Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 177 (2020) 151–161

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)
November 2-5, 2020, Madeira, Portugal

Vehicular Data Offloading by Road-Side Units Using Intelligent Software Defined Network

Sony Guntuka[a,*], Elhadi M. Shakshuki[a], Ansar Yasar[b], Hana Gharrad[b]

[a]*Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, B4P2R6, Canada*
[b]*Transportation Research Institute, B-3500 Hasselt, Hasselt University, Belgium*

Abstract

The evolution of wide variety of applications that are used by vehicular users includes a lot of data hungry applications. This increases the workload on the cellular networks, thereby delivering poor service to the users. We can overcome this problem by sharing this workload with open wireless networks. As such, this improves the Quality of Service provided by cellular networks. Road-Side Units (RSU) are a wireless network which plays a major role in data offloading. Our approach discusses switching the communication network from cellular to RSU whenever there is an opportunity for a vehicle to offload vehicles data. Busy roads/urban traffic consists of several RSUs with many users. In urban environment, the vehicular user needs to choose an RSU from several available RSUs within the vehicle communication proximity. For seamless connectivity, the delay in network communication because of selecting the best RSU and frequent switching of connection between vehicles and RUSs must be minimized. In this paper, we propose a Smart Ranking based Data Offloading (SRDO) algorithm for selecting an RSU and to improve the Quality of Service. In SRDO algorithm, Q-Learning is utilized for RSU selection. This algorithm is modelled in Software Defined Network controller to deal with the problem of choosing the RSU in an intelligent way for data offloading.

Keywords: Road-Side Unit, Software Defined Network, Reinforcement Learning, Mobile Edge Computing;

* Corresponding author. Tel.: 1-902-585-1524.
E-mail address: 152874g@acadiau.ca

10.1016/j.procs.2020.10.023

1. Introduction

The amount of data generated by the vehicles is a real problem for the cellular networks to handle. A lot of research is done finding various approaches to offload the vehicles data. It is noted that a modern car generates around 25 gigabytes of data every hour [17]. According to the expert forecasts in [17], an autonomous vehicle generates up to 3600 gigabytes of data per hour. Based on the technology evolving every day, we witnessed an increase of autonomous vehicles on the road on daily bases. This huge amount of data cannot be processed on the go to act accordingly in the real time. In our work, we are interested in making sure that the RSUs communicate with the vehicles to process the requested service with an optimal speed given the available bandwidth.

Vehicular users communicate with the cellular networks. Vehicles may connect through various devices such as mobiles, tablets, laptops, etc. The basic need and usage of these devices for applications such as maps for road directions, safety messages for the autonomous vehicles, whether conditions, etc. Also, there are data hungry applications such as Netflix and Facebook that need continuous flow of data downloads from the cellular networks on user demand. This increases the data load on cellular network Base Stations (BSs).

Road-Side Unit is a dedicated short-range communication (DSRC) device for vehicular networks located on the roadside that provides connectivity and information support to passing vehicles, including safety warnings and traffic information. RSUs also act as access points along the road that provide opportunity for the vehicle to switch the network and reduce delays. Software Defined Network (SDN) is a centralized approach to network management, where the underlying network infrastructure is abstracted from applications [12,15]. SDN provides choice in automation and programmability across data centers, campuses, and wide-area networks, which helps to make intelligent decisions. SDN is a promising approach that helps in dynamic optimization processes due to the separation of data and control planes [10]. For intelligent decision-making during data offloading, SDN controller consists of a priority manager and load balancer [16]. Our idea is to make the user comfortable with seamless internet connectivity and Quality of Service. These comforts can be drawn when the load on the cellular stations is reduced; hence, the concept of offloading [13].

The road is split into multiple segments called *placeholders*. It is expected that a user may use different types of applications. All the application service requests are classified into safety and non-safety applications [3]. Safety applications include the messages for automotive safety such as accident warnings. Safety application requests are always given top priority to communicate important details and warnings faster and to avoid accidents.

