The Principle of PST Algorithm

In recent years, with the progress of technology, the GPS system in vehicles is more and more accurate, which greatly promotes services based on the location of vehicles. Trajectory prediction has become one

of the hot spots in IOV and some progress has been made. Currently, the algorithms are mainly divided into two categories, one of which is based on the regional division and the other based on processing trajectory data set [16].

The essence of regional division is to divide the traffic network into connected regions according to different methods and map the vehicles' trajectories as the transformation between regions. In the paper [17], the region is meshed and the size affects the accuracy of the prediction. Lei Zhang used k-means clustering algorithm to track the region [18]. The algorithm has high accuracy but the region scalability is poor. At the same time, such methods are predicting within a certain range and will have an impact on the performance and cost of switches in IOV.

Another type is based on the trajectory data set processing. Since the trajectory can be regarded as a discrete series of time-dependent locations, the sequential pattern analysis method is widely used for moving object. However, most mining algorithms based on sequential patterns are modified on the basis of association rules, and there is a problem that the complexity of prediction increases as the number of trajectories increase. On the other hand, the Markov model has been verified to be more in line with the law of vehicle movement [19]. This algorithm has high accuracy, but also has the problem of state space expansion and more complicated training process. And if the parameters are not reasonable enough, the prediction result will have greater errors.

PST (Probability suffix tree), which is actually a variable-length Markov model, is widely used to establish statistical models for complex sequences. Compared with the traditional Markov model, it can process high-order sequence information more effectively and improve the space expansion problem.

Definition 1 (k-Markov Chain). X is a set of states with a finite random variable sequence $\{X_n, n = 0, 1, 2, \ldots\}$. If there is k:

$$P\left(X_{n+1}=j|X_0=i_0, X_1=i_1 \ldots\ldots X_{n-1}=i_{n-1}, X_n=i_n\right)= P\left(X_{n+1}=j|X_{n-k+1}=i_{n-k+1} \ldots\ldots X_{n-1}=i_{n-1} \ldots\ldots X_n=i_n\right). \quad \textbf{(2)}$$

We call this model $\{X_n, n = 0, 1, 2, \ldots\}$ is k-Markov chain. The probability of the next state is determined by the probability of the current state and the past (k-1) known states. In PST, k is dynamic and adaptive, so it is also called variable length Markov model.

The structure of PST model is as follows: It's a non-empty tree based on a sequence set and the edge is marked with a character in the sequence set. The empty root node is represented by ROOT while other nodes by a string which can be generated by traveling from the current node to the ROOT. There is a probability distribution vector that gives the conditional empirical probability of the next character for every node. Fig. 4 shows a PST model by learning the sample "IVOIOVIOV".

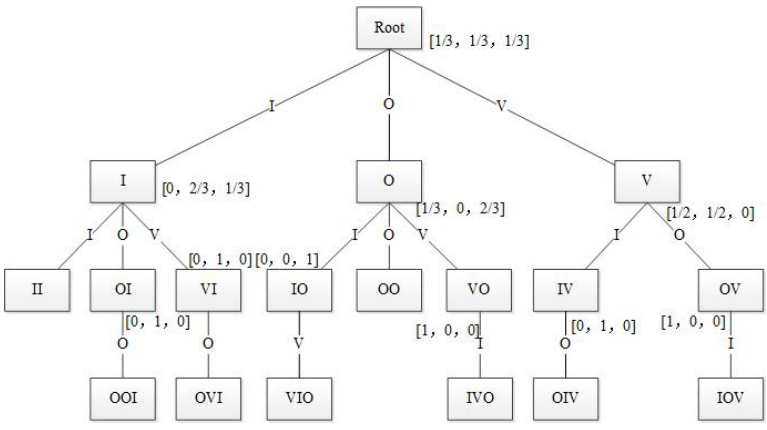

Fig. 4. Probabilistic suffix tree model

Based on the above, we propose an improved PST algorithm with threshold (Threshold-PST), which can fully exploit and train user's historical information and make the prediction dynamically. The thresholds make the model more reasonable, which could reduce the computational complexity.