2. Related Work

A centralized routing scheme for end-to-end unicast communication in VANET is discussed in [19]. The proposed routing scheme has the prediction capability and selects the optimal routing path based on the global information. To adapt to the dynamic changing network topologies, the proposed routing scheme can choose either Vehicle-to-Infrastructure (V2I) or Vehicle-to-Vehicle (V2V) communication. In [25], the authors proposed a new framework of mobile edge cloud-based vehicular networks. Based on the framework, the time consumption and the off-loading cost of various transmission modes are discussed. The authors in [25] also designed a task-file transmission strategy with predictive V2V relay and an optimal predictive combination-mode off-loading scheme. The authors in [22,23,24] proposed different architectures to adopting SDN and Mobile Edge Computing (MEC) together in the VANET environment to increase overall network reliability and scalability under high traffic density conditions. The authors in [4] presented a survey of Machine Learning (ML) approaches applied to SDN. In the survey a detailed background knowledge of SDN with an overview of ML algorithms are also discussed. The authors have shown how both ML algorithms works together on the perspective of traffic classification, routing optimization, QoS prediction, resource management and security. The need for MEC to integrate it with the network is discussed in [5]. A solution towards how this integration technology coexists with Long Term Evolution (LTE) to advance the future of 5G network is presented in [5]. The authors in [8] investigated the synergy between centralized and decentralized (i.e., ad hoc) data scheduling in vehicular ad hoc networks (VANETs) for offloading and balancing the

workloads of roadside units (RSU) in bidirectional road scenarios. They also proposed an algorithm using three mechanisms including a centralized scheduling mechanism at each RSU, an ad hoc scheduling mechanism for vehicles, and a cluster management mechanism.

Machine learning is playing an important role in every field making human work look effortless and faster. The applications of machine learning have attracted a lot of networking researchers too. In [18], the authors proposed a data-driven approach for implementing an artificial intelligent model for vehicular traffic behavior prediction. In this model, the combination of Software Defined Vehicular Network (SVDN) architecture with the machine learning algorithms to model the traffic flow efficiently is introduced. This was achieved by introducing an ingenious approach to find congestion sensitive spots in the VANET by means of clustering algorithm and then predicting the future traffic densities for each spot by recurrent neural networks (RNNs). A protocol is proposed in [21] to store the data in VANETs by transferring data to a new carrier (vehicle) before the current data carrier is moving out of a specified region. In addition, a reinforcement learning-based algorithm is used to consider the future reward of a decision. For data collection, the protocol uses a cluster-based forwarding approach to improve the efficiency of wireless resource utilization. In [2], the authors developed an artificial agent and deployed at the RSU, which will learn a scheduling policy from high-dimensional continuous inputs using end-to-end deep reinforcement learning. A detailed comparison among four machine learning classifiers, namely: Support Vector Machine, C4.5 decision tree, Naïve Bays and Bayes Net classifiers is performed when experimentally applied on a captured network traffic dataset [6]. Similar approaches were adapted and presented in [6] and [7]. In these works, to predict the vehicle positions, the authors experimented with machine learning techniques, including K Nearest Neighbors (KNN), Support Vector Machine (SVM) and Random Forest. A distributed Q-learning algorithm in which each cellular user learns about his local environment and selects the best base station after reaching convergence is proposed in [11]. The authors of [11] introduced a new reward parameter which considers the load of each detected base station, the duration of the vertical handover, the offered gain, as well as the achieved signal-to-interference-plus-noise ratio. The issues and challenges that are faced by many developers to implement applications using RL methods is discussed in [20]. They discussed several issues that are related to dynamic network access, data rate control, wireless caching, data offloading, network security, and connectivity preservation which are all important to next generation networks, such as 5G and beyond.

3. The Proposed Approach

This section discusses our proposed approach on a real time urban traffic environment. This environment is considered as Heterogeneous Vehicular Network (HVN), which means the amount of traffic on the road changes from time to time. For example, the vehicle traffic on the road between 8:00 a.m. to 10:00 a.m. is very high as compared to the traffic during noon time. We are considering these times to improve our proposed algorithm. Figure 1 shows the HVN environment with multiple RSUs, Mobile Edge Computing (MEC) servers, SDN controller, Cloud, and Vehicles.

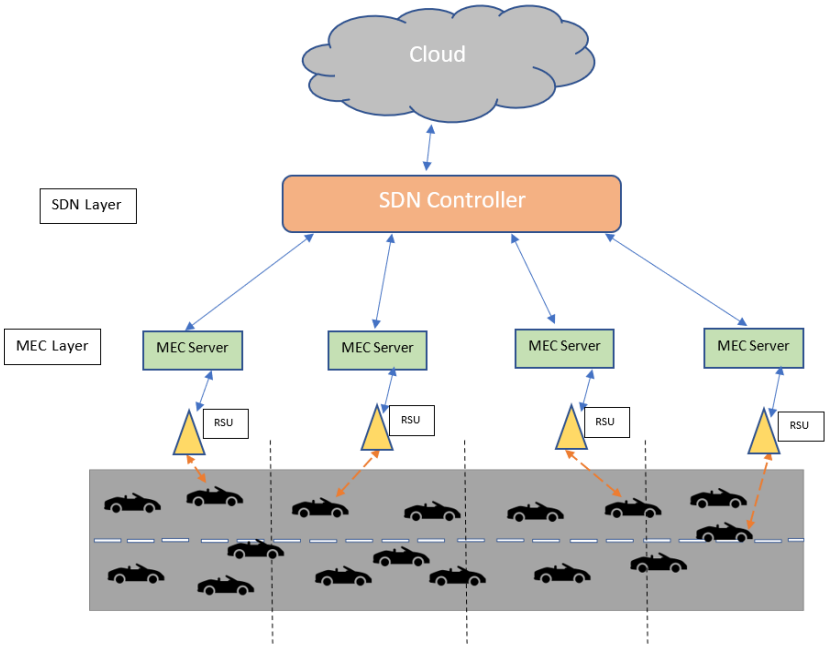

Fig. 1. RSU-MEC environment.

In our approach, we divide the road into segments called placeholders. Each placeholder in this network is in the signal coverage of one or more RSUs. The SDN controller within the network communicates with vehicles through Mobile Edge Computing servers. One of the important actions that SDN Controller must do is to rank the RSUs using RL mechanisms. To make sure all the nodes in the network are connected and make the RSUs intelligent, we utilize Mobile Edge Computing servers [9]. Each RSU is connected to a MEC server. All the MEC servers are connected to the SDN controller and Cloud. This approach provides low latency, location awareness, emergency management, caching, content discovery, and computation. It also improves the quality of services since it is at the proximity to vehicles, and it is used for real-time interaction. The SDN controller performs the RL related computation, while the MEC servers perform the real-time traffic data processing during peak hours of traffic. This makes the data flows easy and reduces the latency. In our approach, we consider low latency is very important because the safety application messages require quick services.

Whenever there is a vehicle passing from one placeholder to another, the RSU communicates with the MEC server with the vehicles real-time data. The real-time data of a vehicle includes its current geographical position, speed, and direction. This information is utilized by the MEC server to choose the next placeholder of the vehicle. This is when MEC server need to choose an RSU for the vehicle to offload the data.

The Q-Learning module in the SDN controller learns from the data at predefined times during the day when the traffic is low. This module ranks each RSU with respect to the corresponding placeholder. Ranking is performed based on the rewards gained by the RSU. This happens when a vehicle is successfully connected and maintained throughout its time in the placeholder. The MEC server stores the information of the placeholders and their respective RSUs in their ranking order. Each placeholder (*Ph*) information is represented by a segment tuple with Ids and a list of the corresponding RSUs, as shown in Fig. 2.

Ph Id	RSU_1	RSU_2	...	RSU_{n-1}	RSU_n

Fig. 2. Placeholder segment tuple.

An RSU serving a vehicle in a placeholder communicates with MEC server. Based on the location, speed, and

direction of the vehicle from the RSU, the MEC server will retrieve the segment tuple of the placeholder and check for the top ranked RSU. If the top ranked RSU is not overloaded, then the vehicle is connected to the RSU as soon as it enters the placeholder. If the top ranked RSU is beyond the threshold load, then the next highest rank RSU is chosen to connect to the vehicle.

3.1. Scenarios

In contrast to the existing ANDSF-based WLAN offloading in the Evolved Packet System (EPS) [1] that chooses the minimal distance RSU, our proposed architecture consists of RL mechanism in SDN controller. It is designed to choose an optimal RSU among the available ones, using the reward computed by the Q-Learning module. Here, an optimal RSU selection depends upon the connectivity with the vehicle in a placeholder. The key for an RSU to receive a reward value depends on the variable Connection Time (*Ct*). *Ct* is defined as the time difference between the time when an RSU and vehicle are connected and disconnected. Based on the user application request, the priority of offloading is chosen, and then a placeholder is selected with the reward values calculated for the local RSUs. By calculating the reward values of all available RSUs, a vehicle *V* selects the RSU with maximal reward value. In the following we adapt the algorithms proposed in our previous work [10] for VANET and we show different cases that are considered and handled by our approach:

Notations for scenarios:

V – Set of vehicles	R_{rank} – Rank of an RSU
R – Set of RSUs	T – time taken for a vehicle to pass the placeholder
PH – Set of placeholders	Ct – connection time of vehicle with RSU in placeholder
V_i – Vehicle at a position	ST – Segment Tuple of each PH
PH_c – Current placeholder of the vehicle	ST_{set} – Set of Segment Tuples of all placeholders in PH
PH_{c-1}, PH_{c+1} – neighbouring placeholders of PH_c	$PH[]$ – Array of RSUs in a PH sorted by ranks
V_{PHc} – Vehicles in the current placeholder PH_c	Th – Threshold load
n – number of vehicles in current placeholder	s – Speed of the vehicle
V_{in} – Vehicle entering the PH_c	l – Location of the vehicle
V_{out} – Vehicle exiting PH_c	d – Direction of the vehicle
S_{rsu} – Set of RSUs available under a PH_c	R_{final} – Chosen RSU to connect with vehicle
R_k– random RSU in set S_{rsu}	

3.2. Algorithms

This section describes our proposed Smart Ranking based Data Offloading (SRDO) algorithms which is a modified version of our previous research work [10]. Algorithm 1 shows how the process of ranking RSUs, while algorithm 2 demonstrates how the selection of the RSUs is performed.

Algorithm 1: Rewarding RSU

1. Input: V, R, PH
2. Output: ST_{set}
3. For each V_i in V // For all the vehicles in set V
   - 3.1 If (V_i belongs to V_{in}) // If the vehicle belongs to a set of vehicles entering current placeholder PH_c
     - 3.1.1 Choose R_k // Randomly choose an RSU
     - 3.1.2 Connect V_i to R_k // Connect the vehicle to the RSU
       - 3.1.2.1 If ($Ct = T$) // If the Connection time is same as time taken for the vehicle
         - 3.1.2.1.1 $Reward_i = reward + 1$ //Update reward of RSU when connected
       - 3.1.2.2 Else If
         - 3.1.2.2.1 $Reward_i = reward - 1$ // Update reward of RSU when disconnected

3.1.2.3 Else
3.1.2.3.1 $Reward_i = reward$ // Update reward of RSU when idle
3.1.2.4 EndIf
3.1.3 Return $Reward_i$ // Assign reward value to RSU R_k
3.2 EndIf
3.3 Map R_k to Ph // Mapping all the RSUs with the final rewards with Phc
4. EndFor
5. Sort Ph from high to low // Sort the RSUs according to ranks and put together as tuple

Algorithm 2: Selecting RSU in MEC Server

1. Input: V, R, ST_{set}
2. Output: R_{final}
3. For each V_i in V // For all the vehicles in set V
   3.1 If (V_i belongs to V_{in}) // If the vehicle belongs to a set of vehicles entering PH_c
      3.1.1 Find Ph // Search for the right Ph of V_i using location l, speed s, and direction d from previous RSU
      3.1.2 Retrieve ST of PH // Retrieve the Segment tuple computed by SRDO algorithm
      3.1.3 For each i in PH // For loop to find the right RSU
         3.1.3.1 If ($Ph[i] < Th$) //If the load on RSU is less than threshold value
            3.1.3.1.1 Connect V_i to $Ph\ [i]$ // Connect the RSU to Vehicle
            3.1.3.1.2 Offload (V_i) // Offload the data to RSU
            3.1.3.1.3 Break // Choose the RSU and exit loop
         3.1.3.2 EndIf
      3.1.4 EndFor
   3.2 ElseIf (V_i belongs to V_{out}) // If the vehicle belongs to a set of vehicles entering PH_c
      3.2.1 Store (l, s, d) // Collect vehicle location, speed and direction
      3.2.2 Disconnect to RSU in Ph_{c-1} // Disconnect from the existing RSU
   3.3 Else
      3.4.1 Connect V_i to R_k // Connect vehicle to random RSU
   3.4 EndIf
4. EndFor

4. Implementation and Results

In this section, we describe the Urban traffic VANET environment with RSU-MEC Architecture proposed in this paper. To demonstrate the algorithms presented in our approach, we use an object-oriented discrete event simulator Objective Modular Network Testbed in C++ (OMNeT++) [26]. It provides infrastructure and the required tools for our planned experiments.

Our experiments are performed on simulated Urban traffic environment. For this purpose, we integrate OMNeT++ and Simulation of Urban MObility (SUMO) [27] with Vehicular Network Simulation (Veins) [28]. SUMO is a road traffic simulator which we utilized to make our environments as realistic as possible. The proposed SRDO algorithm is implemented using a python module utilizing Tensorflow [29]. The output produced from the Python module is fed to the MEC Servers. Then, an optimal RSU is chosen from the available RSUs.

For example, a vehicle A in a placeholder P connects to an RSU X. Here, the Q-Learning passes through multiple steps to assess the ability of the RSU by rewarding based on its performance. The following are a few cases that are discussed to show how the reward is assigned to RSU X.