We simplified the traffic network and made the following definitions:

Definition 2 (Traffic direction graph). It consists of roads, intersection units (IU) and topologies between them. We define G=<C, R> denotes the traffic direction graph, where C={ c_n , n = 0, 1, 2 …} denotes the set of IUs and R={ r_n , n = 0, 1, 2 …} represents the collection of roads (see Fig. 5).

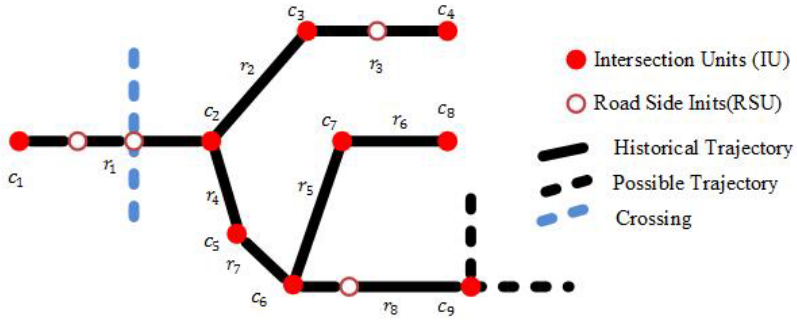

Fig. 5. Traffic direction graph

Definition 3 (Trajectory sequence). We define the nodes ordered by the passing time in the traffic graph as trajectory sequences $T = c_1 c_2 \ldots\ldots c_i$ and the length of each trajectory is L. Traj = $\{T_1, T_2 \ldots\ldots T_n\}$ is a set of trajectory sequences.

Definition 4 (Sub-trajectory sequence). The sub-trajectory sequence $Sub_i(T) = c_i c_{i+1} \ldots\ldots c_n (1 \le i \le n)$ is a subset of the vehicle trajectory sequence, which indicates that the vehicle trajectory sequence removes several earlier intersections.

4.2 Improved Threshold-PST Algorithm

4.2.1 Historical Trajectory Sequence Training

The historical trajectory sequence training is mainly based on constructing the tree. We define the depth (H) of the tree is smaller than the length (L_{max}) of the longest trajectory sequence T_{1max}. We need to traverse all the trace sequences in the sample set starting with the minimum length. The steps to construct the PST are as follows:

Tree initialization. We firstly assume that the tree contains only root node. The probability distribution vector of the root node is the relative frequency of each IU in the intersection node set, which we usually call the unconditional empirical probability $P(c_i)$, which is calculated as equation (3):

$$P(c_i) = \frac{N_{c_i}}{N_c}. \tag{3}$$

Where N_{c_i} represents the number of one IU in the sample set, and N_c represents the number of all IUs.

In order to reduce the computational complexity, we introduce the unconditional confidence of the node, a pre-set threshold P_{min}. We consider the node whose unconditional empirical probability is bigger than this threshold to be a candidate node and remove whose is less.

Tree iteration. After initialization we get an empty root node and a set of candidate child nodes. Then we need to iterate until the depth reaches *H*.

Firstly, we take one node T_l from the candidate set and calculate the probability vector of its subsequent node. If there is an IU c_m whose conditional probability $P(c_m | T_l)$ is bigger than a certain threshold, we add this node to the tree and represent it by $c_m T_l$. We call this threshold the conditional confidence P'_{min}, which is calculated as follows:

$$P(c_m | T_l) = \frac{N_{c_m T_l}}{N_{Traj}}. \tag{4}$$

Where N_{Traj} denotes the number of intersections c_m close to T_l in the trajectory sample set and $N_{c_m} T_l$ denotes the times.