- *Case 1*: *X* is connected to *A* and continues to serve *A* throughout its time in *P*. *X* did maintain the connection with *A*. Hence, *X* need to be rewarded in this case.
- *Case 2*: *X* is connected to *A* and disconnected due to some reason such as weak signal range of *X*. Although *X* is connected and served *A* for a certain amount of time, the connection is interrupted, and this considered as a downside of *X*. Therefore, in this case, *X* is initially rewarded and later given a negative reward showing *X* may not the best one in *P*.
- *Case 3*: *X* is not connected to *A* due to connection timeout. In this case, there is no connection established between *X* and *A*. Hence, there is no reward given to *X*.

4.1. Learning module operation

Reinforcement learning mechanism works on trial-and-error approach by assigning reward and penalty for the actions taken at each state. We choose a model free policy based RL mechanism called Q-Learning. The elements of Q-Learning algorithm are State, Action, Agent and Reward. Each of these elements in our environments are explained, as follows:

- *State: s* $\in S$, where a set of states, *S,* represents the positions of the vehicle entering the placeholder.
- *Action:* Action *a* belongs to $A(s)$, where the actions are the RSUs with signal range detected in the placeholder.
  $A(s) = \{R_0, R_1, R_2, R_3, \ldots, R_n\}$, where *n* is the number of RSUs detected. The vehicle connected to R_0 enters the current placeholder. So, when $A(s)$ is R_0, the vehicle stays connected to R_0. If the vehicle selects the next action from $R_1, R_2, R_3, \ldots, Rn;$ then, it connects to it.
- *Agent:* Let us assume the vehicle connects to different RSUs (R_1, R_2, R_3, R_4) in its path to offload data while staying in the placeholder.
- *Reward*: *PH Ratio* is calculated for each RSU for a vehicle *V* that connects within the placeholder *P*. Based on the *PH Ratio* of each RSU, the reward value is calculated. The ratio of connection time of a vehicle with RSU in placeholder and time taken by the vehicle to pass the placeholder. A *PH ratio* close to one is the best case which represents that the RSU is connected to the vehicle throughout its time in the placeholder without any interruptions. PH Ratio is calculated using Equation (1).

$$PH\ Ratio = \frac{CTrsu}{Tv} \tag{1}$$

Where, CT_{rsu} represents the amount of time an RSU is chosen and remain intact with the vehicle and T_v is the time taken by a vehicle to travel through the placeholder distance.

Q-Learning mainly focuses on maximizing the future reward value to improve the result in the next steps. This way an optimal path is determined. Therefore, it finds an optimal policy $\pi(s,a)$ to increase the reward in a short time with lesser number of iterations. Here, the next state of the network does not depend on the previous state and the actions get executed based on that state. Bellman derived equations which allows us to start solving Markov Decision Processes (MDPs). Solving MDPs is a step by step process. Initially, we find a set of possible set of states, *S*, and actions *A*.

If we start at state *s* and take an action *a*, we arrive at state, s`. s` is the new state with probability $P_{ss'}^{a}$ and s_t represents the step taken at certain time *t*. $P_{ss'}^{a}$ is the transition probability. It can be defined by Equation (2).

$$P_{ss'}^{a} = P_r(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2}$$

$R^{a}_{ss'}$ is the expected reward that we receive when starting in state, *s*, taking an action *a*, and moving to state, *s`*. This is represented by Equation (3).

$$R^{a}_{ss'} = E(r_{t+1} | s_t = s, s_{t+1} = s', a_t = a) \tag{3}$$

E is the expected immediate reward. With the help of Equations (2) and (3) in this derivation, we get the state value function. Where, the state value function $V^{\pi}(s) = E_{\pi}[r_t | s_t = s]$ is represented by Equation (4):

$$V^{\pi}(s) = E_{\pi}[r_{t+1} + \Upsilon . r_{t+2} + \Upsilon^2 . r_{t+3} + \cdots | s_t = s]$$

$$V^{\pi}(s) = E_{\pi}[\Upsilon^k . r_{t+k+1} | s_t = s] \tag{4}$$

Υ is a discount factor $0<\Upsilon<1$. The discount factor helps getting a finite value for infinite series and it gives greater weight to sooner rewards. This way, we get immediate rewards rather than getting the rewards in a later time. A smaller value of Υ gives more accurate results. Equation 5 is used to calculate $V^{\pi}(s)$ for pulling out the first reward.