We remove some useless nodes and simplify the tree by setting the node unconditional and conditional confidence. At the same time, taking the actual road conditions into account, the next candidate node of a trajectory sequence must be adjacent to the last intersection unit in T_l and the vehicle will be less possible to repeat the current path. Therefore, it is not necessary to traverse n times (n is the number of all IUs in the set) when calculating the probability distribution vector and we only need to traverse $(|S(c_l)|-1)$ times, where c_l denotes the last intersection in trajectory sequence T_l and $S(c_l)$ represents the set of the roadside units connected to c_l. This method can effectively reduce the algorithm complexity.

The vehicles' trajectory information in the real world is constantly updated, so when a new trajectory sequence appears, we consider the following two cases. If the length of new sequence is less than the depth H, we use conditional confidence as a limit to update. If not, we take its sub-trajectory with length H. The specific process of algorithm is shown in Table 1.

Table 1. Historical trajectory sequence training

Algorithm 1. Historical trajectory sequence training

C_set←Ø;//Candidate child nodes collection;
Threshold-PST←Root←Ø;//ROOT is empty;
initialize P_{root} ;//Probability distribution vector ;
for all C_i in C do
$P_{Root}^{i} = P(c_i)$;
if $P(c_i) \geq P_{min}$ then
C_set.push(c_i);
end if
end for
while C_set ≠ Ø //PST iteration;
T_l = C_set.pop();
initialize P_{T_l} ;
for all C_m in $\{S(c_l) - c_{l-1}\}$ do
$P_{T_l}^{m} = P(c_m \mid T_l)$
if $P_{T_l}^{m} \geq P'_{min}$ then
Threshold-PST← T_l ;
end if
end for
if L_m < H then // New sequence is added;
for all c' in $\{S(c_l) - c_2\}$ do
if $(Suh_H(T_m \mid c') \geq P'_{min}$ then
C_set.push($c'T_m$);
end if
end for
else if L_m > H
for all c' in $\{S(c_l) - c_2\}$ do
if $P(Suh_H(T_m \mid c') \geq P'_{min}$ then
C_set.push($(c'Suh_H(T_m)$);
end if
end for
end if
end while
return Threshold-PST;

4.2.2 Real-time Trajectory Prediction

After the historical trajectory training, we obtain a complete tree for real-time vehicles' trajectory prediction. We firstly get the current trajectory information to build the sequence T_c. Starting from the root node, we aim to find the matching node in reverse order of the current trajectory sequence. If we can't make it, we should use the sub-trajectory sequences of T_c followed by $Sub_2(T_c), Sub_3(T_c), Sub_4(T_c)$ because of the less impact of information long time ago. The most likely next destination is the intersection unit corresponding to the maximum probability in the probability distribution vector at the matching node. The specific process of algorithm is shown in Table 2.

Table 2. Real - time trajectory prediction

Algorithm 2. Real - time Trajectory Prediction
1. for i=1 to l do
2. Q = get tree node that match $Sub_i(T_c)$;
3. if Q is not empty then
4. $P_Q = P(c_i \mid Q)$;
5. end if
6. end for
7. c_{next} = arg max(P_Q);
8. return c_{next} ;

Through these two algorithms, we can achieve trajectory prediction and complete service preloading.

5 Simulation and Performance Analysis

In this paper, we propose a new architecture called SDC-IOV and a services preloading scheme based on trajectory prediction. We need to verify the accuracy and complexity of the algorithm and the superiority of service preloading respectively.

5.1 Trajectory Prediction

In order to verify the superiority of the Threshold-PST algorithm, we design experiments from three aspects of accuracy, time complexity, space complexity. We compare our algorithm with Markov algorithm, random guessing method and the traditional PST algorithm. We use the method described in [20] to simulate a traffic scene using a traffic simulator. The experimental parameters are as Table 3.

Table 3. Experimental parameters

Parameters	Value	Description
N	100	Number of intersection units
M	10000	Number of trajectory sequences
L	1~15	Length of trajectory sequence
P_{min}	0.0005	Unconditional confidence
P'_{min}	0.005	Conditional confidence

In the experiment, 10000 trajectory sequences are randomly divided into five groups evenly. Four groups are used to train and one to predict, which is repeated for five times.

The comparison of the four algorithms' accuracy is shown in Fig. 6. It can be seen from the experimental results that the accuracy of the algorithm is nearly the same when the length of the trajectory is one and when the length is two, the accuracy of our algorithm, traditional PST and Markov algorithm is obviously increased. When the length is longer, due to the adaptive adjustment, our algorithm and traditional PST have better performance while Markov model has small up and down fluctuations. At the same time, because this paper deletes some useless nodes by setting the confidence,

the accuracy is improved compared with the traditional PST algorithm, which is the best in all three algorithms.

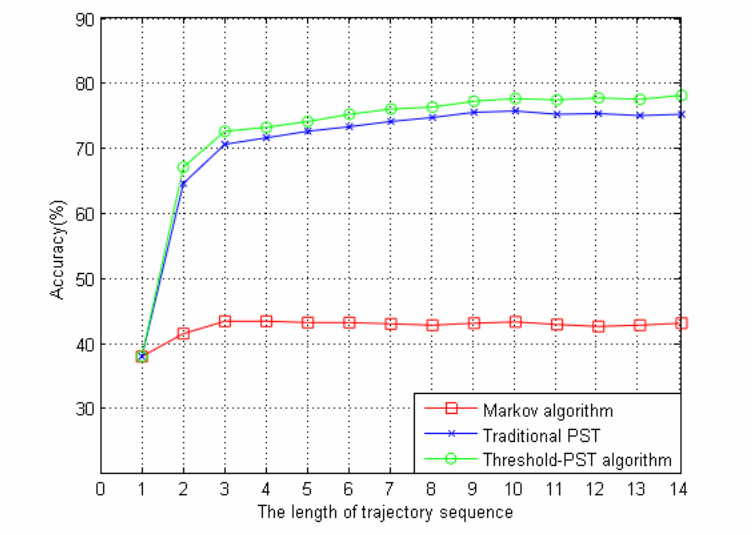

Fig. 6. Trajectory prediction accuracy

Then, the comparison of time complexity is carried out. In the experiment, the running time is used as a reference and the performance of the algorithms is observed by processing different sequence, as shown in Fig. 7. The result shows that our algorithm and traditional PST algorithm outperform the Markov algorithm in time consumption and the gap will gradually increase as the sequence length increases. This is because the Markov algorithm is solved by the transfer matrix, so the time complexity is $O(kn^3)$, k is the order of Markov. While the two PST algorithms use the current trajectory sequence to match the tree model, so the time complexity is O(LM). At the same time, due to the confidence, there is a slight improvement in time performance of Threshold-PST model compared with traditional PST.

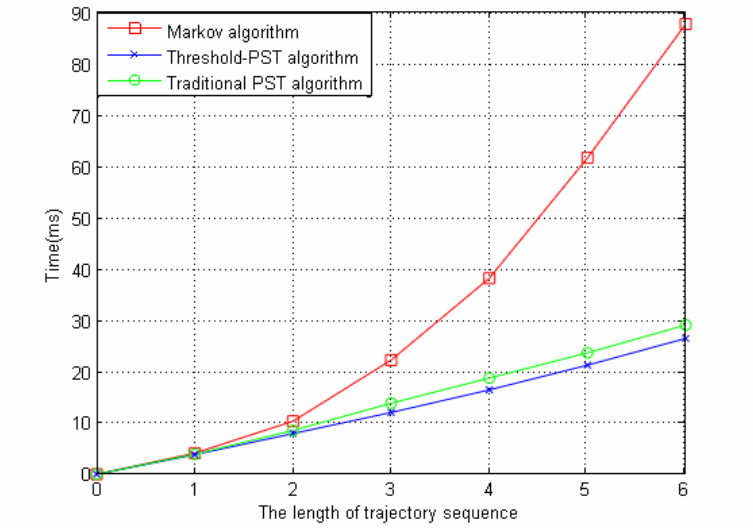

Fig. 7. The time consuming comparison of multiple trajectory prediction algorithms