$$V^{\pi}(s) = E_{\pi}\left[r_{t+1} + \Upsilon . \sum_{k=0}^{\infty} \Upsilon^k . r_{t+k+2} | s_t = s \right] \tag{5}$$

Equation (6) means that we are expecting the return when continued from state, *s*, by following policy π. The sum of all the transition probabilities and expected rewards are used to determine the expected immediate reward. Hence, the expected reward is calculated using Equation (6).

$$E_{\pi}[r_{t+1} | s_t = s] = \sum_{a} \pi(s, a) \sum_{s`} P^{a}_{ss'} R^{a}_{ss'} \tag{6}$$

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s`} P^{a}_{ss'} [R^{a}_{ss'} + \Upsilon . V^{\pi}(s`)]$$

The importance of Bellman equation for our approach is that it allows us to express the values of states as values of other states. This means that if we know the value of s_{t+1}, we can easily calculate the value of s_t. Q-table will reach convergence values in a finite number of episodes in a certain time frame. This state of Q-table with rewards for all the RSUs in the placeholder are the optimal values. These finalized values of the RSUs along with their placeholder's information is sent to MEC Servers for further usage in real-time traffic environment. A sample data set showing the parameters of Vehicle and RSU which are used for learning is shown in the Table 1.

Table 1. A sample dataset.

Timestamp	Request Size (in bytes)	Vehicle Latitude	Vehicle Longitude	Vehicle Speed	RSU Latitude	RSU Longitude	Request Received	Request Processed
8:32:45.126	1470	33.816008N	84.421498W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.127	1470	33.816009N	84.421499W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.128	1470	33.816010N	84.421500W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.129	1470	33.816011N	84.421501W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.130	1470	33.816012N	84.421502W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.131	1470	33.816013N	84.421503W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.132	1470	33.816014N	84.421504W	21.099998	33.826032N	84.424257W	Y	Y
8:32:45.133	1470	33.816015N	84.421505W	21.099998	33.826032N	84.424257W	Y	N
8:32:45.134	1470	33.816016N	84.421506W	21.099998	33.826033N	84.424250W	Y	Y
8:32:45.135	1470	33.816017N	84.421507W	21.099998	33.826033N	84.424250W	Y	Y
8:32:45.136	1470	33.816018N	84.421508W	21.099998	33.826033N	84.424250W	Y	Y

Based on the above dataset in Table 1, each of the requests are received and processed until one point. At this point

another RSU takes the request and process it. Based on the Timestamp in the dataset, the total time for an RSU to connect and process the requests is determined. In this way, the information of PH Ratios of each RSU in the placeholder is used for learning purposes.

4.2. Performance analysis and results

In the evaluation process, our performance metrics considered to achieve a higher QoS are throughput, latency, and packet loss. Each of these parameters are defined and explain in the following paragraphs.

- *Throughput:* A measure to know how much data is offloaded successfully from vehicle to RSU from start to end of the connection. Let the packets $p_1, p_2, p_3, \ldots, p_n$ are transferred successfully to RSU during times $t_1, t_2, t_3, \ldots, t_n$ respectively for the packets to be delivered to RSU. The throughput is calculated using Equation (7).

$$Throughput = \frac{\sum_{i=0}^{n} pi}{\sum_{i=0}^{n} ti} \tag{7}$$

- *Latency:* Time delay for the vehicle to switch the connection from BS to RSU or from one RSU to another RSU. Let T_E and T_A are the expected time and actual time taken by the vehicle to connect with BS or RSU. The latency is calculated using Equation (8).

$$Latency = TE - TA \tag{8}$$

- *Packet Loss:* We intend to minimize the packet loss to be close to zero, especially for safety messages. Difference between the number of data packets sent by source RSU (D_s) to the number of data packets received by destination vehicle (D_d). Packet loss is calculated using Equation (19).

$$Packet\ Loss = Ds - Dd \tag{9}$$

Our goal is to increase the throughput of the network and to minimize the latency and packet loss values. A higher throughput proves that the connection between the vehicle and RSU is reliable. The values of the latency and the packet loss are indicators to show that by utilizing our proposed approach we expect less delays and loss of data. This makes the safety message requests delivered faster.