Finally, taking the system memory occupied during the operation of algorithms as a reference, this paper designs the experiment to compare the space complexity. Experimental results show that the system memory occupied by the three algorithms is basically the same when the sequence length is 2 (see Fig. 8). But the performance of the Threshold-PST algorithm and the traditional PST algorithm is much better than that of the Markov as the sequence length increases. Meanwhile, our algorithm has a weak advantage. By theoretical analysis, we know that the space complexity of the Markov model is $O(kn^2)$. However, our algorithm and traditional PST algorithm are predicted by constructing a tree model, so the space complexity is O(M). Therefore, our algorithm is slightly better in space complexity than the other two algorithms.

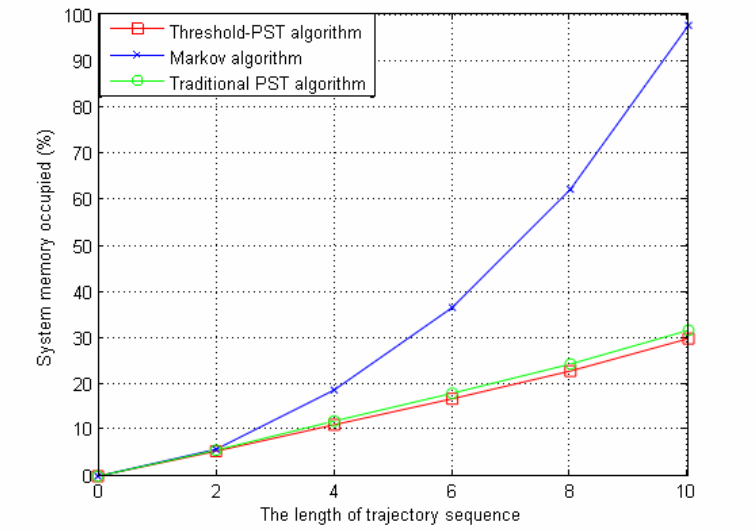

Fig. 8. The memory usage comparison of multiple trajectory prediction algorithms

5.2 Service Preloading

In this paper, the new service preloading model is mainly to alleviate the interaction delay between control plane and data plane. By predicting the trajectory, the real-time services are ensured. So delay is taken as the basis to measure the service quality and the number of flow tables as a standard for switches' performance in the experiment.

The experiment is run on 8-core Cpu and 32Gb memory computers with Ubuntu system. In order to verify the superiority of service preloading, we use Mininet to construct a soft defined network environment to simulate different switch hops and amount of vehicles. Then we observe the number of flow tables and transmission delay.

In the experiment, the scheme in this paper will be compared with the fully preloading scheme and the free response model. The fully preloading scheme is one of the worst deployment schemes. The idea is to make preloading in all switches. While the free response mode is currently used in most practical ways. It can be seen from the Fig. 9 that the number of flow tables increases with the number of vehicles in all three modes. However, the fully preloading scheme obviously increases fast. The main reason is that the flow tables are preloaded in each switch. So with the increasing number of vehicles, the scale of the flow meter will increase rapidly. However, compared with the free response mode, the proposed scheme has a slight decrease. The main reason is that the two flow tables in the traditional progress of service request and service reception are reduced to one. When the service request sent by the vehicle arrives at the corresponding switch, the matching flow tables are already loaded, which deletes the process of sending the service request to the cloud. Thus the performance of switched is guaranteed. This situation will become more and more evident as the number of vehicles increases. Therefore, our scheme can improve the performance to a certain degree compared with the traditional ways.