During our simulations, we utilized the combination of software VEINS, SUMO and OMNET++. We performed our experiments on VANET environment shown in Fig. 6. In this figure, we have simulated multiple nodes using SUMO; a snapshot is shown in Fig. 7. which represent vehicles moving on a unidirectional road placeholder with a traffic signal and then integrated with OMNET++. To obtain this simulation, multiple xml files and a sumo configuration file is defined. For this purpose, we have used netedit 1.4.0 which is an in-built application in sumo source files. To show the vehicles in the proximity of the RSU components, rsu[0], rus1[0], rsu2[0], rsu3[0] and rsu4[0] are designed in NED file and configured in omnetpp configuration settings file. The coordinates returned by SUMO and TraCI methods in VEINS are different. Hence, we used an internal method called omnet2traci to show the components together as an environment. Other elements used include Cloud, SDN Controller and MEC Servers.

We have defined two variables numSent and numReceived to track the packets of information communicated between RSU and Vehicles, as demonstrated in Fig. 8. Initially, applying the equation (10) is used to calculate the difference between numSent and numReceived. Later percentage of packet loss is calculated. These values are stored in a buffer and saved in vector file of results in the project.

During our experiments, we have calculated the packet loss in TraCIMobility.cc file in VEINS throughout the simulation time of 100 seconds. The python module that incorporates all Equations (3-7) are not implemented as of this writing. However, to demonstrate the results we have assigned reward values and threshold load for each RSU

and stored in MEC Servers during the simulation. Once the MEC Servers acquired information from the python module, all the nodes, RSUs and MEC servers communicate using channel switching and continuous beaconing.

Due to the lack of results from other contributions, we have compared our results with our own findings. During this experiment, we compared the achieved results with and without using machine learning approach, as shown in Fig. 3. The red line shows the results without utilizing machine learning, while the blue line shows the results of our approach when an RSU is ranked with respect to a road segment utilizing Q-Learning. Unlike the traditional approach, our results show a maximum of 7% of packet loss for a very short time and a consistent behavior in the performance.

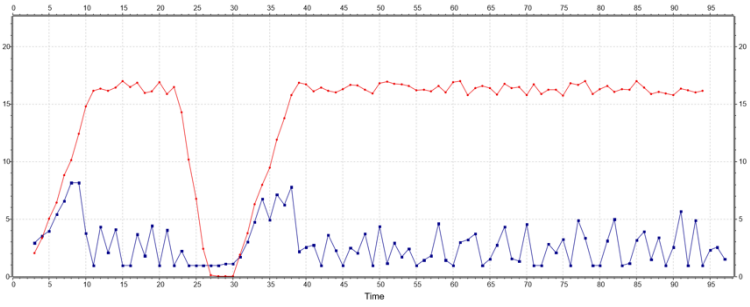

Fig. 3. Time vs. packet loss

5. Conclusions and Future work

This paper discussed the problem of cellular networks due to Urban traffic environments high data load. A brief background on Urban road environment and its infrastructure is also discussed. Our proposed approach is one way to relieve the excessive load on cellular networks using Reinforcement learning. A model free policy-based reinforcement learning called Q-Learning is utilized to choose the optimal RSU which serves vehicles for longer time. For this purpose, we developed an algorithm Smart Ranking based Data Offloading (SRDO) that runs in SDN controller at scheduled times. To run this algorithm, roads are divided into multiple segments called placeholders. This concept of dividing the road into placeholders gave a prior map for computing and analyzing the geographical area. Our experiments are performed by considering a unidirectional road segment. Given the priority to the RSUs based on the reward value of each RSUs available, our simulated vehicles chosen the RSU with highest reward value. The results have shown a noticeable difference by reducing the percentage of packet loss by 9%. This improvement is impacted on the overall network performance by increasing the throughput of the network and decreasing the latency time. Our future research includes finding the ideal size of a placeholder to improve our results in real time environment. We also plan to improve the efficiency of our algorithms with advanced RL techniques such as deep reinforcement learning with double Q-Learning.

References

[1] Joseph Orimolade, Neco Ventura, Olabisi Falowo (2016) “ANDSF-based WLAN offloading in the Evolved Packet System (EPS)” *18th Mediterranean Electrotechnical Conference (MELECON)*

[2] Ribal Atallah, Chadi Assi, and Maurice Khabbaz (2017) “Deep Reinforcement Learning-based Scheduling for Roadside Communication Networks” *15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*

[3] Xin Wang (2011) “Textbook Mobile Ad-Hoc Networks: Applications” *IntechOpen section 2.1*

[4] Junfeng Xie, F. Richard Yuy, Tao Huang and Others (2018) “A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges” *IEEE Communications Surveys & Tutorials 21(1): 393 - 430*

[5] Dario Sabella, Alessandro Vaillant, Pekka Kuure and Others (2016) “Mobile-Edge Computing Architecture” *IEEE Consumer Electronics Magazine 5(4): 84-91*