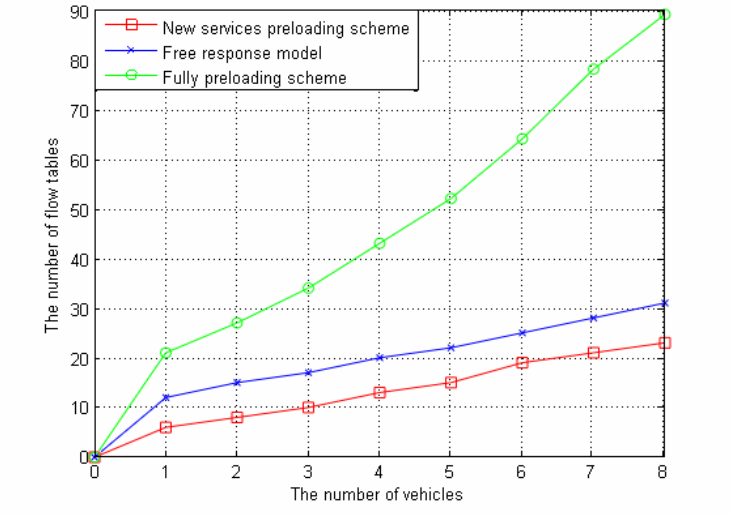

Fig. 9. Number of flow tables

Next, when the number of vehicles is fixed, we observe the delay of obtaining service by simulating different hops. The result is shown in Fig. 10. This paper mainly contrasts with the traditional free response mode. The results show that although the communication delay increases with more vehicles in the network, the delay of services preloading scheme in this paper is always lower than the free response mode. The main reason is that the proposed scheme is based on the prediction of vehicle trajectory. And the required services are preloaded on the driving route in advance, which simplifies the process of connection establishment. And experimental data shows that with the increase of the number of switches, the network topology will be more complicated and the gap between two modes will also become larger and larger.

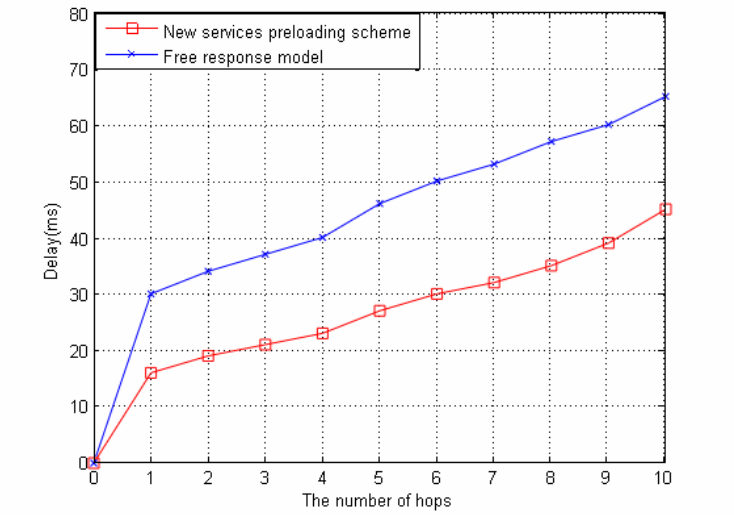

Fig. 10. Transmission delay of different hops

To sum up, on the one hand, the Threshold-PST algorithm has certain advantages in accuracy, time complexity and space complexity. On the other hand, service pre-loading based on trajectory prediction can improve the performance of switches. In different scenarios, the communication delay is the best. Therefore, the scheme in this paper can be well adapted to the computing environment in IOV under the condition of complex traffic network and more intersection nodes. So its validity and superiority are proved.

6 Conclusion

This paper proposes a new architecture based on SDN and cloud computing which can improve the heterogeneity and strengthen the mobile support of traditional IOV. The resource utilization is also improved. At the same time, we propose a services preloading scheme based on trajectory prediction. The scheme regard trajectory prediction and service preloading as the core mechanism in order to limit the size of flow tables and reduce the transmission delay for performance optimization. In order to improve the validity of the scheme, this paper adopts the improved threshold probabilistic suffix tree algorithm. And its computational accuracy and complexity are improved compared with other methods. Finally, the advantages of our method are proved by the experimental simulation and services preloading scheme is proved to be effective.