[6] Muhammad Shafiq, Xiangzhan Yu (2016) "Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms" *2nd IEEE International Conference on Computer and Communications (ICCC)*
[7] Mamoudou Sangare, Soumya Banerjee, Paul Muhlethaler, Samia Bouzefrane (2018) "Predicting Vehicles' Positions using Roadside Units: A Machine-Learning Approach" *IEEE Conference on Standards for Communications and Networking (CSCN)*
[8] Byungjin Ko, Kai Liu, Sang Hyuk Son (2019) "RSU-Assisted Adaptive Scheduling for Vehicle-to-Vehicle Data Sharing in Bidirectional Road Scenarios" *IEEE Transactions on Intelligent Transportation Systems: 1-13*
[9] Chung-Ming Huang, Meng-Shu Chiang, Duy-Tuan Dao and Others (2018) "V2V Data Offloading for Cellular Network based on the Software Defined Network (SDN) inside Mobile Edge Computing (MEC) Architecture" *IEEE Access 6: 17741 - 17755*
[10] Sony Guntuka, Elhadi M. Shakshuki, Siddardha Kaja, Ansar Yasar (2020) "Queue based Vehicular Ad Hoc Network Prognostic Offloading Approach" *The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) 170: 584-593*
[11] E. Fakhfakh and S. Hamouda, IET Communications (2017) "Optimised Q-learning for WiFi offloading in dense cellular networks" *Journal of The Institution of Engineering and Technology 11(15) 2380-2385*
[12] Wei Quan, Kai Wang, Yana Liu, Nan Cheng, Hongke Zhang, and Xuemin Shen (2018) "Software-Defined Collaborative Offloading for Heterogeneous Vehicular Networks" *Wireless Communications and Mobile Computing*
[13] D. Suh, H. Ko and S. Pack (2016) "Efficiency Analysis of WiFi Offloading Techniques" *IEEE Transactions on Vehicular Technology 65(5) 3813-3817*
[14] Leslie Pack Kaelbling, Michael L.Littman (1996) "Reinforcement Learning: A Survey" *Journal of Artificial Intelligence Research 4*
[15] Open Networking Foundation (2012) "Software-Defined Networking: The New Norm for Networks" *ONF white paper*
[16] Ali Akbar Neghabi, Nima Jafari Navimipour and Others (2019) "Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network" *International Journal of Communication Systems 32 (4): 1-26*
[17] Francois Fleutiaux, Director of T-Systems (2018), "Vehicle data is more profitable than the car itself" *Telekom*
[18] Jitendra Bhatia, Ridham Dave, Heta Bhayani (2020), "SDN-based real-time urban traffic analysis in VANET environment" *Journal of Computer Communications 149: 162-175.*
[19] Yujie Tang, Nan Cheng, Wen Wu (2019) "Delay-Minimization Routing for Heterogeneous VANETs with Machine Learning based Mobility Prediction" *IEEE Transactions on Vehicular Technology 68(4) : 3967 - 3979*
[20] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong (2019) "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey" *IEEE Communications Surveys & Tutorials 21(4): 3133 - 3174*
[21] Celimuge Wu, Tsutomu Yoshinaga, Yusheng Ji (2017) "A Reinforcement Learning-Based Data Storage Scheme for Vehicular Ad Hoc Networks" *IEEE Transactions on Vehicular Technology 66(7): 6336 - 6348*
[22] Ammar Muthanna, Regina Shamilova, Abdelhamied A. Ateya and Others (2019) "A mobile edge computing/software ‐ defined networking ‐ enabled architecture for vehicular networks" *Internet Technology Letters Special Article Issue*
[23] Wafa Ben Jaballah, Mauro Conti, Chhagan Lal (2019) "A Survey on Software-Defined VANETs: Benefits, Challenges, and Future Directions" *Networking and Internet Architecture Cornell University*
[24] Xuefeng Ji, Wenquan Xu, Chuwen Zhang, Bin Liu (2020) "A Three-level Routing Hierarchy in improved SDN-MEC-VANET Architecture" *IEEE Wireless Communications and Networking Conference (WCNC)*
[25] Zhang Ke, Mao Yuming, Leng Supeng (2017) "Predictive Offloading in Cloud-Driven Vehicles: Using Mobile-Edge Computing for a Promising Network Paradigm" *IEEE Vehicular Technology Magazine 12(2): 1556-6072*
[26] https://omnetpp.org/
[27] https://www.eclipse.org/sumo/
[28] https://veins.car2x.org/
[29] https://www.tensorflow.org/agents/