We admit that there are some limitations in this paper. For example, the construction of traffic map can't fully reflect the characteristics of IOV and the data used for the simulation is simulated. We will focus on it in the future. At the same time, we believe that considering more factors, such as vehicles' type and speed, is an important research direction.

Acknowledgements

This research is supported by the National Natural Science Foundation of China under grant 61271308 and the Fundamental Research Funds for the Central Universities, No.W17JB00060.

References

[1] S. Li, L. Xu, S. Zhao, The internet of things: a survey, Information Systems Frontiers 17(2)(2015) 243-259.

[2] H. Qin, Z. Li, Y. Wang, X. Lu, W. Zhang, G. Wang, An integrated network of roadside sensors and vehicles for driving safety: concept, design and experiments, in: Proc. IEEE International Conference on Pervasive Computing and Communications, 2010.

[3] Z. He, J. Cao, X. Liu, High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility, in: Proc. 2015 IEEE Conference on Computer Communications, 2015.

[4] IEEE Standards Association, IEEE 802.11p-2010-IEEE Standard for International technology-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, July, 2010.

[5] M. Chen, Y. Zhang, L. Hu, T. Taleb, Z. Sheng, Cloud-based wireless network: virtualized, reconfigurable, smart wireless network to enable 5G technologies, Mobile Networks & Applications 20(6)(2015) 704-712.

[6] National Science Foundation, MobilityFirst Future Internet Architecture Project Overview. <http://mobilityfirstwinlab.rutgers.edu/>.

[7] NEBULA.

[8] C. Chaudet, Y. Haddad, Wireless software defined networks: challenges and opportunities, in: Proc. IEEE International Conference on Microwaves, Communications, Antennas and Electronics Systems, 2013.

[9] M. Mendonca, K. Obraczka, T. Turletti, The case for software-defined networking in heterogeneous networked environments, in: Proc. 2012 ACM Conference on CONEXT Student Workshop, 2012.

[10] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, E. Cerqueira, Towards software-defined VANET: architectures and services, in: Proc. 13th Annual Mediterranean Ad Hoc Networking Workshop, 2014.

[11] M. Eltoweissy, S. Olariu, M. Younis, Towards autonomous vehicular clouds, in: Proc. International Conference on Ad Hoc

Networks, 2010.

[12] D. Bernstein, N. Vidovic, S. Modi, A cloud PAAS for high scale, function, and velocity mobile applications - with reference application as the fully connected car, in: Proc. fifth International Conference on Systems and Networks Communications, 2010.

[13] K.M. Alam, M. Saini, A.E. Saddik, Toward social Internet of vehicles: concept, architecture, and applications, IEEE Access 3(2015) 343-357.

[14] K. He, J. Khalid, S. Das, A. Akella, L.E. Li, M. Thottan, Mazu: taming latency in software defined networks, CS Technical Reports, 2014.

[15] P.N. Pathirana, A.V. Savkin, S. Jha, Location estimation and trajectory prediction for cellular networks with mobile base stations, IEEE Transactions on Vehicular Technology 53(6)(2014) 1903-1913.

[16] M. Morzy, Mining frequent trajectories of moving objects for location prediction, in: Proc. International Conference on Machine Learning and Data Mining in Pattern Recognition, 2007.

[17] R. Li, F. Li, X. Li, Y. Wang, QGrid: Q-learning based routing protocol for vehicular ad hoc networks, in: Proc. PERFORMANCE Computing and Communications Conference, 2014.

[18] L. Zhang, B. Yu, J. Pan, GeoMob: a mobility-aware geocast scheme in metropolitans via taxicabs and buses, in: Proc. IEEE INFOCOM, 2014.

[19] J. Krumm, A Markov model for driver turn prediction, Sae World Congress 22(1)(2008) 1-25.

[20] Y. Liu, T. Yan, R. Zhang, Protein sequences analysis based on smoothed PST, in: Proc. International Conference on Bioinformatics and Biomedical Engineering, 2009.